AGU ADVANCING EARTH AND SPACE SCIENCES

Water Resources Research

RESEARCH ARTICLE
10.1029/2024WR038554

A Probabilistic Approach to Surrogate-Assisted Multi-Objective Optimization of Complex Groundwater Problems

Reygie Q. Macasieb[1], Jeremy T. White[2], Damiano Pasetto[3], and Adam J. Siade[1,4]

[1]The University of Western Australia, School of Earth and Oceans, Crawley, WA, Australia, [2]INTERA Inc. Austin, Austin, TX, USA, [3]Department of Environmental Sciences, Ca' Foscari University of Venice, Venice, Italy, [4]Commonwealth Scientific and Industrial Research Organisation, Environment, Waterford, WA, Australia

Key Points:

- We present an effective algorithm for solving constrained nonlinear multi-objective optimization problems with less computational effort
- The method applies a new acquisition strategy to efficiently aid sequential retraining of the surrogate model thereby improving its accuracy
- Probabilistic Pareto dominance is used to incorporate uncertainty in evaluating the dominance relationship between emulated positions

Supporting Information:

Supporting Information may be found in the online version of this article.

Correspondence to:

A. J. Siade,
adam.siade@csiro.au

Citation:

Macasieb, R. Q., White, J. T., Pasetto, D., & Siade, A. J. (2025). A probabilistic approach to surrogate-assisted multi-objective optimization of complex groundwater problems. *Water Resources Research*, *61*, e2024WR038554. https://doi.org/10.1029/2024WR038554

Received 8 AUG 2024
Accepted 14 MAR 2025

Abstract Groundwater management involves a complex decision-making process, often with the need to balance the trade-off between meeting society's demand for water and environmental protection. Therefore effective management of groundwater resources often involves some form of multi-objective optimization (MOO). Many existing software tools offer simulation model-enabled optimization, including evolutionary algorithms, for solving MOO problems. However, such analyses involve a huge amount of numerical process-based model runs, which require significant computational effort, depending on the nonlinearity and dimensionality of the problem, in order to seek the optimal trade-off function known as the Pareto front. Surrogate modeling, through techniques such as Gaussian Process Regression (GPR), is an emerging approach to significantly reduce the number of these model evaluations thereby speeding up the optimization process. Yet, surrogate model predictive uncertainty remains a profound challenge for MOO, as it could mislead surrogate-assisted optimization, which may result in either little computational savings from excessive retraining, or lead to suboptimal and/or infeasible solutions. In this work, we present probabilistic Pareto dominance criteria that considers the uncertainty of GPR emulation during MOO, producing a "cloudy" Pareto front which provides an efficient decision space sampling mechanism for retraining the GPR. We then developed a novel acquisition strategy to manage the solution repository from this cloud and generate an ensemble of infill points for retraining. We demonstrate the capabilities of the algorithm through benchmark test functions and a typical density-dependent coastal groundwater management problem.

1. Introduction

As communities heavily rely on groundwater resources that are increasingly under pressure for over use, groundwater managers and stakeholders are more frequently using numerical groundwater models as tools that enable them to assimilate large amounts of data to make informed decisions. Process-based models (PBMs) such as numerical groundwater models have been instrumental in synthesizing complex aquifer structures and properties, boundary conditions, and all external factors affecting groundwater availability. These numerical models have broadened from merely a tool for understanding the effects of interacting environmental processes to an important support tool for guiding critical management decisions (Doherty & Moore, 2020). As decision support tools, they are used for analyzing several "what if" scenarios and predicting the system's behavior under different management strategies.

The scope of management often involves finding a compromise between competing interests of the economy and the environment; however, in some cases, it may include ecology, society, and even culture (Gorelick & Zheng, 2015; Rajanayaka et al., 2021). Moreover, with other issues such as population growth, and other socio-economic and environmental issues in the picture, groundwater management is even more complex and complicated as it must also reconcile these competing interests or objectives. For example, to meet increases in water demand, extraction must be increased, often at the expense of the health of groundwater dependent ecosystems (GDEs). Hence, favoring one objective without sacrificing another is impossible; such compromise is also known as a trade-off. In order to formally assess the optimal trade-off between these competing objectives, multi-objective optimization (MOO) can be applied to a numerical model, resulting in a Pareto optimal set of solutions or a Pareto front (Miettinen, 1998).

Evaluating the Pareto front calls for a robust search algorithm that aims to optimize all objectives simultaneously while optionally adhering to any strict management constraints. Classical mathematical programming techniques such as linear and nonlinear programming (Ahlfeld & Mulligan, 2000; Dantzig et al., 1955; Heidari et al., 1971;

Sawyer & Lin, 1998) cannot handle MOO very well without scalarization (i.e., combining the competing objectives into a single objective through the use of user-defined weights for each objective). While mathematical programming methods often exhibit fast convergence, they generally struggle with finding the global solution to the MOO, especially in the presence of strong nonlinearities. Alternatively, evolutionary, or meta-heuristic algorithms (MHA) offer an effective means of solving MOO problems that are complex and highly nonlinear by testing a large sample of candidate solutions and evolving them into better solutions globally based on simple evolutionary principles observed in nature. Most popular MHAs include non-dominated sorting genetic algorithm (NSGA) (Deb et al., 2000), strength Pareto evolutionary algorithm (Kim et al., 2004), differential evolution (Li et al., 1997), and multi-objective particle swarm optimization (MOPSO) (Coello et al., 2004). MOPSO's memory-like nature based on swarm theory makes it a compelling choice for various applications, including reservoir operation optimization (Baltar & Fontane, 2008; Moradi & Dariane, 2009; SaberChenari et al., 2016) and groundwater management and uncertainty quantification (Alaviani et al., 2018; Siade et al., 2019). Siade et al. (2019) proposed modifications to MOPSO to improve its performance while promoting diversity of solutions along the Pareto front, which was later adopted by White et al. (2022) in a software called PESTPP-MOU. Although MHAs are well-tested and proven effective for a wide range of difficult MOO problems, they are very computationally expensive as they require a very large number of PBM evaluations. PBMs can require tremendous computational resources for a single forward run, let alone hundreds, even thousands, of runs needed for convergence.

Surrogate modeling is an emerging technique that has been tested in various applications (Deb et al., 2020; Forrester & Keane, 2009; Jin et al., 2001) to reduce the number of PBM runs required during management optimization. A surrogate model is a computationally cheaper approximate model that runs much faster than a full PBM; thus, evaluating each potential management scenario is computationally cheap. The speed of executing surrogate models combined with the robust global search mechanism of MHAs can empower more resource management analysis to use formal MOO, which can only lead to more informed decisions in a more timely manner; thus, it has been the subject of recent studies (Gaur et al., 2018; Mozaffari et al., 2022; Siade et al., 2020; Zare & Koch, 2021). Surrogate models are classified as either lower-fidelity or data-driven. The reader is referred to Razavi et al. (2012) and Asher et al. (2015) for a more extensive review of surrogate modeling techniques. Data-driven surrogate models have been increasingly popular in decision support modeling due to their very fast run times and non-intrusive nature that provides flexibility and easy deployment using numerous software options (Asher et al., 2015; Kleijnen, 2009; White et al., 2022). Such models are based solely on input/output data sets drawn from the complex model, often referred to as "training data sets", and employ a machine learning algorithm to approximate the outputs for other/untried inputs.

The accuracy of a data-driven surrogate model depends on the amount and quality of information in the data set used to train it. If not sufficiently sampled, the surrogate model may not yield an accurate emulation of the PBM, which in turn could mislead decision-makers. Conversely, over-sampling of the training data set using the PBM increases the computational demand and decreases the attractiveness of the surrogate model. Therefore, an adaptive strategy for resampling the most informative points is required to ensure that the surrogate model remains sufficiently accurate as the optimization algorithm proceeds, while minimizing the number of PBM evaluations. What qualifies as informative points or infills (Forrester et al., 2008) are those that would lead the search to global optima quickly and efficiently. These are points worth spending the computational budget for PBM runs to update the surrogate model and are found by trading-off exploitation of the known data set and exploration of promising solutions (Jones, 2001). In the case of single-objective optimization, Mo et al. (2017) proposed an adaptive strategy to improve coverage in nonlinear and discontinuous regions in the response surface with minimal PBM evaluations. Siade et al. (2020) utilized swarm theory while other researchers (Frazier, 2018; Gelbart et al., 2014; Wilson et al., 2018) suggested using the Bayesian global optimization (BGO) approach that was first introduced by Mockus et al. (1978) to identify infills through so-called acquisition functions (AQFs). Some examples of AQFs are probability of improvement (PI), expected improvement (EI), simple regret (SR), and upper confidence bound (UCB) (Brochu et al., 2010; Jones, 2001; Snoek et al., 2012). Nevertheless, adaptive resampling is still a matter of active research, and, while it has been studied extensively in single optimization settings, has received little attention for MOO problems.

The application of surrogate modeling for MOO problems is quite limited, where many modern studies still do not perform resampling, and instead use a brute-force (re)training scheme, leading to potentially excessive

evaluations of the PBM (Vahdat-Aboueshagh et al., 2022). Zhao et al. (2020) and Wang et al. (2022) developed a Gaussian process regression (GPR)-based algorithm for petroleum production MOO problems; both used the Reference Vector Guided Evolutionary Algorithm (RVEA) (Cheng et al., 2016) for estimating the Pareto front. Zheng et al. (2022) also proposed a similar approach that uses RVEA with a mechanism to switch between GPR and radial basis function depending on the uncertainties in the predictions. While these studies implemented simple resampling between iterations, they did not take advantage of the efficiency of the exploitation/exploration concept of modern machine learning techniques, for example, through AQFs, and therefore their efficiency could likely be improved. However, the extension of AQFs to MOO problems is still largely unknown; to the author's knowledge, the Expected Hypervolume Improvement (EHVI) criterion (Emmerich et al., 2006) is the only one available in literature. Emmerich et al. (2016) and Yang et al. (2019) demonstrated using EHVI as an infill criterion through MOO numerical experiments in other disciplines. Nevertheless, whether AQF or another resampling criterion was used, these studies implemented a one-at-a-time sampling scheme, similar to BGO for single-objective optimization, which cannot take advantage of parallelized computation. Parallel computing is the only practical way to attain significant gains in wall-time performance by evaluating an entire population of solutions for the PBM simultaneously. Especially now that parallel computing resources have become easier for groundwater modelers, whether through a massively-parallel high performance computing (HPC) systems or a simple commercially available multi-core laptop, it is only prudent that algorithms should adapt to such advancement in computing architecture.

In this study, we develop an acquisition strategy for iteratively resampling an ensemble of the most informative infills so as to guide MOPSO, through an adaptively retrained GPR, to the Pareto front quickly with dramatically less computational burden, that is, minimal PBM runs. Since GPR emulated outputs are not deterministic representations of the PBM, Pareto dominance relationships required by MHAs cannot be directly evaluated with these outputs. Instead, we extend the dominance probability concept described in Khosravi et al. (2021), termed herein as probabilistic Pareto dominance (PPD), to accommodate these noisy predictions, resulting in an iterative "cloud" of Pareto optimal solutions, each consisting of its own unique combination of exploitation and exploration properties. This set of noisy solutions are optimized via MOPSO, using a novel fitness function to address both diversity and optimality at each of MOPSO's "inner iterations", where the surrogate model is subjected to evolutionary MOO. Subsequently, the resulting ensemble of solutions are executed through the PBM in parallel via HPC, and the resulting input/output pairs are added back to the GPR design; this overall process of performing MOPSO on the GPR and executing the resulting ensemble through the PBM comprises a so-called "outer iteration". Outer iterations are repeated until acceptable convergence to the Pareto front is attained. We mainly focus on groundwater management as the motivation for developing the algorithm presented herein; however, as demonstrated by Kollat et al. (2012), MOO can also be applied in multi-objective model calibration, which is also an expensive process with which the proposed algorithm would also be useful for. We show that our method improves computational performance (as measured by PBM evaluations) dramatically over direct use of the PBM, and offers the first fully-parallel ensemble-based algorithm for surrogate-assisted MOO.

2. Process-Based Multi-Objective Optimization

The primary focus of this study is to solve a general constrained MOO problem with the following form (Boyd & Vandenberghe, 2004; Miettinen, 1998):

$$\begin{aligned} &\min_{\mathbf{x}} \boldsymbol{\Phi}(\mathbf{x}) \\ &\text{subject to :} \\ &\mathbf{x} \in \Omega \subseteq \mathbb{R}^{n_d} \\ &\boldsymbol{\theta}(\mathbf{x}) \leq \mathbf{c} \end{aligned} \tag{1}$$

where, $\boldsymbol{\Phi}(\mathbf{x}) = \left(\phi_1(\mathbf{x}), \ldots, \phi_{n_o}(\mathbf{x})\right)$ is a vector of n_o objective functions, $\mathbf{x}$ is a vector of n_d decision variables, Ω is the feasible region for $\mathbf{x}$, $\boldsymbol{\theta}(\mathbf{x}) = (\theta_1(\mathbf{x}), \ldots, \theta_{n_c}(\mathbf{x}))$ is the vector of n_c constraint functions, and $\mathbf{c}$ is a vector of constraint limits. Physically-based numerical groundwater models, that is, PBMs, may be required to quantify elements of $\boldsymbol{\Phi}(\mathbf{x})$ and/or $\boldsymbol{\theta}(\mathbf{x})$.

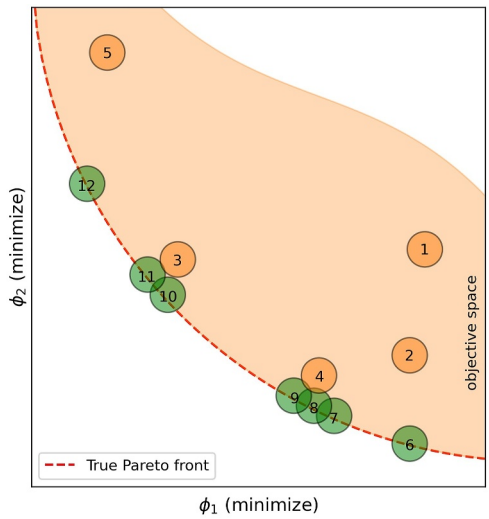

Figure 1. A hypothetical illustration of Pareto dominance using a discrete set of positions. $\mathbf{\Phi}^6$-$\mathbf{\Phi}^{12}$ are better in all objectives than $\mathbf{\Phi}^1$-$\mathbf{\Phi}^5$. $\mathbf{\Phi}^6$ improves ϕ_2 but worsens ϕ_1 when compared to $\mathbf{\Phi}^7$. Since no other feasible solutions are better for all objectives, this trade-off is inevitable, and both are deemed Pareto optimal. The non-dominated positions $\mathbf{\Phi}^6$-$\mathbf{\Phi}^{12}$ comprise the Pareto front.

The solution to the MOO problem is not a single vector of decision variables, but rather a set of decision variable vectors commonly known as a Pareto optimal set and, their corresponding objective function values follow a trade-off function known as the Pareto front (Miettinen, 1998). While classical mathematical programming algorithms have been proven successful in solving single objective optimization problems in groundwater management (Gorelick & Zheng, 2015), they generally perform poorly when two or more objectives need to be simultaneously optimized. This becomes even more challenging when highly nonlinear, discontinuous, non-differentiable functions are involved, which is often the case for groundwater models. As a result, metaheuristic or evolutionary algorithms are perhaps the only means available for solving MOO problems, and have thus received a great deal of attention in the literature (Baltar & Fontane, 2008; Cheng et al., 2016; Gu et al., 2021; Kim et al., 2004; Zitzler & Thiele, 1999).

The vast majority of MHAs in use today operate on an ensemble, or discrete set of solutions, iteratively based on some form of evolutionary concept. They use the concept of dominance to determine which solutions are Pareto optimal at each iteration. The concept of Pareto dominance is discussed briefly as follows. A vector of objective values, written in simplified notation as $\mathbf{\Phi}^i = \left[\phi_1^i, \phi_2^i, \ldots, \phi_{n_o}^i\right]^\top$, corresponds to the i^{th} position in decision space, $\mathbf{x}^i$. Consider n_s realizations of $\mathbf{x} \in \Omega$, resulting in n_s realizations of $\mathbf{\Phi}$ when evaluated through the PBM. A position, $\mathbf{\Phi}^i$, is Pareto dominant over another position, $\mathbf{\Phi}^j$, if and only if, (a) $\forall\, k \in \{1,\ 2, \ldots, n_o\}$, $\phi_k^i \leq \phi_k^j$, and (b) at least one objective in $\mathbf{\Phi}^i$ is strictly better than $\mathbf{\Phi}^j$ (i.e., $\phi_k^i < \phi_k^j$). Pareto dominance is also extended to the corresponding decision variables. For the rest of this text, dominance will be denoted by $\preccurlyeq$ (not to be confused with matrix/vector inequalities as defined by Boyd and Vandenberghe (2004)), and therefore we say $\mathbf{\Phi}^i \preccurlyeq \mathbf{\Phi}^j$. In Figure 1, $\mathbf{\Phi}^2$, $\mathbf{\Phi}^3 \preccurlyeq \mathbf{\Phi}^1$ because they improve on both objectives simultaneously, while $\mathbf{\Phi}^4 \preccurlyeq \mathbf{\Phi}^2$ but $\mathbf{\Phi}^4 \not\preccurlyeq \mathbf{\Phi}^3$.

When a position i is not dominated by any other positions, it is said to be non-dominated, that is, each position known thus far has at least one objective value inferior to that of position i. The global set of non-dominated positions and their corresponding decision variables comprises the Pareto optimal set $\mathcal{P}$ of solutions. These solutions collectively sample the Pareto front, which describes the best trade-off between objectives that decisions can have. To ensure that the front is well-defined by these discrete points, one would want these positions to be diverse enough to cover the entire front. For example, positions $\mathbf{\Phi}^7$-$\mathbf{\Phi}^9$ and $\mathbf{\Phi}^{10}$-$\mathbf{\Phi}^{11}$ in Figure 1 are relatively close, and thus considerably redundant, and they leave a gap between their clusters. Thus, to promote diversity, one would want these positions to be spread out across the front evenly.

The primary objective of MHAs is therefore to employ dominance relationships iteratively to arrive at the globally optimal Pareto front, while promoting diversity in the final set. The relative performance of various MHAs is quite problem-specific. While we have selected MOPSO in this study due to both, the existence of readily available software with a parallel run manager (Siade et al., 2019; White et al., 2022) and the fact that preliminary simulations showed better performance over differential evolution, it's important to point out here that our overall methodology is applicable to any MHA and that no one MOO algorithm is superior to all others in all settings.

Since MHAs are robust global search techniques they are quite effective at avoiding local minima, but they do require hundreds, if not thousands, of PBM runs to achieve convergence. In the case of groundwater modeling, each forward run of the PBM may take hours or days of CPU time especially for regional-scale models employed in management and policy-making, for example, Siade et al. (2017, 2020). This limitation is a major drawback when it comes to MOO for real-world management problems. However, surrogate modeling, in particular data-driven supervised machine learning techniques, may hold the key to overcoming this limitation.

3. Surrogate-Assisted Multi-Objective Optimization

Using surrogate models to reduce the burden of expensive optimization runs has been the subject of various applications and reviews (Asher et al., 2015; Deb et al., 2020; Forrester & Keane, 2009; Jin et al., 2001; Razavi et al., 2012). The overarching idea of surrogate-assisted optimization is to directly replace the PBM with a much faster, approximate surrogate model to either optimize the objective functions themselves, or maximize AQFs, which take surrogate model predictions of objective function values as inputs, to sample promising regions of decision space while considering emulation error, for example, BGO (Mockus et al., 1978).

While much focus has been dedicated to single objective optimization in this regard, little has been developed for MOO problems. As the surrogate model is merely an approximate model that mimics the PBM, its emulations would naturally have approximation errors. Our preliminary tests conducted during the early development stages of the algorithm where the Pareto dominance criteria are directly applied on the emulated positions resulted in suboptimal solutions or the algorithm struggled to find solutions in some segments of the true front, particularly where there is discontinuity, and at the ends of the front (see Figures S1–S6 in Supporting Information S1). Very little work has been done on developing AQFs for MOO, that formally consider emulation error, and what little has been done (Emmerich & Deutz, 2018; Emmerich et al., 2016) operates in more of a sequential manner and is not readily amenable to parallel computation and thus still arguably expensive in terms of overall wall time. In this study, we propose a new AQF technique for surrogate-assisted MOO (SAMOO) that maintains the ensemble nature of MHAs and an associated fitness function that allows for discrimination between non-dominated solutions.

3.1. Probabilistic Pareto Dominance

Considering surrogate model approximation error as a source of uncertainty, a multi-objective emulated position is actually characterized by a multivariate probability distribution likely centered on the emulated value. This probability distribution describes the possible values of the objectives, if they were computed through the PBM, using the same decision variables as the surrogate. Without loss of generality, this concept will be illustrated herein through a two-objective optimization problem. Taking into account the surrogate approximation error, the true objective values would fall (with a certain probability) within a confidence area around the emulated position, which is hypothetically illustrated as a hyperellipsoid in Figure 2. In this context it is more challenging to decide which emulated positions are dominating, and thus how to proceed with the MOO algorithm. In fact, the hyperellipsoids from solutions in proximity to one another can overlap; thus, an emulated position that appears to dominate another may actually be worse, and become dominated, when confirmed through evaluation of the solution with the PBM.

Given such ambiguity, positions with significantly overlapping confidence hyperellipsoids should be deemed non-dominant until confirmed by the PBM; Wang et al. (2022) proposed using a surrogate model as a screening strategy based on the concept of overlapping confidence intervals as follows: At a particular iteration, the positions produced by the MHA are tested with a surrogate model, and if their confidence ellipsoids indicate that they would likely be dominated if run through the PBM, they are discarded. While this screening approach improves the efficiency of MOO, it does not utilize the surrogate model to search for promising regions of the decision space. To do this, one would need to estimate the probability that some emulated position i would dominate another position j. Then, position j would only be considered to be dominated if this probability is sufficiently high. As such, the probability of dominance (PD) is simply defined as the probability that position i dominates position j (Khosravi et al., 2021):

$$\mathrm{PD} = P\left(\hat{\mathbf{\Phi}}^i \preccurlyeq \hat{\mathbf{\Phi}}^j\right) = \prod_{k=1}^{n_o} P\left(\hat{\phi}_k^i \leq \hat{\phi}_k^j\right) \tag{2}$$

where $\hat{\mathbf{\Phi}}$ denotes an emulated position (as opposed to $\mathbf{\Phi}$ that denotes a deterministic position). Calculating Equation 2 requires an estimation of the uncertainty of emulated positions, which can be challenging for most machine learning techniques. Some studies (Fieldsend & Everson, 2005; Hughes, 2001; Khosravi et al., 2021) explored the concept of PD for MOO with uncertainty caused by modeling error or decision variable noise, and proposed methods for quantifying objective uncertainty without the presumption of an underlying distribution for the objective functions. We extend this concept of PD in machine-learning surrogate modeling by using GPR to

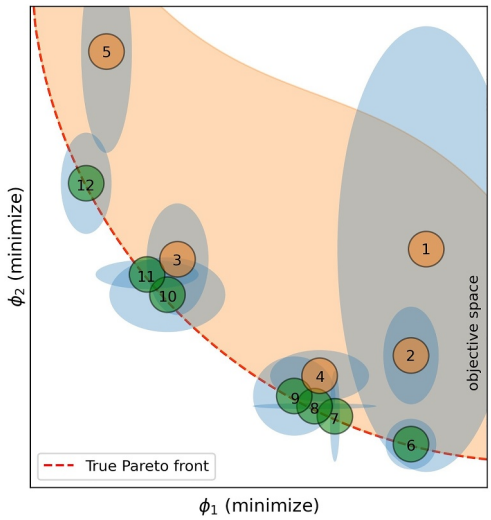

Figure 2. The true location of an emulated position in objective space are likely to be within the confidence ellipsoids. Probabilistic Pareto dominance incorporates this region of uncertainty, for example, $\hat{\mathbf{\Phi}}_3$ and $\hat{\mathbf{\Phi}}_4$ could possibly dominate some of the green positions; thus, both are considered non-dominated even though their emulated means appear dominated by other nearby emulated means.

emulate objective functions, which is currently the only available surrogate modeling technique that offers a straightforward closed-form expression for estimating the variance of emulated predictions by exploiting its underlying Gaussian assumptions.

3.2. Surrogate Modeling via GP Regression

Rasmussen and Williams (2006) discussed the concepts of GPR, which are briefly introduced here. A Gaussian process is a collection of random variables that have a joint Gaussian distribution which can be completely described by a mean function and a covariance function. GPR fits Gaussian random functions through some observations comprised of input-output (IO) pairs, $\mathbf{y}^{(i)} = f\left(\mathbf{x}^{(i)}\right), i = 1, \ldots, n_s$, collected in the so-called training data set or design sites, $D = \left(\mathbf{x}_c, \mathbf{y}_c\right)$. In our model, the design sites are n_s realizations of the decision variables $\mathbf{x} \in \Omega$ and the corresponding n_s values of $\mathbf{y}$ when evaluated through the PBM. Note, the variable $\mathbf{y}$ is generalized here but can be any PBM-derived constraints or objectives (i.e., $\boldsymbol{\theta}$ or $\mathbf{\Phi}$ in Equation 1). GPR assumes that the values of $\mathbf{y}$ corresponding to a decision variable outside the training data set, indicated with $\mathbf{x}^*$, has a joint Gaussian distribution with mean $\bar{f}(\mathbf{x}^*)$ and covariance that is expressed through a correlation function κ depending on the distance between $\mathbf{x}^*$ and the training data set, then

$$\mathbf{y} \sim \mathcal{N}\left(\bar{f}(\mathbf{x}^*),\ \tau^2\kappa(\mathbf{x}^*, \mathbf{x}_c)\right) \tag{3}$$

where τ is a scale parameter needed to express variability of the observed IO pairs in the posterior distribution upon GP regression.

Without going through the derivations, which can be found in numerous studies including the textbook by Rasmussen and Williams (2006), the mean prediction of a GPR assuming a Gaussian correlation function is as follows,

$$\hat{y} = \bar{f}(\mathbf{x}^*) = \mathbf{r}^\top \mathbf{R}^{-1} \mathbf{y}_c \tag{4}$$

where $\hat{y}$ is the GPR emulated mean prediction, $\mathbf{y}_c$ is the vector of known outputs in the training data set, $\mathbf{R}$ is the correlation matrix of the training data set, $\mathbf{x}_c$, and $\mathbf{r}$ is the correlation vector between $\mathbf{x}^*$ and $\mathbf{x}_c$. The expression for the estimated covariance of the emulated prediction, denoted as $\hat{v}$, is as follows,

$$\hat{v} = \hat{\tau}^2\left(1 - \mathbf{r}^\top \mathbf{R}^{-1} \mathbf{r}\right) \tag{5}$$

The convenience of this expression for variance is critical when addressing the reliability and EI of an emulated prediction during optimization.

3.3. Probabilistic Pareto Dominance Using GPR

Because of the underlying Gaussian distribution in Equation 3, we can now derive a formula for PD in Equation 2. Since $\hat{\phi}_k$ is the emulated objective k through GPR, following Equation 5,

$$\begin{aligned} \hat{\phi}_k^i &\sim \mathcal{N}\left(\bar{\phi}_k^i, \hat{v}_k^i\right) \\ \hat{\phi}_k^j &\sim \mathcal{N}\left(\bar{\phi}_k^j, \hat{v}_k^j\right) \end{aligned} \tag{6}$$

Then, the probability of position i being better than position j with respect to objective k can be computed as follows (using basic probability rules):

$$P\left(\hat{\phi}_k^i \leq \hat{\phi}_k^j\right) = P\left(\hat{\phi}_k^i - \hat{\phi}_k^j \leq 0\right) = \Psi\left(-\frac{\bar{\phi}_k^i - \bar{\phi}_k^j}{\sqrt{\hat{v}_k^i + \hat{v}_k^j}}\right) \tag{7}$$

since,

$$\hat{\phi}_k^i - \hat{\phi}_k^j \sim \mathcal{N}\left(\bar{\phi}_k^i - \bar{\phi}_k^j, \hat{v}_k^i + \hat{v}_k^j\right) \tag{8}$$

where $\Psi(\cdot)$ is the cumulative distribution function of the standard normal distribution.

To incorporate all objectives in computing the PD, Equation 2 is written as follows:

$$PD_{i,j} = P\left(\hat{\mathbf{\Phi}}^i \preccurlyeq \hat{\mathbf{\Phi}}^j\right) = \prod_{k=1}^{n_{do}} \Psi\left(-\frac{\bar{\phi}_k^i - \bar{\phi}_k^j}{\sqrt{\hat{v}_k^i + \hat{v}_k^j}}\right) \tag{9}$$

Using the PD concept, we can generalize the conventional Pareto dominance definition in Section 2 to provide a PPD criteria as follows: consider n_s realizations of $\mathbf{x} \in \Omega$ for the optimization problem of Equation 1, resulting in n_s emulated positions, $\hat{\mathbf{\Phi}}$, when evaluated through the surrogate model. An emulated position, $\hat{\mathbf{\Phi}}^i$, is Pareto dominant over emulated position $\hat{\mathbf{\Phi}}^j$, if and only if $PD_{i,j} \geq \beta$, where $0 \leq \beta \leq 1$. Otherwise, solutions $\hat{\mathbf{\Phi}}^i$ and $\hat{\mathbf{\Phi}}^j$ cannot be distinguished based on their emulated objective values.

Surrogate-assisted-MHAs can employ PPD iteratively to obtain positions that are likely to be Pareto optimal. This process happens in the algorithm's "inner iterations" which is presented in Section 3.6. However, instead of the positions appearing to follow a curve that approximates a traditional Pareto front, there will be a fuzzy set of positions as the front will now include some solutions that would have been dominated had the emulated positions been considered deterministic, for example, $\hat{\mathbf{\Phi}}^3$ and $\hat{\mathbf{\Phi}}^4$ in Figure 2. This cloudy emulated front will be referred herein as Pareto cloud. The user-specified parameter β sets the confidence level of whether a slightly worse (or a slightly better) emulated position is dominant over another, given the probability that it could actually be better (or worse). A value closer to 1.0 allows more exploration for the MHA, which allows for a thicker Pareto cloud as a result of more emulated solutions being deemed non-dominated, and vice versa for small values of β.

When two positions have very low or zero uncertainty, either because they are PBM-evaluated or the GPR is already well-trained to emulate said positions accurately, the corresponding PD can be approximated from Equation 9 as follows:

$$PD_{i,j} = \prod_{k=1}^{n_{do}} \lim_{\hat{v}_k^i + \hat{v}_k^j \to 0} \Psi\left(-\frac{\bar{\phi}_k^i - \bar{\phi}_k^j}{\sqrt{\hat{v}_k^i + \hat{v}_k^j}}\right) \tag{10}$$

Thus,

$$PD_{i,j} = \begin{cases} 0 & \text{if for any } k \in \{1, 2, \ldots, n_{do}\}, \quad \bar{\phi}_k^i - \bar{\phi}_k^j \geq 0 \\ 1 & \text{if for all } k \in \{1, 2, \ldots, n_{do}\}, \quad \bar{\phi}_k^i - \bar{\phi}_k^j < 0 \end{cases} \tag{11}$$

which is equivalent to the original deterministic criteria for Pareto dominance. In other words, the PPD criteria can still be applied to deterministic positions by setting their variances to a very small number, which results in a very thin Pareto cloud that resembles a traditional Pareto front. Therefore, PPD is a generalization of Pareto dominance under uncertainty.

3.4. Infill Acquisition

The application of PPD within an MHA provides the basis for sampling a diverse set of potential solutions, generating a large ensemble of positions in the Pareto cloud to support retraining of the surrogate model, which requires running the PBM to obtain additional GPR training pairs to improve the surrogate model accuracy for the next round of sampling, and so on. Such an ensemble approach for this infilling process takes advantage of parallel computation that is becoming more readily available to researchers and practitioners. However, many of these samples may be redundant with one another, which may render many of them as uninformative in terms of how much they can help SAMOO improve upon the current front; therefore, running the PBM with all of them may be wasteful. Thus, we need a method to discriminate the most informative infills from the rest of the cloud.

3.4.1. Repository for Non-Dominated Positions

Embedded in many MHAs is a mechanism for recording non-dominated solutions at every iteration called a repository or archive which we adopt in this study in two ways: (a) As a container of accurate non-dominated positions in each outer iteration, and (b) to facilitate the improved selection of a candidate infill ensemble during each inner iteration of the SAMOO analysis. The former, which will be referred as outer repository, is generated from a conventional Pareto dominance evaluation of deterministic positions, while the latter, referred herein as surrogate repository, results from PPD evaluation of emulated positions.

When the inner iterations are initialized, outer repository positions are loaded into the surrogate repository; then at any given inner iteration, PPD evaluation is performed on the combined set of the current surrogate repository positions and swarm population to update the surrogate repository. As the algorithm progresses, the surrogate repository would naturally grow in size. Many MHAs prescribe that a maximum repository size, n_{rep}, must be specified by the user. The same should apply here for SAMOO, that is, if the number of likely non-dominated solutions by PPD evaluation exceeds the repository size, some of these positions must be discarded until the threshold is reached. Many of the MHAs in literature (Coello et al., 2004; Deb et al., 2000; Kim et al., 2004; Siade et al., 2019) use a fitness function as a metric for managing repository size to maximize diversity, such that the repository is re-subjected to Pareto dominance and diversity analyses to cull dominated solutions. The fitness functions used for the culling are based on neighborhood crowding distance that measures the "loneliness" of a candidate solution in the front, so that lonelier positions are more likely to persist in the repository. The repository is filled up to n_{rep}, after which positions with fitness values less than the last repository position are discarded.

While distance-based fitness is effective in promoting diversity among the repository positions for deterministic MOO, it does not take into account the crowding uncertainty due to approximation errors when using a surrogate model within the SAMOO inner iteration loop. This lack of consideration could be addressed by computing the expected distance between positions, in a probabilistic sense. To test this, we have performed several MOO experiments on benchmark test functions (Appendix A), using expected distance between two emulated points, as discussed in Mathai and Provost (1992) (Equation 12), for evaluating crowding-distance-based fitness of positions in the Pareto cloud. We can assume that relative distances between positions in objective space can be represented as the sum of the squares of the distances between each objective function value. Since each emulated objective function value is a normally distributed random variable, such a product will be χ^2-distributed, whose expected value is as follows,

$$E(D_{i,j}) = \|\hat{\mathbf{\Phi}}^i - \hat{\mathbf{\Phi}}^j\|_2 + (\hat{v}_i + \hat{v}_j) \tag{12}$$

However, employing such a measure of distance in a fitness function biases the algorithm toward extreme exploration as highly uncertain points are deemed less crowded and are therefore assigned a high fitness value over more certain points, as evident by the second term in Equation 12. Employing this criterion for crowding distance calculations actually results in divergence of the algorithm. This outcome is demonstrated for the ZDT benchmark problem (Appendix A) is provided as supporting information (Figure S7 in Supporting Information S1). Therefore, when emulation uncertainty is present, it may be most effective to develop a fitness function that promotes not only diversity, but also simultaneously respects Pareto optimality, to mitigate the potential for extreme exploration and thus divergence away from the Pareto front.

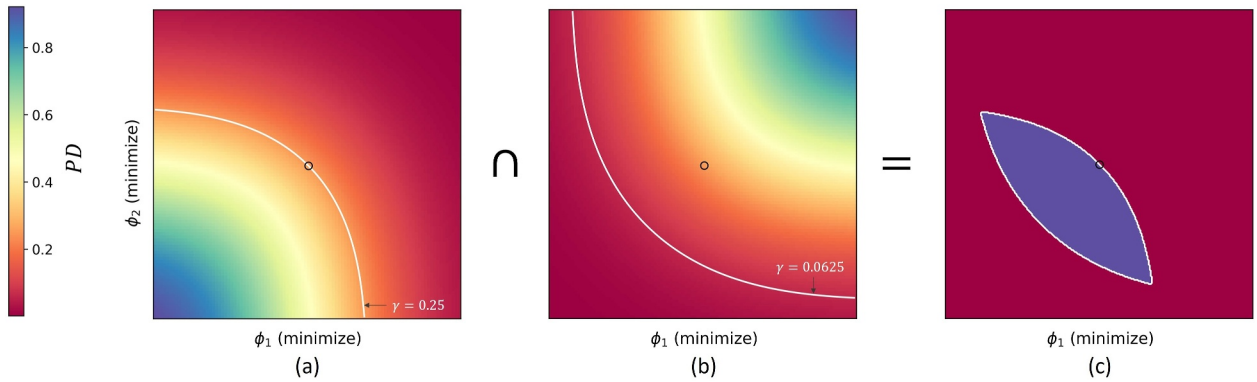

Figure 3. This shows the distribution of the values of (a) $PD_{j,i}$ and (b) $PD_{i,j}$ of any $\hat{\mathbf{\Phi}}^j$ around $\hat{\mathbf{\Phi}}^i$ (represented by a black hollow circle at the center), with each points given a fixed variance for illustration purposes. The domains shown are discretized and probability of dominance (PD) values are computed for each point centered in the grid cell (representing a $\hat{\mathbf{\Phi}}^j$ in each cell) and illustrated by the color gradients. (c) Shows the region of the neighborhood (purple area) of i where another position j would be included in the fitness function (Equation 14). This is only a simplified hypothetical representation for illustration purposes, as PD distributions will vary depending on the uncertainties of both positions under consideration.

3.4.2. Fitness Function for Repository Management

We can take advantage of the PD concept to incorporate optimality alongside crowding (i.e., loneliness) of positions in the fitness function. Consider some position in the repository, $\hat{\mathbf{\Phi}}^i$. Suppose instead of evaluating the probability of $\hat{\mathbf{\Phi}}^i$ dominating some other position, $\hat{\mathbf{\Phi}}^j$, we consider the probability of $\hat{\mathbf{\Phi}}^i$ being *dominated by* $\hat{\mathbf{\Phi}}^j$. Positions with high potential for dominating $\hat{\mathbf{\Phi}}^i$ are either clustered nearby $\hat{\mathbf{\Phi}}^i$, or are further away, but simply likely to dominate $\hat{\mathbf{\Phi}}^i$. Therefore, the more positions around $\hat{\mathbf{\Phi}}^i$ with high PD, the lesser the fitness should be for $\hat{\mathbf{\Phi}}^i$, as it either resides in a cluster (non-diverse) of positions in close proximity to each other, or it has less chance of being Pareto optimal when run through the PBM.

Such a calculation is straightforward using Equation 9 by simply rearranging terms. If we set a limit, γ, such that we consider only those positions around $\hat{\mathbf{\Phi}}^i$ where $PD_{j,i} \geq \gamma$, then as shown in Figure 3a, these points are likely located below the white curve ($\gamma = 0.25$). We then count up the positions that meet this criteria, N_i, and assign a fitness value, f_i, that varies inversely with N_i, such that, positions that are more likely to remain non-dominated are given a higher fitness and therefore have better chances of persisting in the repository.

As mentioned, counting positions that satisfy $PD_{j,i} \geq \gamma$ alone does not guarantee that only nearby positions would be accounted for in N_i. While this can be seen as a good thing in general, it can be counterproductive when the emulated positions in question have a weak Pareto optimality relationship but have relatively large uncertainties. For example, as demonstrated in Figure 4a, when evaluating the PD for the positions on the right, nearly every other position meets the criteria, simply because the positions on the right are much greater in ϕ_1 but similar in ϕ_2, that is, the Pareto cloud is relatively noisy and largely horizontal in shape. If N_i alone were used to define fitness in this case, the MHA will eventually discard positions on the right, driving the ensemble toward the left until everything collapses to the left corner of the Pareto cloud. This is clearly undesirable, and therefore some additional consideration is warranted.

To overcome this drawback, the fitness function should be able to discern which among the positions that satisfy $PD_{j,i} \geq \gamma$ are actually close to $\hat{\mathbf{\Phi}}^i$, but avoid the spurious issues introduced by retaining weakly Pareto optimal solutions. Such positions would therefore also have some, albeit smaller, chance of being dominated by position $\hat{\mathbf{\Phi}}^i$. Hence, among the positions that satisfy $PD_{j,i} \geq \gamma$, the fitness function should only consider those that also satisfy $PD_{i,j} \geq \gamma'$ in its count, N_i; where, γ' is a threshold smaller than γ,

$$N_i = |\{j \in \{1,2,\ldots,n_s\} : i \neq j \wedge PD_{j,i} \geq \gamma \wedge PD_{i,j} \geq \gamma'\}| \tag{13}$$

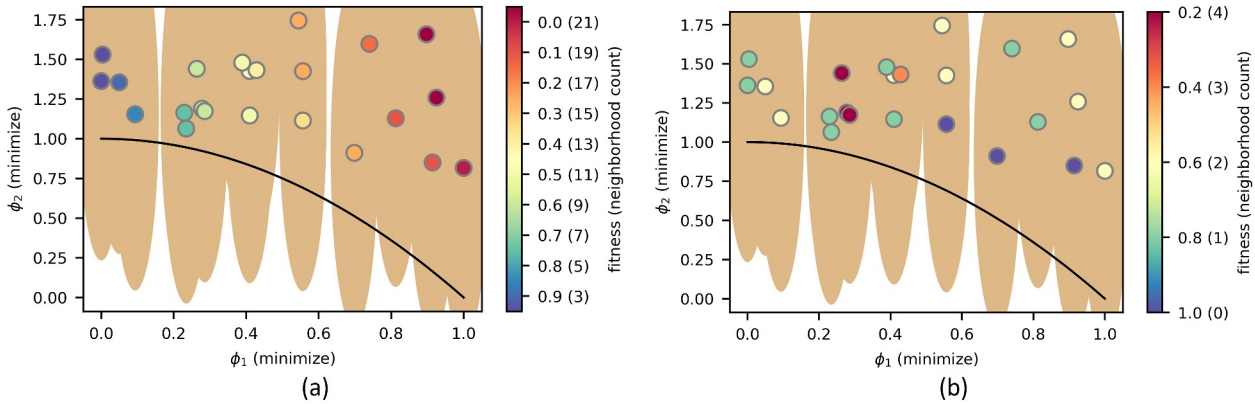

Figure 4. In a hypothetical situation where the black curve represents the true Pareto front and positions are spread out in the cloud, the fitness value assigned depending on $PD_{j,i}$ alone does not seem to reflect the loneliness of a position (a). When $PD_{i,j}$ is added as a qualifier for the neighborhood of $\hat{\mathbf{\Phi}}^i$, the fitness values reflect both optimality and loneliness of each position (b).

The parameter γ' can be set to a low value less than γ such that, as shown in Figure 3b, only neighboring positions above the white curve ($\gamma' = 0.0625$) are counted. Based on the some experimentation, setting $\gamma' = \gamma^2$ and $\gamma = 0.25$ works very well in most cases. A higher value of γ contracts the neighborhood region of $\hat{\mathbf{\Phi}}^i$ (purple region in Figure 3c), which in effect, puts more emphasis on neighboring points that are more probabilistically Pareto optimal than $\hat{\mathbf{\Phi}}^i$.

Once N_i has been determined, a fitness value that ranges between [0,1] and varies inversely with N_i can be assigned through

$$f_i = \left(1 - \frac{N_i - \min(N)}{\max(N) - \min(N) + 1}\right)^{\alpha} \tag{14}$$

where the exponent α controls the emphasis placed on lonelier positions. Figure 3c illustrates the resulting neighborhood of $\hat{\mathbf{\Phi}}^i$ within which positions are accounted for in f_i. By considering only positions in this neighborhood, which are likely to dominate $\hat{\mathbf{\Phi}}^i$, we preserve both optimality and diversity in fitness calculations while alleviating convergence issues associated with weakly optimal Pareto front estimates, for example, as can be seen in Figure 4b.

While this method is effective for highly uncertain emulated positions, it will lose its efficacy eventually as the algorithm converges and the variance of emulated positions becomes very small. Geometrically, if the standard deviation among emulated positions is much smaller than the distance between nearby positions, each position in the repository will result in $N_i = 0$ and therefore have an assigned fitness of 1.0, regardless of the crowding taking place.

Since Equation 13 is predicated on the overlap of confidence ellipsoids, this drawback can be overcome by imposing a minimum variance for each repository position, termed here as pseudo-variance. The value of this pseudo-variance, ν_k, should be large enough so that in the event that all positions in the repository are nearly deterministic, the resulting confidence ellipsoids of neighboring positions still have some overlap. A simple method for setting a value for ν_k is to divide the current range of objective k values by the repository size:

$$\nu_k = \frac{\max(\bar{\phi}_k) - \min(\bar{\phi}_k)}{n_{\text{rep}}} \tag{15}$$

Then, when computing the required probabilities of dominance for Equation 13, the variance in Equation 9 is replaced with ν_k when the estimated variance of emulation, $\hat{v}_k$, becomes less than ν_k. However, it is important to

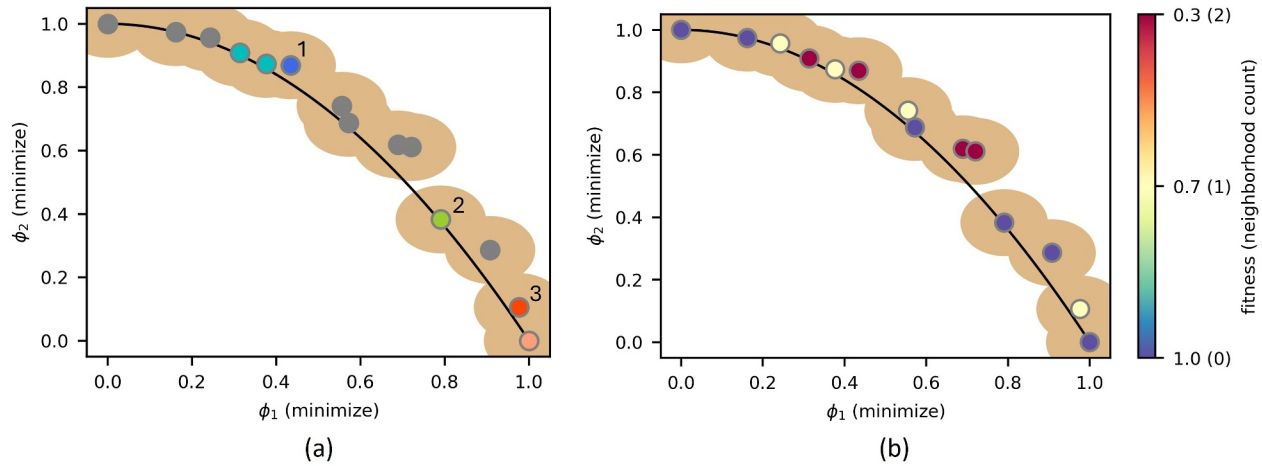

Figure 5. (a) By imposing a pseudo-variance, the fitness of positions are reasonable. Illustrated here are three of the positions represented by dark hues of blue ($\hat{\mathbf{\Phi}}^1$), green ($\hat{\mathbf{\Phi}}^2$), and orange ($\hat{\mathbf{\Phi}}^3$). Their neighbors per Equation 13 are represented by the corresponding light hues of the said positions. Among the scenarios shown, $\hat{\mathbf{\Phi}}^2$ is the loneliest and therefore assigned the highest fitness, followed by $\hat{\mathbf{\Phi}}^3$ and $\hat{\mathbf{\Phi}}^1$; (b) The fitness values of all position are shown.

emphasize that ν_k is only to be used for the purpose of fitness calculations (Equations 13 and 14), and PPD should always still use the $\hat{v}_k$ obtained through Equation 5.

Figure 5 illustrates a hypothetical example of how the use of pseudo-variance allowed Equation 13 to determine a reasonable fitness criteria even when all positions are deterministic. Figure 5a demonstrates the neighbors selected for each of three positions in the repository. As can be seen, the method picks up two neighbors near position 1, position 2 has no neighbors and 3 has a single neighbor. The resulting fitness values (Figure 5b) are reasonable representations of crowding and relative optimality. Although there is a gap to the right of $\hat{\mathbf{\Phi}}^1$, it is given a low fitness value because it is still relatively Pareto suboptimal compared to other positions. In typical deterministic crowding-based fitness evaluation, $\hat{\mathbf{\Phi}}^1$ would have been assigned a higher fitness value which in the case of SAMOO may result in divergence from the Pareto front.

3.5. Handling of Constraints

Many MOO problems are subject to constraints (Equation 1). Some of these constraints may depend on model outcomes, for example, groundwater drawdown, and therefore must be supported by the surrogate model, in addition to the objective functions. Therefore, the uncertainty in emulated constraints must be accounted for in the MHA.

Since, for some emulated position i the estimate for a model-derived constraint m is

$$\hat{\theta}_m^i \sim \mathcal{N}\left(\bar{\theta}_m^i, \hat{v}_{\theta_m}^i\right) \tag{16}$$

A probabilistic method may also be employed when evaluating the feasibility of positions. The probability of feasibility of a position for constraint m can be computed as

$$PF_m^i = P\left(\hat{\theta}_m^i \leq c_m\right) = \Psi\left(\frac{c_m - \bar{\theta}_m^i}{\sqrt{\hat{v}_{\theta_m}^i}}\right) \tag{17}$$

Then, an emulated position is deemed feasible if $PF_m \geq \lambda_m$ where $\lambda_m \leq 0.50$. The value of λ_m sets the forgiveness threshold for accepting a position whose emulated mean for a constraint violates the prescribed limit. A value of 0.50 is similar to treating the mean emulated constraint value as deterministic.

There are a number of methods for dealing with infeasible positions among the MHAs available. The method employed here is largely based on that described in White et al. (2022), with infeasible positions assigned an arbitrarily large objective function value such that it has no impact on the convergence of the MHA, that is, it will always be dominated with certainty. It is possible, and quite likely for most real-world problems, that the initial ensemble is completely infeasible. This can slow the convergence significantly, if not prevent it altogether. Our results show that surrogate modeling offers another advantage in this regard, as it provides a means to identify (potentially) feasible solutions with a relatively small number of PBM evaluations.

3.6. Overall Algorithm for Fast, Adaptive Surrogate-Assisted MOO

We present an algorithm that is the first among a handful SAMOO algorithms that: (a) Incorporates uncertainty in the non-dominance concept, (b) employs a multi-objective acquisition strategy to characterize intelligence among a large, informative, population to more effectively aid machine learning, (c) utilizes probability concepts to better handle uncertain constraints; and (d) uses an ensemble method that exploits the power of parallel computation on HPCs. The algorithm employs GPR, but it can be implemented with other surrogate modeling techniques as long as the predictive uncertainty of its emulations can be efficiently calculated. The user sets the number of inner iterations to perform and checks convergence by visual assessment of the progression of the Pareto cloud in the inner iterations. The algorithm allows for the user to restart the inner iterations from the last iteration if the Pareto cloud is still fluctuating. Similarly, convergence of the true repository is assessed visually by looking at the diversity and improvement of the current Pareto optimal solutions compared to the previous iteration. Quantitative convergence measures, such as hypervolume-based metrics (Appendix C), may also be employed to assess convergence. Figure 6 provides a logical flowchart of the algorithm.

Our algorithm can be easily implemented using PESTPP-MOU (White et al., 2022) and any number of open-source GPR implementations. PESTPP-MOU is an open-source decision support tool that provides a flexible way to carry out optimization algorithms in a non-intrusive way, that is, the premises of the problem need not be hard-coded into the tool and it can operate on any model, whether PBM or GPR. Moreover, PESTPP-MOU allows for parallel computing on an HPC system. Thus, the computational gains from using GPR in the inner iterations is magnified further and the expensive PBM runs of the infill ensemble in outer iteration is executed more efficiently using multiple compute cores. Modifications to the current version of PESTPP-MOU were made to incorporate PD calculations for PPD and fitness evaluations.

4. Numerical Experiments and Implementation

4.1. Benchmark Test Functions

The presented algorithm was applied to three test functions to demonstrate its performance in solving MOO problems and validate the results with the known solutions of these functions. The test functions chosen were by Zitzler et al. (2000), Kursawe (1991), and Osyczka and Kundu (1995), which will be simply referred to as ZDT, KUR, and OSY test cases, respectively. These test functions were specifically chosen to illustrate the performance of the presented algorithm in solving nonlinear, discontinuous, and constrained MOO problems. Moreover, these functions are standard problems widely used to evaluate the performance of MOO algorithms in various studies (Datta & Regis, 2016; Khosravi et al., 2018; Kim et al., 2004; Lim et al., 2015; Wang et al., 2022; Xiaohui & Eberhart, 2002). Table 1 shows the summary of the optimization problem structure for these functions while the problem statements can be found in Appendix A.

The benchmarks were first optimized using PESTPP-MOU which operates by executing a script that evaluates the test functions, mimicking an expensive PBM evaluation, as a gauge for assessing the relative performance of the proposed algorithm in terms of speed and efficiency. As MOPSO is used as the generator of new solutions in every iteration, this expensive optimization procedure will be referred to as the "process-based-MOPSO" or simply PB-MOPSO to distinguish it from the presented algorithm, which will be referred to as the "surrogate-assisted-MOPSO" or SA-MOPSO. The surrogate model was constructed using laGP regression developed by Gramacy (2016) through an associated R package. Both PB-MOPSO and SA-MOPSO were initialized with the same population.

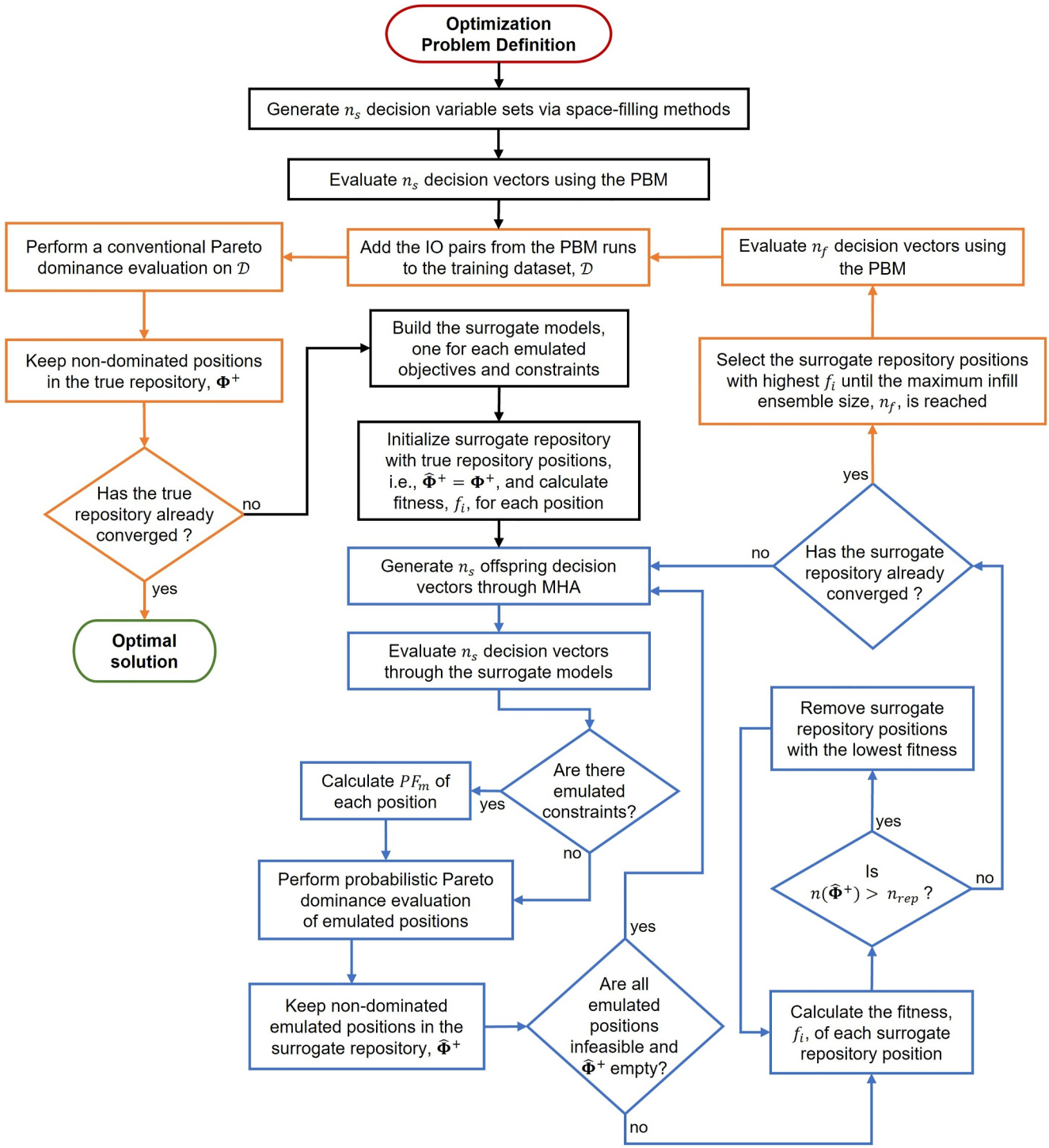

Figure 6. This flowchart shows the overall algorithm proposed in this study. Outer iterations are represented by orange arrows and include the processes that produce the true Pareto optimal solutions and involve the most expensive part of the algorithm—the Process-based model runs. The blue arrows represent the inner iterations which utilize the surrogate model to obtain an ensemble of most informative infills for the next outer iteration.

4.2. Application—Coastal Aquifer Management Problem

Extensive coastal developments may lead to excessive groundwater withdrawal which results in a steep decline of the water table resulting in seawater intrusion. Barrier wells that artificially recharge the coastal aquifer to control

Table 1
Benchmark Function Characteristics

Parameters	KUR	ZDT	OSY
n_o	2	2	2
n_d	3	30	6
n_c	0	0	6

the hydraulic gradient and prevent or deter the intrusion of seawater is commonly used as mitigation measure to ensure that production wells are able to withdraw potable water (salinity $\leq$ 1.0 g/L) and meet the demand. GDEs such as coastal wetlands also need to be protected from too much withdrawal; hence, while barrier wells may allow for more extraction, the pumping rate in the production wells needs to be managed to maintain the water table at safe level for GDEs. It is therefore prudent to develop a management strategy that sets the optimum extraction rate in each production well and the rate of injection in the barrier wells.

A density-dependent groundwater model is necessary to fully understand the coastal aquifer conditions and simulate the migration of seawater inland. The model used in this example application was developed using MODFLOW-6, which facilitates both the density-dependent groundwater flow model (Langevin et al., 2017) and solute transport model (Langevin et al., 2022). The model domain is patterned after the aquifer system in coastal Los Angeles (USA) known as the Dominguez Gap (Nishikawa et al., 2009). One of the interesting features of that aquifer system is the presence of faults which caused discontinuity in its lithological structure that is primarily composed of sand and clay. This discontinuity creates a possible pathway for seawater to migrate from shallow to deep aquifers. This feature is represented in the model in this study by a break and a gap in the clay layer of the domain.

The model domain is shown in Figure 7 and is a 2D vertical slice of the aquifer system with a depth of 90 m, a length of 8,000 m, and an arbitrary width of 1 m. To properly simulate and visualize the migration of saltwater across the domain, it is discretized to 360,000 (90 $\times$ 4,000) cells.

The inland boundary has ground surface elevation of 20 m above sea level (masl) which gradually drops toward the wetland (bottom cell elevation at 0 masl, 2,800 m from the inland boundary), then meets the coast on the other end of the domain at 0 masl. Model cells above the ground surface are kept in the domain nonetheless and a recharge of 0.82 mm/day is applied on the highest active cell to maintain model stability. The influx of groundwater from inland is represented as a specified flux boundary. Saltwater with a concentration of 35.0 g/L enters the domain through the general head boundary in the coastal end of the domain.

Eight production wells are placed between the inland boundary and the wetland—four (W1-W4) are located 1,250 m from inland and the other four (W5-W8) are placed 1,800 m from inland. Further, four of these are shallow wells tapping the unconfined sandy aquifer in layers 40 (W1 and W5) and 55 (W2 and W6), while the other four extracts water from the confined aquifer in layers 70 (W3 and W7) and 85 (W4 and W8).

Saltwater is expected to migrate inland due to abstraction at the production wells; thus, the wells are vulnerable to saltwater intrusion and water table depletion may impact the wetland. The wetland may also experience saltwater intrusion but this is not an issue in this management problem. To alleviate these issues, two injection bores are installed—one shallow (B1) bore, located near the wetland at layer 40, 2750 m from inland, and one deep bore (B2), located at layer 85, 2000 m from inland. The salinity of the injectant is assumed to be 0 mg/L; in reality, injectants may have small nonzero salinity, and while this does not affect the applicability of the methods proposed in this study, it will have some small effect on the orientation of the resulting Pareto front. More detailed characteristics and other model parameters are provided in Appendix B.

A pre-development period of 1,000 years (i.e., steady-state simulation) without any extraction or injection was first simulated to establish the water table and salinity across the domain under natural conditions (Figure 7b). Figure 7c shows a closer look at the saltwater wedge where the effect of the clay layer is evidently seen in two distinct toes formed above and below the clay layer. The concentration and head values are then used as the initial condition for the 100-year predictive period (i.e., transient simulations) where the extraction and injection wells are operational.

The relationship between inputs and outputs in this model is highly nonlinear and the computational burden arises from solving the solute transport equations governing saltwater intrusion. A single forward simulation takes at least 3.0 hr to finish on a typical desktop computer. Longer CPU run time may be required for some combinations of decision variable values as the transport solver may take numerous iterations to meet the required convergence criteria and sometimes fail to converge altogether. Optimizing the pumping distribution and operating solely on

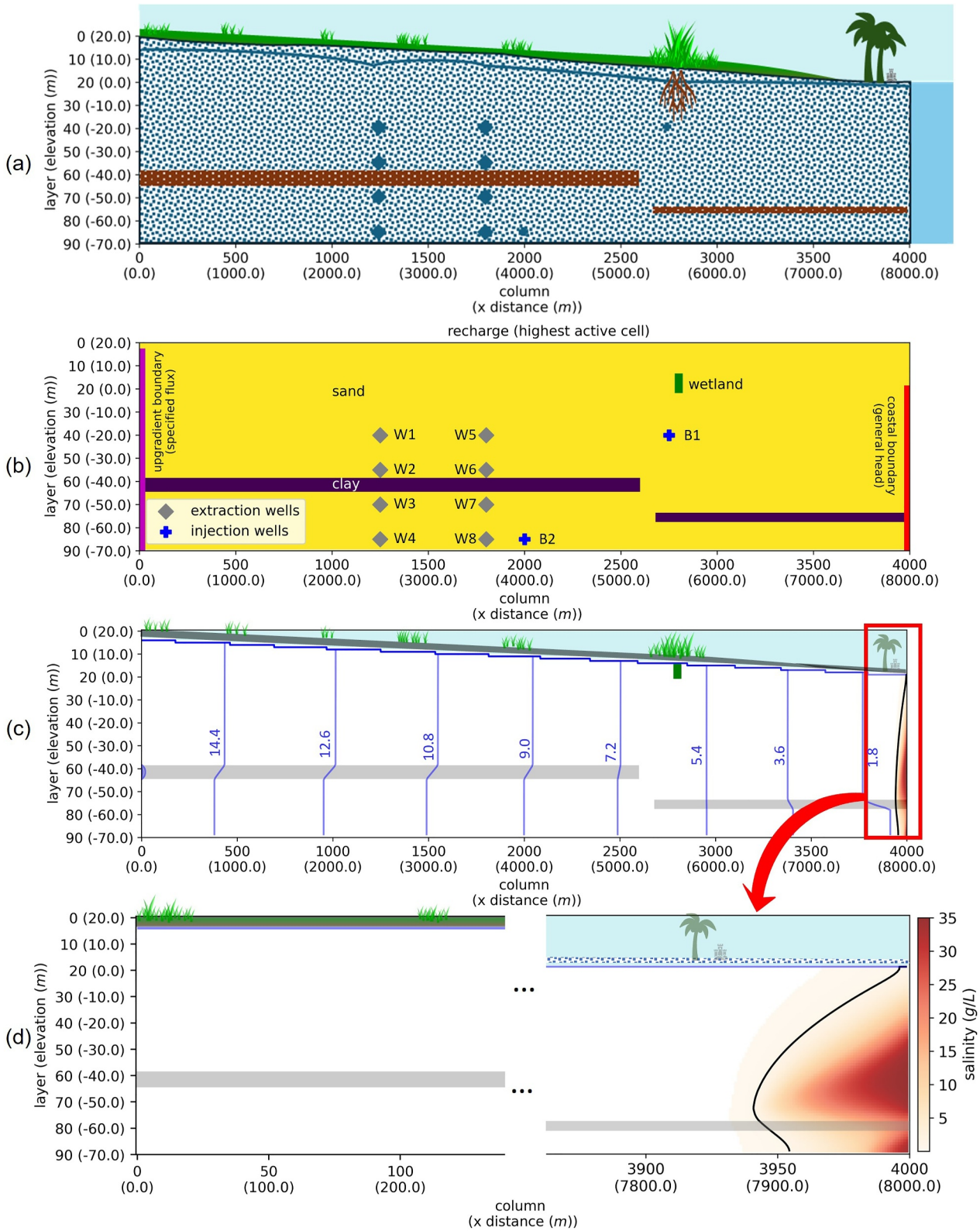

Figure 7. (a) An illustration of the 2D slice of the aquifer; (b) model domain and boundary conditions; (c) saltwater concentration in coastal groundwater after a pre-development period of 1,000 years; (d) the groundwater-seawater interface formed after the pre-development period. The blue contour lines represent heads and the black line represents the salinity threshold for potable water (1.0 g/L).

the PBM is, therefore, a slow and expensive undertaking. Surrogate-assisted optimization using the algorithm presented in this study provides the means necessary to expedite this process and make timely decisions.

4.2.1. Problem Formulation

The optimization problem can be formulated in many ways to address the competing objectives of maintaining acceptable water quality and supply volumes, while striving to protect the wetland and also minimizing the implied cost of operating an artificial barrier; there is a clear conflict between the rate of injection of the barrier and the quality of the of the extracted water. Installing and operating an artificial barrier could be a costly endeavor (Vanderzalm et al., 2022). Hence, economically, one would prefer to minimize, if not to avoid, injecting artificial recharge into the barrier. However, doing so could jeopardize the quality of produced water in terms of salinity by failing to mitigate landward migration of high salinity seawater, and, ultimately lead to a failure to meet the required potable water demand. It would therefore be interesting to know the trade-off between the barrier injection rate and the quality of produced water during a 100-year pumping period. Suppose the location of the barriers is fixed as shown in Figure 7, we can look into minimizing the total injection rate in the barriers while minimizing the flux-weighted mean salinity of the combined extracted water from the wells.

We seek to optimize the operation of each production and injection well—a total of 10 decision variables. Each extraction well has a capacity of up to 4.75 m^3/d and each injection well can operate up to 7 m^3/d. Additionally, the other issues mentioned above should also be considered in the problem as constraints: (a) Minimum groundwater demand of 28.5 m^3/d should be met, (b) total injection rate is limited to 8 m^3/d, (c) drawdown at the wetland compared to pre-development conditions cannot exceed 5.75 m, and (d) to maintain freshwater quality, the flux-weighted mean salinity cannot exceed 1.0 g/L at any point in the simulation. The MOO problem can be written as

$$\begin{aligned} \min_{\mathbf{x}} \boldsymbol{\Phi}(\mathbf{x}) :&= \{\phi_1(\mathbf{x}), \phi_2(\mathbf{x})\} \\ \phi_1(\mathbf{x}) &= \mathbf{1}^\top \mathbf{b} \\ \phi_2(\mathbf{x}) &= \max_t \left(\frac{\mathbf{s}_t^\top \mathbf{w}}{\mathbf{1}^\top \mathbf{w}}\right) \end{aligned} \tag{18}$$

subject to:

$$\begin{aligned} \mathbf{1}^\top \mathbf{w} &\geq 28.5 \text{ m}^3/\text{d} \\ \mathbf{1}^\top \mathbf{b} &\leq 8.0 \text{ m}^3/\text{d} \\ \max_t \left(\frac{\mathbf{s}_t^\top \mathbf{w}}{\mathbf{1}^\top \mathbf{w}}\right) &\leq 1.0 \text{ g/L} \\ d_{wl} &\leq 5.75 \text{ m} \\ 0 \leq \mathbf{w} &\leq 4.75 \text{ m}^3/\text{d} \\ 0 \leq \mathbf{b} &\leq 7.0 \text{ m}^3/\text{d} \end{aligned} \tag{19}$$

where $\mathbf{x} = [\mathbf{w}, \mathbf{b}]^\top$ is the vector of decision variables consisting of the well extraction rates, $\mathbf{w}$, and the barrier injection rates, $\mathbf{b}$; $\mathbf{s}_t$ represents the salinity in each extraction bore at time $t = 4, 8, 12, \ldots, 100$ years, and d_{wl} is the drawdown at the wetland.

5. Results and Discussion

5.1. Benchmark Test Functions

5.1.1. Convergence Performance

The comparison of the performances of SA-MOPSO and PB-MOPSO was made by looking at their respective outputs after having utilized the same total cumulative computational effort. However, the total wall time for running each algorithm is not comparable for these benchmark problems because the PBMs, being simply closed-

form functions that are easy to evaluate, run faster than the GPR. Thus, we compare the relative convergence in terms of how often the PBM is called, which is at (a) each iteration of PB-MOPSO, and (b) each outer iteration of SA-MOPSO. Repository positions after iteration-n of PB-MOPSO and after outer iteration-n of SA-MOPSO are overlain in the succeeding plots for comparison. The full results of PB-MOPSO runs are provided as supporting information (Figures S8–S10 in Supporting Information S1).

The first objective of ZDT is simply the value of one of the decision variables; hence, emulating this objective is unnecessary and only the second objective was emulated. We demonstrate that in such cases where one of the objectives is simple (and at least one other is model-derived), the presented SAMOO algorithm can still be applied. On the other hand, both objectives were emulated for KUR and OSY. Further for OSY, all the six constraints were emulated.

The plots in Figure 8 show the Pareto optimal solutions at the first outer iteration of SA-MOPSO and the final repository solutions for both SA-MOPSO and PB-MOPSO, while Table 2 summarizes the performances of SA-MOPSO and PB-MOPSO in terms of the number of non-dominated solutions obtained after having conducted the same number of PBM runs. Plots of converged SA-MOPSO results for all the benchmarks are provided as supporting information (Figures S9, S16, and S21 in Supporting Information S1). A comparison of hypervolume metrics for these results is also provided in supporting information Table S1in Supporting Information S1.

It is worth noting that SA-MOPSO outperformed PB-MOPSO in all the benchmarks in terms of the number of expensive function evaluations required to achieve convergence on the true front. Moreover, it is remarkable how after just one retraining of the surrogate models, SA-MOPSO already found a good number of solutions on the true front for ZDT and OSY. This is somewhat expected for ZDT as the problem is the simplest among the three benchmarks having only one emulated objective. Nevertheless, despite ZDT's seemingly simple formulation, PB-MOPSO still required more than 10 iterations to get some positions on the true front. As for KUR, while PB-MOPSO obtained one solution that is already in the front at the first iteration, it still took at least 20 iterations to get to the convergence that SA-MOPSO had in three outer iterations.

The added challenge of constraints in OSY makes it an interesting benchmark to illustrate how the algorithm performs when the true Pareto front is governed largely by constraints. Figure 9 provides a snapshot of the progression of the cloud near the beginning and end of the inner iterations along with the development of the front at each outer iteration after retraining the surrogate model. It is evident in the plots that PPD mattered most in the early stages of the optimization when the surrogate model has limited knowledge about the problem. By allowing some uncertainty in dominance evaluation, PPD identified the regions of interest after 150 inner iterations within which the surrogate model should gather more information. Then by filtering out redundant positions from this ensemble, the surrogate model was able to learn as much as it can as early as possible, which results in already accurate predictions after Outer-2, as if subsequent retraining is no longer necessary.

As the obtained positions already spanned most of the true front from Outer-2, the succeeding iterations were focused more on small improvements and diversification of the incumbent optimal solutions and less of exploration. It is important to note that PPD is still applied to all subsequent inner iterations, but because of the small uncertainties in the emulations, the cloud at this stage has been reduced to the usual front, which demonstrates that PPD is a generalization of the conventional Pareto dominance concept under uncertainty.

These results show that the computational requirements of SA-MOPSO is far cheaper than PB-MOPSO. This reduction may not be significant in terms of wall time when optimizing test functions that can be evaluated instantly, but, as will be demonstrated in a more computationally expensive problem later, such reduction means hours, even days, of CPU-time saved. Moreover, results suggest that SA-MOPSO is even more effective than PB-MOPSO in solving a highly constrained MOO problem. And as will be demonstrated in the groundwater modeling application, SA-MOPSO can also be more effective at finding feasible solutions to highly constrained problems as well.

5.1.2. Role of β in Sequential Learning

The value of β and the prevailing uncertainty of the emulated positions determine the appearance of the Pareto cloud in inner iterations. The fuzziness of the Pareto cloud is more pronounced during the early stages of the optimization when GPR is not yet trained enough. The cloud tells the algorithm the points of interest in the

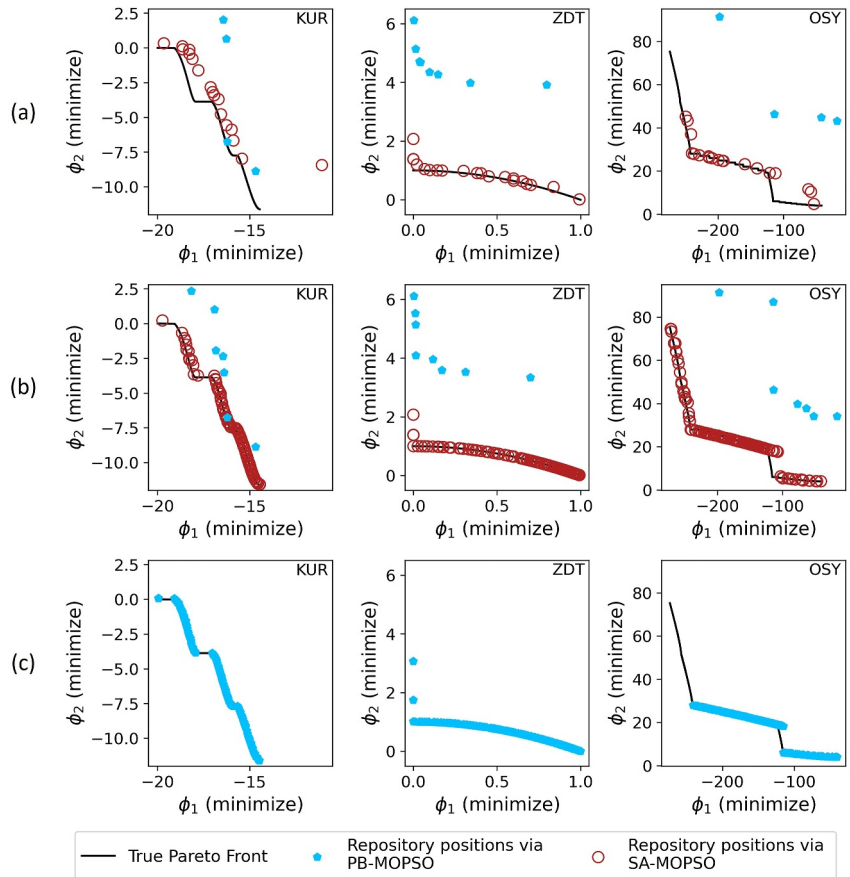

Figure 8. The obtained Pareto fronts for KUR, ZDT, and OSY problems using the proposed SAMOO algorithm (a) after the first retraining (outer iteration 1), and (b) the final converged front after outer iteration 4, 3, and 5, respectively. Overlain are the solutions obtained using PB-MOPSO at each problem's corresponding iteration, while (c) shows the converged front obtained through PB-MOPSO after 100, 50, and 300 iterations for each respective problems. Evidently, SA-MOPSO solutions have better coverage of the true Pareto front and this was achieved with more than 10x fewer Process-based model evaluations compared to PB-MOPSO.

objective space where the GPR should be retrained next; thus, the choice for the value of β also determines how much information the GPR learns at each outer iteration.

To demonstrate how the value of β affects the Pareto cloud, we performed five cases of SA-MOPSO on KUR with different values of β: 0.40, 0.50, 0.60, 0.70, and 0.80, each having 50 inner iterations for four outer iterations. Plots showing the inner and outer iterations of the algorithm for each of these cases are provided as supporting information (Figure S11–S15 in Supporting Information S1). The plots of the resulting Pareto cloud after inner iteration 50, prior to its appendment to the design and retraining the GPR, are shown in Figures 10a–10e. For this benchmark, it appears that a value of 0.50 seems to perform best which is evident from the accuracy in the results of Outer-3 as well as the clear reduction in variance observed in the inner iterations of Outer-1 and Outer-2.

We also demonstrate how the value of β affects GPR's learning of $\mathcal{P}$. Since $\mathcal{P}$ is already known from the results of PB-MOPSO, we used the incumbent GPR (i.e., the GPR built using $\mathcal{D}$ after outer iterations 0, 1, 2, and 3) to predict $\mathcal{P}$, using the corresponding decision variables as inputs, and observe the errors in these predictions. Theoretically and ideally, these errors should lessen as $\mathcal{D}$ grows, which directly relates to the learning progress of GPR in accurately predicting $\mathcal{P}$. The learning curve shown in Figure 10f illustrates the error of emulating $\mathcal{P}$, after each outer iteration, measured through the sum of the Euclidean distances of each Pareto optimal position's predicted mean to their respective true objective values:

$$\ell_t = \frac{\sum_{i=1}^{n_p} \sqrt{\left(\hat{\phi}^i_{1|\mathcal{D}_t} - \phi^i_1\right)^2 + \left(\hat{\phi}^i_{2|\mathcal{D}_t} - \phi^i_2\right)^2}}{\sum_{i=1}^{n_p} \sqrt{\left(\hat{\phi}^i_{1|\mathcal{D}_0} - \phi^i_1\right)^2 + \left(\hat{\phi}^i_{2|\mathcal{D}_0} - \phi^i_2\right)^2}} \tag{20}$$

Table 2
Performance Comparison of SA-MOPSO and PB-MOPSO in Terms of the Number of Pareto Optimal Solutions Obtained

Function	Algorithm	No. of PBM runs	No. of GPR runs	No. of non-dominated solutions
KUR	SA-MOPSO	400	15,000	101
	PB-MOPSO	400	–	7[a]
ZDT	SA-MOPSO	300	10,000	108
	PB-MOPSO	300	–	8[a]
OSY	SA-MOPSO	500	60,000	101
	PB-MOPSO	500	–	4[a]

[a]Solutions are not Pareto optimal, but the best thus far.

where ℓ_j is a normalized measure of emulation error w.r.t. to the true front at outer iteration t, $n_{\mathcal{P}}$ is the number of solutions in $\mathcal{P}$, $\hat{\phi}^i_{1|D_t}$ is the predicted mean of objective 1 of position i given the $\mathcal{D}$ used to train the GPR at outer iteration t, $\hat{\phi}^i_{1|D_0}$ is the predicted mean of objective 1 of position i given the initial $\mathcal{D}$, ϕ^i_1 is the true objective 1 value of position i, and similar notations apply for objective 2. An ℓ_j value closer to zero indicates that much has been learned about $\mathcal{P}$ while a value near or over 1.0 indicates that nothing has been learned as errors persisted.

Probabilistic Pareto dominance with $\beta = 0.40$ is too lenient for dominance resulting in positions getting discarded too easily. It is similar to expanding the dominance region of each position, exploiting what the GPR knows, and trusting that they are correct. The Pareto cloud formed is more locally clustered at the regions close to the optimal solutions found in the previous outer iteration as seen in Inner-50($\mathcal{D}_0$) of Figure 10a. As a result, the infill ensemble would likely contain a lot of redundant positions. These positions have little added value on learning which is evident by consistently large values for ℓ, suggesting that the predictions generally become worse and any positions that happen to be on the true front are merely by chance. Even though the fitness function is meant to avoid such redundancies, it can only effectively do so if the Pareto cloud is wide enough. Hence, it is generally not a good idea to start the optimization being too exploitative given the very limited knowledge that the GPR has initially.

Increasing the value of β allows more exploration, which results in wider and noisier Pareto cloud. As seen among the plots in Figures 10a–10e, the Pareto cloud when $\beta = 0.80$ is the noisiest because it sets a relatively strict requirement for dominance; hence, only few positions get dominated while the rest have wide enough overlapping ellipsoids to not dominate each other and remain in the cloud. However, higher β does not necessarily benefit the optimization overall as learning efficiency starts to deteriorate and some disinformation may happen as seen in slight increase of ℓ_3 when $\beta = 0.80$. This disinformation arises from overshooting the sampling too far from the vicinity of emulated $\mathcal{P}$ (Inner-50($\mathcal{D}_2$) of Figure 10e). This becomes overly exploratory, placing too much confidence in the surrogate model, resulting in inefficient learning.

As mentioned previously, setting $\beta = 0.50$ for solving KUR is the most efficient in terms of learning which is consistent with the values of ℓ. On the other hand, $\beta = 0.60$ had a slow start for GPR's learning progress as a result of PPD slightly misinforming the algorithm of good infill candidates at the early stages of the optimization. However, such drawback is still rectifiable by performing additional outer iterations to make up for the missed opportunities for learning. As shown in Supporting Information Figure S10 in Supporting Information S1, the test case with $\beta = 0.60$ finally converged in Outer-4.

Results of using different values of β for ZDT and OSY are also provided in supporting information (Figures S16–S25 in Supporting Information S1) While some values of β are more efficient for GPR's learning than others, the goal is not to find the single most effective value for any particular MOO problem. Generally, β values between 0.50 and 0.70 work well in most cases and any misinformation or inefficiency of the PPD can be easily corrected with 1–2 additional outer iterations. Since the entire SAMOO process is much faster than a PBM-based MOO, the additional cost of performing additional outer iterations is still manageable. Other factors such as the initial design size, infill size, swarm size, and parameters of MHA and laGP could also potentially affect GPR's learning efficiency and different combinations of these values may require different β values to work more effectively. Nevertheless, we kept these parameters consistent for all the benchmarks

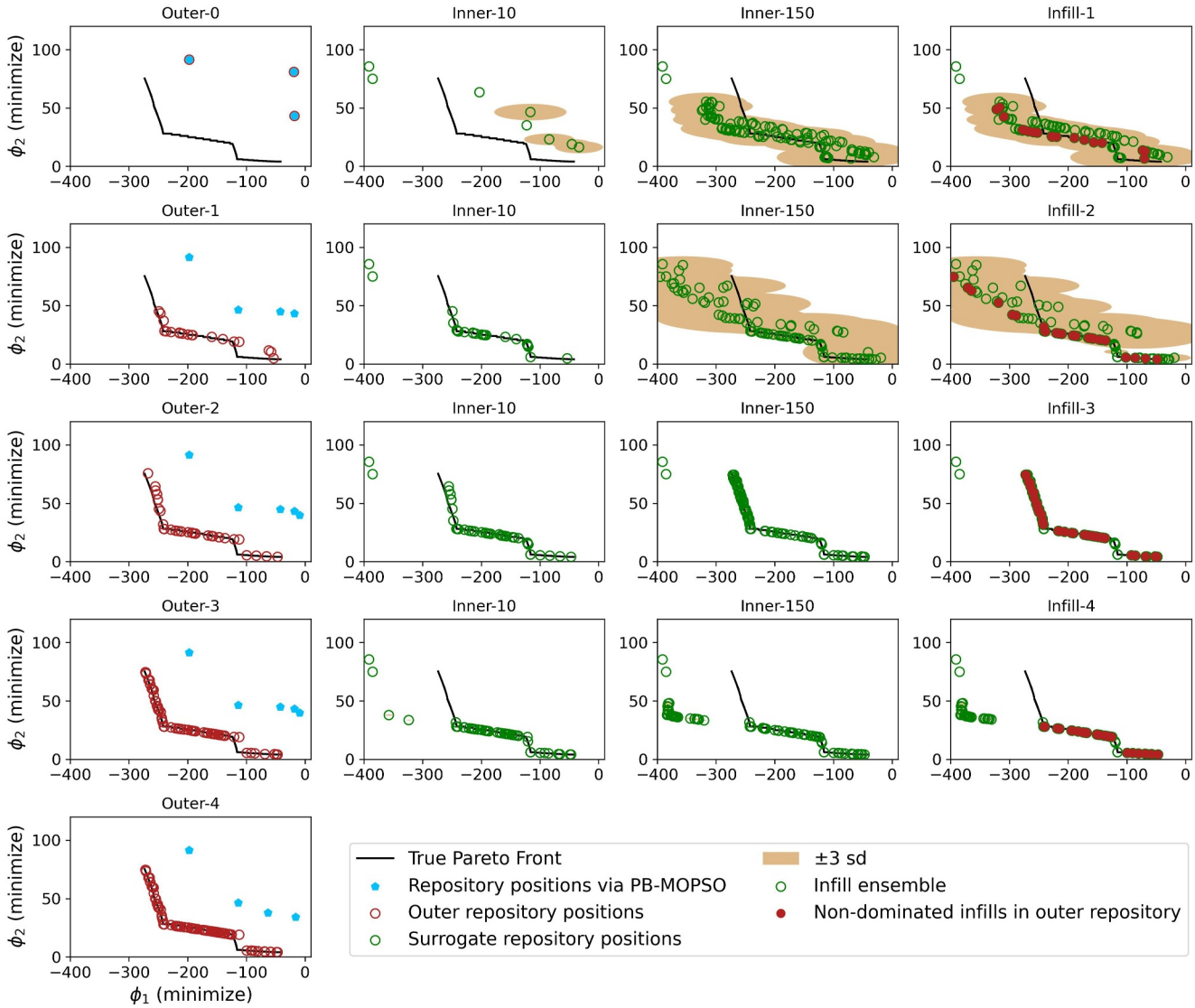

Figure 9. The proposed GPR-assisted algorithm for OSY quickly reached the true front with fewer Process-based model (PBM) evaluations than what multi-objective particle swarm optimization could have done when operating solely on the PBM.

tested in this study. The effect of the different combinations of values for these parameter on the performance of the algorithm, however, is outside the scope of this research.

5.1.3. Dealing With Uncertain Constraints

Several SA-MOPSO runs were performed on the highly-constrained OSY problem using different values of λ_m (Equation 17): 0.10, 0.25, 0.40, and 0.50. Results of each of these runs are provided as Supporting Information (Figures S24, S26–S28 in Supporting Information S1) while the resulting fronts at Outer-4 are provided in Figure 11.

A value of $\lambda_m = 0.50$ is similar to simply comparing the emulated mean value of a model-derived constraint against the respective constraint limit of the problem. As seen in Figure 11d, $\mathcal{P}$ after four outer iterations did not fully span the true front, even when continued for a few more iterations (see supporting information Figure S28 in Supporting Information S1). When the value of λ is lowered, then positions in the upper segment of the true front begun showing up in the cloud as also demonstrated in Figure 9. As a result, this relaxation of the constraint allowed for an opportunity for exploration, spanning this segment of the true front.

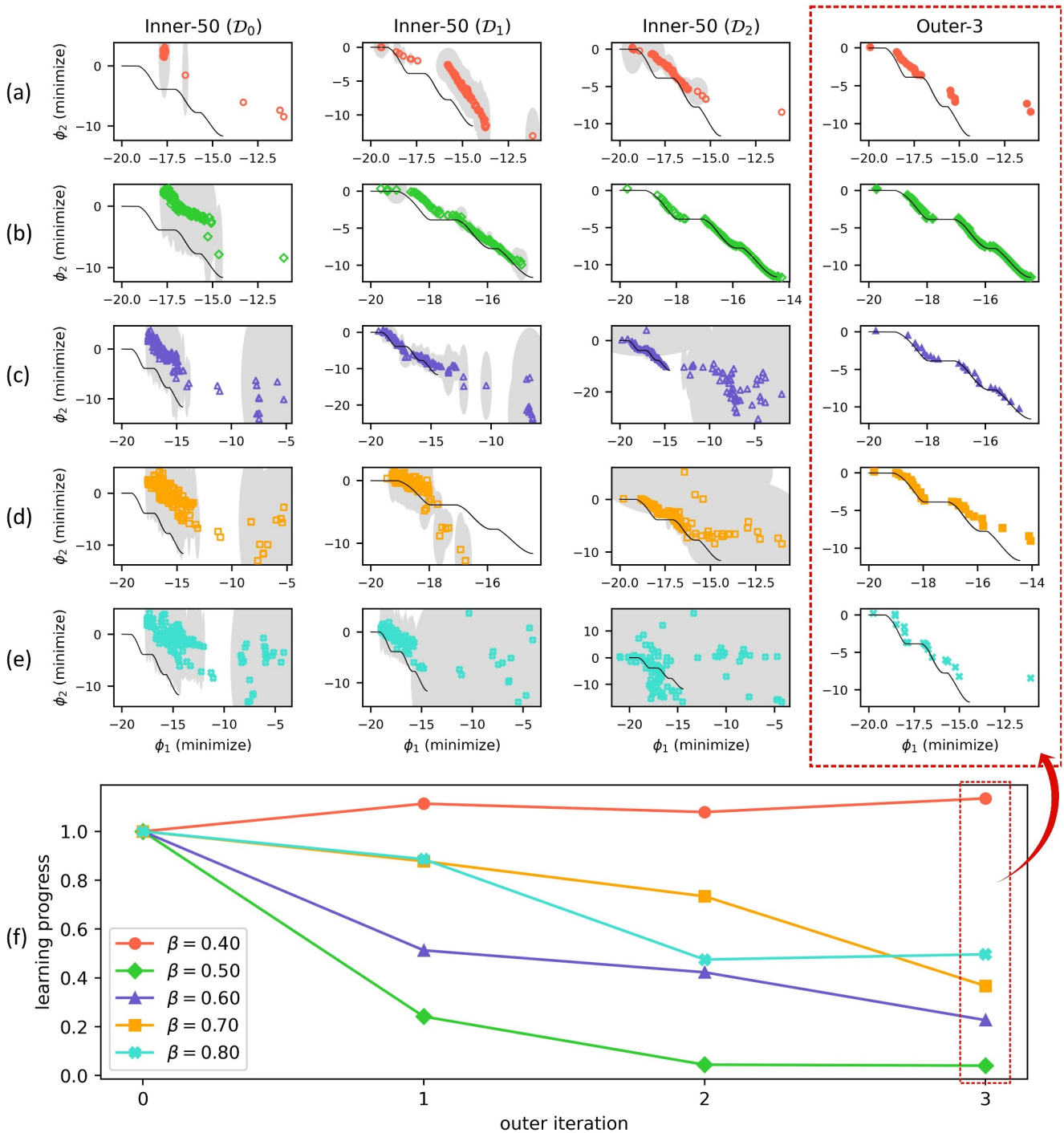

Figure 10. (a–e) Different β values produce different Pareto clouds, which affect the learning efficiency of the Gaussian Process Regression (GPR) upon retraining. The Pareto cloud for every 50th inner iteration and the front at outer iteration 3 are shown for each of β values tested. The gray ellipsoids indicate the uncertainty of emulating each point. The infill ensemble is selected from the cloud and used to retrain the GPR at outer iteration; (f) The learning curves (Equation 20) describe how accurate the retrained GPR can emulate the true $\mathcal{P}$ after retraining.

Among the values tested, $\lambda_m = 0.25$ is most effective in completely converging to the entire true front in this benchmark. Obtaining a robustly optimal value for λ_m is outside the scope of this study. For $\lambda_m = 0.10$ and 0.40 that were tested in this study, a few more outer iterations can further improve the part of the front that is constraint-

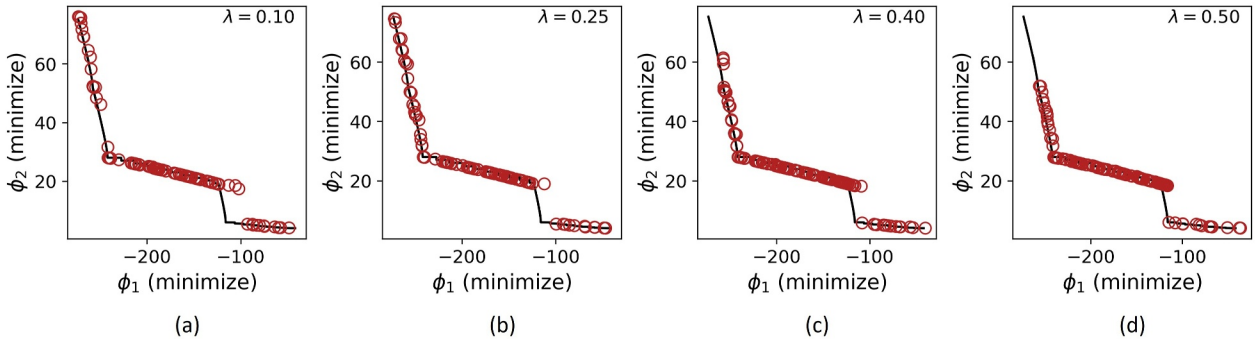

Figure 11. Comparison of the final fronts after four outer iterations produced with different values of λ used.

driven. This could generally be true for any $\lambda_m < 0.50$; however, further studies should be conducted to appropriately determine the optimal λ_m to use, but this may be problem-specific.

5.2. Application—Coastal Aquifer Management Problem

5.2.1. Initial Design and Overcoming Infeasibility

An initial design consisting of 100 decision vectors was first generated by Latin Hypercube sampling (LHS). Upon initial PBM evaluation of these vectors, none of them satisfied all the constraints of the problem. In situations like this, MOPSO may require some iterations to overcome infeasibility. One way to help MOPSO is to initialize it with a design that contains at least one feasible solution. This solution does not necessarily need to be optimum and can be obtained from the manager's experience or expert knowledge about the problem. If expert knowledge is not available, or does not yield a single feasible solution, one may have to resort to trial-and-error methods of finding a feasible solution, which is counterproductive and computationally expensive. An alternative is to increase the initial design size, although this entails additional computational cost. Another workaround is to have initial designs that satisfy the simple constraints (for this problem, these are constraints on $\mathbf{1}^\top\mathbf{w}$, $\mathbf{1}^\top\mathbf{b}$, $\mathbf{w}$ and $\mathbf{b}$) since they are easy to evaluate and do not require expensive PBM runs.

Nonetheless, to first observe how the algorithm performs in a worst-case scenario where designs are initially infeasible, both PB-MOPSO and SA-MOPSO were initialized with 100 all-infeasible LHS design sites (Outer-0). Only results of SA-MOPSO are shown in Figure 12 as PB-MOPSO was not able to produce a single feasible solution in 4 iterations. On the other hand, SA-MOPSO was able overcome infeasibility by the first outer iteration after 300 inner iterations were performed before retraining of the GPR. A β value of 0.7 was used for PPD evaluation in inner iterations. Model-derived constraints were also replaced by probability of feasibility (Equation 17) Different values of λ_m were used: 0.10, 0.25, and 0.40, of which 0.25 performed the best. The following discussions show results that used $\lambda_m = 0.25$, while plots for other λ_m values are provided as supporting information (Figures S29 and S31 in Supporting Information S1).

It is noticeable that the surrogate repository positions in the first set of inner iterations (first row of plots in Figure 12) are concentrated around the salinity of 0 g/L. This is because the initial training data set is also concentrated around 0 g/L salinity for ϕ_2, which indicates that the initial design is unable to capture the sensitivity of salinity to pumping (Figure 13a). This is a consequence of sampling the pumping rates between its bounds (0–4.75 m^3/d), which results in infeasible total pumping rate values that are mostly below the minimum demand and are therefore not enough to draw saltwater into the wells. As pointed out earlier, and later demonstrated in another experimental run, this drawback can be addressed by employing alternative sampling techniques. Even so, SA-MOPSO exploited this limited information and with the guidance of PPD, it was able to explore and identify where to focus next in learning and obtained two feasible and non-dominated solutions in Outer-1. Figure 13b shows how effective the first resampling was, that in the second set of inner iterations, GPR emulations of salinity is more varied and the cloud started taking the shape of the front as seen in the second row of plots in Figure 12.

Another important observation in Figure 12 is the presence of non-dominated solutions with negative emulated salinity values. The surrogate model at this stage has not yet learned enough resulting in such inaccurate and

10.1029/2024WR038554

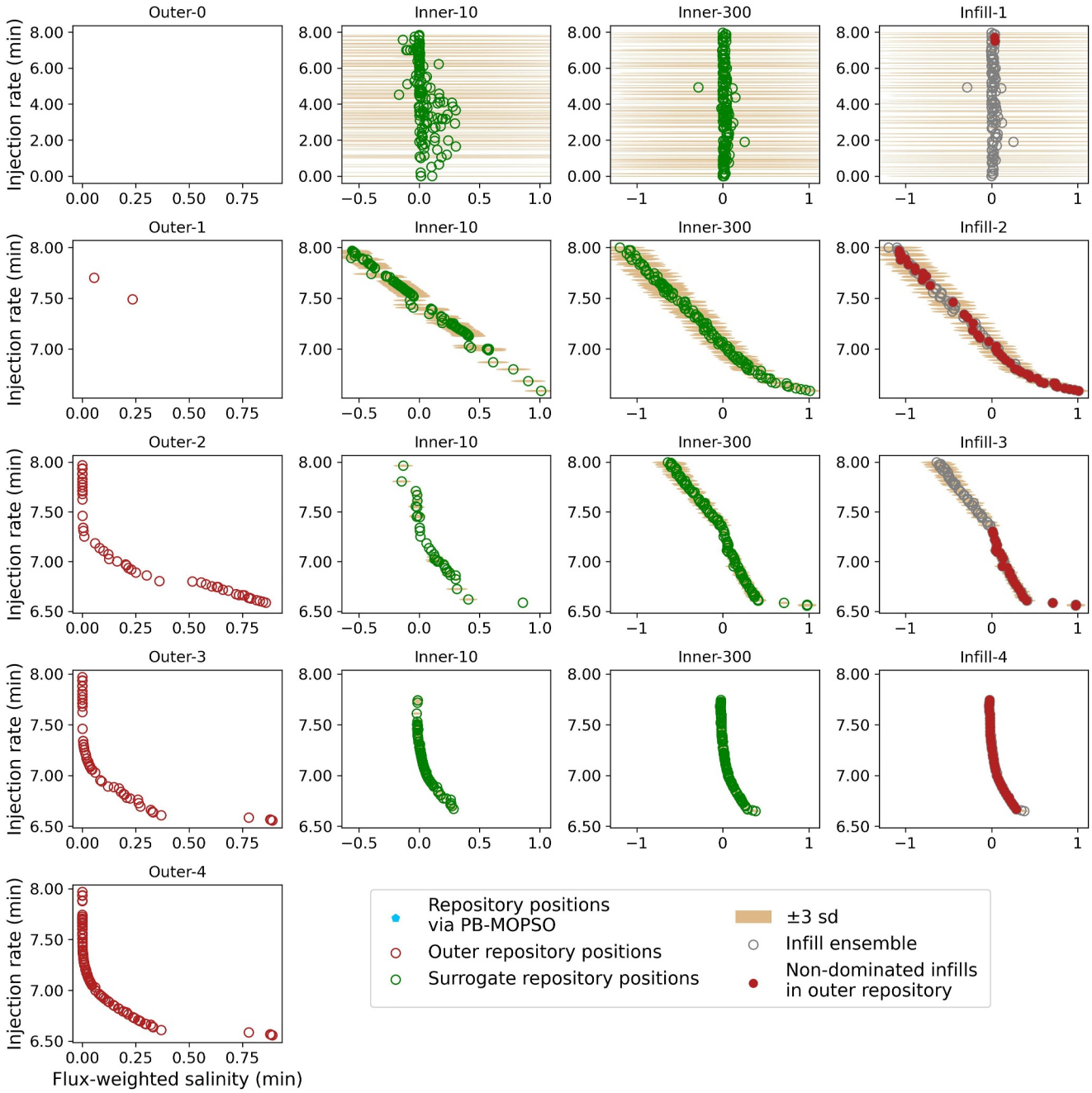

Figure 12. Results of SA-MOPSO-A for the seawater problem.

unrealistic predictions. As demonstrated in the benchmarks earlier, learning requires exploration, which means the surrogate model must be allowed to take on unrealistic values, that is, we must relax the surrogate model's adherence to physical reality in order to allow it to explore. This relaxation is in line with the idea of handling constraints through probability of feasibility and setting λ_m to be less than 0.50 to allow some violations of the constraints, because we recognize that surrogate model predictions are imperfect. It is worth noting from Infill-2 that many of the chosen infill, which have negative salinity values, turned out to have non-negative values when evaluated in the PBM and are actually non-dominated (Outer-2). Had there been no relaxation allowed, these infills would be deemed infeasible and discarded. But because the surrogate model was allowed to explore the said region of the objective space (as also seen in Figure 13), it effectively learns from its mistakes producing salinity approximations that eventually become less negative, and after Outer-3 (Figure 12), there are no longer

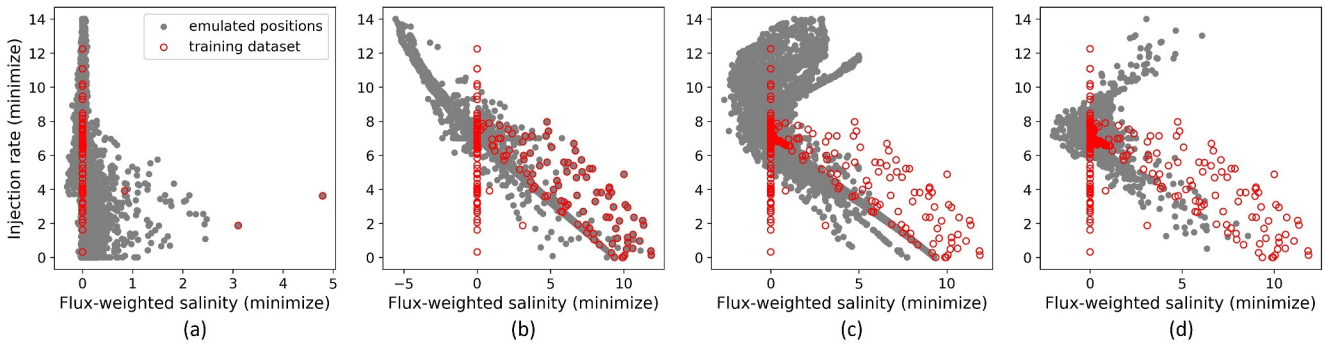

Figure 13. These plots show all the emulated positions vis-a-vis the training data set for each set of 300 inner iterations that follows: Outer-0 (a), Outer-1 (b), Outer-2 (c), and Outer-3 (d).

any non-dominated solutions in the surrogate repository with negative salinity values. Imposing a constraint on the minimum salinity such that it does not go below 0.0 g/L and use PF $\geq$ 0.25 to handle this in SA-MOPSO may also help to reduce the instances of extremely negative salinity values.

In just five outer iterations, SA-MOPSO performed very well, converging to the front, while PB-MOPSO still struggled to find a single feasible solution. When continued up to iteration 42, which took 6 days to complete (in parallel using HPC system with 100 cores), PB-MOPSO still failed to find any feasible solution. Table 3 shows a summary of the computational cost of running each algorithm. Note that the run time refers to a single instance of running a GPR or PBM and wall time refers to the elapsed time when PEST++ started a run manager to distribute runs to 100 cores in the HPC until it completed all runs it is tasked to do. The latter also includes (a) variable run-times between different decision variables, (b) some overhead time to send and receive decision variables over the network and (c) reattempts at failed runs. Developing low level workflows or code optimization to eliminate the time needed for I/O data transfers can potentially offer significant reductions in associated wall time, but these efficient methods are still a topic of further research.

Due to the fact the PB-MOPSO did not obtain any feasible solution, the actual wall time in these runs are not comparable. But just to illustrate the computational savings for using SA-MOPSO, suppose PB-MOPSO eventually found some feasible solution after iteration 42 and eventually converged after 100 iterations, the total wall time for PBM-MOPSO would have been 13.94 days. SA-MOPSO, on the other hand, took 1.10 days—a 13x reduction in computational cost. Note that this is a conservative estimate assuming PB-MOPSO can converge without an initial feasible solution; thus, the reduction factor could actually be several times larger. SA-MOPSO's impressive performance suggests that the GPR learning process not only speeds up convergence, but also aids in finding feasible solutions at negligible computational cost.

5.2.2. Validation and Overall Performance

To validate the Pareto front obtained, additional PB-MOPSO and SA-MOPSO runs with a starting population that are all compliant with the simple constraints were performed. For clarity in comparison, the previously presented results of runs in Section 5.2.2 will be referred to as Run A (e.g [PB/SA]-MOPSO-A) while the validation run will be referred to as Run B. For SA-MOPSO-B and PB-MOPSO-B, LHS was conducted to generate 25,200 decision

Table 3
Run Time and Wall Time Comparison of SA-MOPSO and PB-MOPSO for the Application Problem

	Average forward run time (mins)		Average parallelized wall time (mins)[a]	
Algorithm	GPR	PBM	Inner iteration	Outer iteration
SA-MOPSO	0.033	199.67	0.12	286.83
PB-MOPSO	–	131.28	–	200.79[b]

[a]Using 100 cores in an HPC. [b]Regular expensive iteration for PB-MOPSO.

Table 4
Performance Comparison Between SA-MOPSO and PB-MOPSO for the Application Problem

Run[a]	Algorithm	No. of PBM runs	No. of GPR runs	No. of non-dominated solutions
A	SA-MOPSO	500	120,000	125
	PB-MOPSO	500	–	0[b]
B	SA-MOPSO	400	90,000	61
	PB-MOPSO	400	–	5[b]

[a]Plots of these runs are provided as supporting information (Figures S30 and S32 in Supporting Information S1). [b]Solutions are not Pareto optimal, but the best thus far.

vectors, of which 100 satisfied the simple constraints (these constraints can be evaluated without running the expensive PBM) and are then used as starting population for both runs.

Table 4 summarizes the performance of Runs A and B. Performing a full PB-MOPSO until convergence could take weeks of wall time, even in an HPC cluster. Hence, it is difficult to compare SA-MOPSO's final converged front against that of PB-MOPSO's. Even so, PB-MOPSO-B runs were continued up to iteration 25, and the resulting front is also shown in Figure 14 where it can be seen that PB-MOPSO-B was not able to fully converge.

Although SA-MOPSO-A took one more outer iteration than SA-MOPSO-B, Figure 14 confirms that both runs obtained similar fronts particularly where trade-off between objectives is more significant. The additional cost is the consequence of initializing SA-MOPSO-A with all infeasible design. There are some weakly Pareto solutions still present in the front of SA-MOPSO-A, where injection rate is below 6.6 m^3/d, but these can likely be resolved by another outer iteration. Nonetheless, these solutions are less significant in decision-making. These results show that initial infeasibility does not significantly affect the overall performance and cost of SA-MOPSO.

Due to the inability of PB-MOPSO-A to find a feasible solution, and the computational expense required for PB-MOPSO-B to converge, another validation run performed to further confirm that the final fronts of SA-MOPSO-A and SA-MOPSO-B have fully converged. This was done by taking the Pareto optimal solutions at Outer-4 of SA-MOPSO-A as a starting population for 50 iterations of PB-MOPSO. Results of this validation run are provided as Supporting Information (Figures S34 and S35 in Supporting Information S1) and verify that the front obtained by the SA-MOPSO runs is likely the true Pareto front for this problem. Overall, SA-MOPSO runs A and B obtained solutions where actual trade-off is present (between injection rate of 6.6 and 7.2 m^3/d) which are more critical in decision-making. This conclusively suggests that the solutions have converged and are indeed Pareto optimal. Moreover, these runs demonstrate the additional benefit of using surrogate modeling to overcome infeasibility, which in traditional optimization could require numerous expensive iterations. This means that the computational savings of SAMOO over PBMOO is actually greater than simply speeding up convergence.

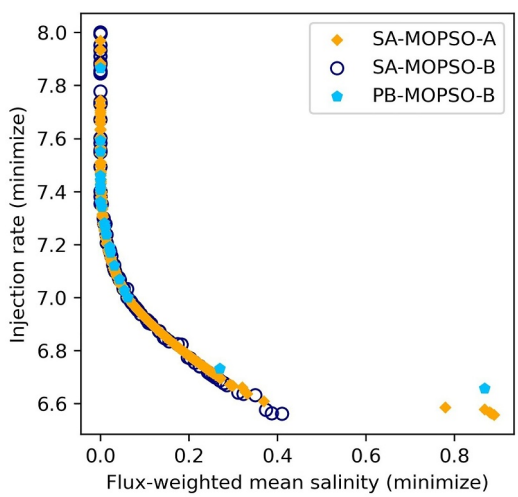

Figure 14. Comparison of the final Pareto fronts comprised of PBM-evaluated solutions at the last outer iteration in various SA-MOPSO runs vis-a-vis the solutions obtained by PB-MOPSO at the same computational cost. PB-MOPSO-A did not produce feasible solution and is not shown here.

5.2.3. Pareto Optimal Designs

It is apparent in Figure 14 that there are two inflection points in the Pareto front which divides it into three distinct segments. Interestingly, these segments (S1–S3) become more obvious when the percentage contribution of injection at B2 and drawdown are overlain on the front as shown in Figures 15a and 15b. To illustrate the effects of decisions in these segments, plots of the resulting concentration gradient and heads in the aquifer for a Pareto optimal solution taken from each segment are also shown in Figures 15c–15e.

Among these segments, S2 has the highest consequence in decision-making as this is where the biggest trade-off between the objectives is happening. In these designs, barrier B2 does almost all of the work in pushing back the toe of the saltwater wedge with which B1 would not have helped given its shallow location. Moreover, despite its deep location below the confining

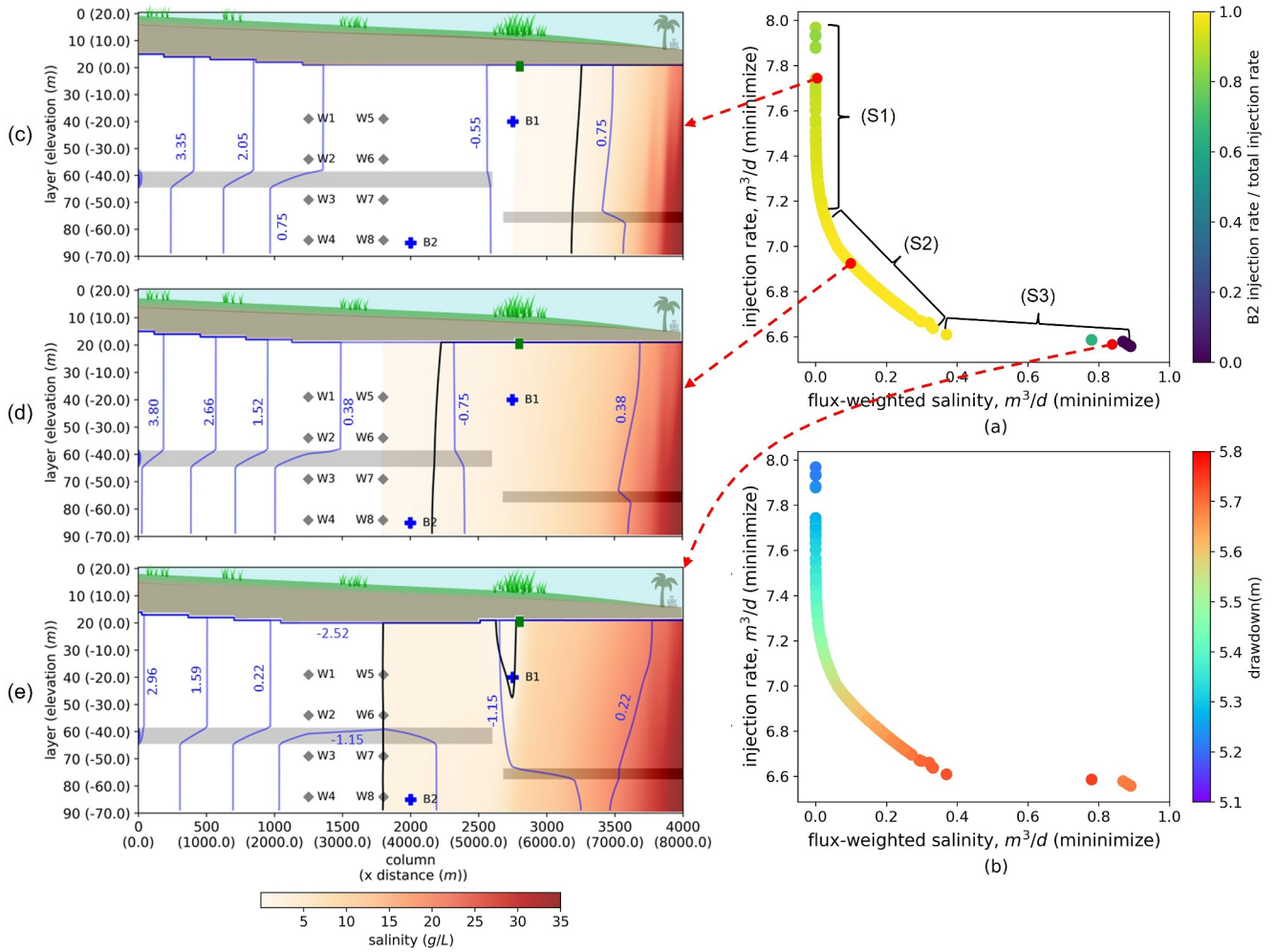

Figure 15. The contribution of B2 to the total injection rate (a) and drawdown at the wetland (b) are overlain on the fronts to provide insights on the final designs. (c–e) illustrates the resulting situation in the aquifer at the end of predictive period for some Pareto optimal solutions obtained from each distinct segment of the Pareto front. The blue lines represent head contours and the black line is the salinity threshold of 1.0 g/L.

layer, B2 still influences the head in the wetland through the gap in the clay layer.

It can be seen in Figure 15d that B2 is doing a good job of keeping the 1.0 g/L salinity contour line at some distance away from the wells. As the amount of injection from B2 decreases, that is, as one moves along S2 in the Pareto front (total injection also decreases as most are coming from B2), the salinity threshold contour line moves inland, suggesting that saltier water is coming out of the wells, and the drawdown in the wetland also increases up to the limiting value of 5.75 m, where another inflection in the front is observed.

Designs from S3 produce saltier water than that of S2, but still within the acceptable level. While Figure 15e suggests that none of the wells actually produce highly saline water, the drawdown constraint (notice the drawdown values in Figure 15b) limits pumping so that B1 can still manage to keep the wetland from drying up. Drawdown is an active constraint that influences S3 designs in the Pareto front—B1 should augment the reduced injection at B2 to protect the wetland at this point.

On the other end of S2, the inflection occurs at around total injection of 7.2 m^3/d. From this point on (S1), further increasing the injection rate has little to no effect on the total salinity of pumped water. This is also apparent in Figure 15c where the contour of salinity threshold limit is kept near the coast.

A closer inspection of the Pareto optimal designs reveals one prevalent trend anywhere along the front, that is, to operate six of the wells at their full capacity (4.75 m^3/d) and to turn off the other two, and that most of the artificial

recharge must be injected at B2. The four deep wells (W3, W4, W7, and W8) tend to be preferred because of their close proximity to B2, which provides direct protection from seawater intrusion. This also explains the need for most of injected water to be coming from B2. The shallow wells are less impacted by saltwater intrusion because they directly draw water from the water table; however, they directly affect the wetland. As such, the design prefers fully operating only two of the shallow wells and to compensate for their impact to the wetland, some injection may be needed at B1.

6. Summary and Conclusions

Machine learning techniques such as utilizing a surrogate model to mimic an expensive PBM has been an emerging technique to reduce the computational cost of simulation particularly for optimization problems that involves numerous of these PBM runs. However, one major challenge in using surrogate models is dealing with its prediction uncertainties. An adaptive resampling procedure has been an effective strategy in single-objective optimization to acquire new training points, that is, infill points, for the surrogate model and sequentially improve its accuracy. However, such adaptive technique requires special considerations in MOO problems to incorporate both optimality and diversity of chosen infill points. In this study, we utilized the PD concept in developing the SAMOO algorithm to incorporate uncertainty in dominance evaluation of surrogate emulations, which we termed as PPD. Then, we extended the use of PD concepts to an acquisition strategy within the SAMOO algorithm to integrate both crowding and probable optimality in a fitness function for selecting an ensemble of most informative infill points to be used for retraining the surrogate model. We also used the probability of feasibility concepts to allow some degree of forgiveness for an uncertain emulated position that violates some constraints.

We utilized GPR as the surrogate modeling technique because it provides a closed-form expression for estimating the variance of emulations, which can be used directly for probability calculations required for PPD, fitness function, and probability of feasibility. As for the MHA, we utilized PSO because of its efficiency and its memory-like nature of keeping track the best positions of the swarm complements with the selection of infill ensemble. It is important to reiterate, however, that PPD can be adopted in any MHA and surrogate modeling methods where the associated uncertainty in its predictions can be calculated. It is important to note though that the physical parameters of the model itself are assumed to be known and fixed during optimization; thus, in this study predictive uncertainty originates only from errors in emulations, not parameter noise. However, the concept presented here can be extended to include parameter error and will be the subject of future research.

We demonstrated that the presented SAMOO algorithm can be easily developed into existing software platforms (PESTPP-MOU, White et al. (2022)) and was very effective in solving three standard benchmark test functions and a computationally expensive constrained groundwater MOO problem. The tests demonstrated the benefits of using SAMOO:

1. SAMOO performed a lot better in terms of: (a) Computational effort required, (b) diversity of solutions in the front, (c) overcoming initial infeasibility due to constraints, and (d) achieving full convergence of solutions in the final front.
2. SAMOO was remarkably effective in attaining a fully converged Pareto front of a constrained problem, which even the traditional MOO (i.e., PBMOO) algorithm cannot achieve.
3. The algorithm facilitates efficient machine learning and its benefit can be fully realized through parallelization of runs.

Although the application was a groundwater problem, we reiterate that the algorithm can be easily used in a wide range of management problems where computationally expensive simulation models are required. It is important to emphasize, however, that a single, sufficiently calibrated PBM is used to generate the model outputs for the training data set of the surrogate model, and that our method does not currently take into account parameter or conceptual uncertainties, which entails having ensmebles of parameters or multiple models as inputs to GPR alongside decision variables. Such complexity remains a topic of further research.

Further studies and testing need to be done on problems that involve higher dimension of decision variable where GPR is known to have some performance issues. In such case, the algorithm may require dimensional reduction methods, for example, partial least squares regression or active subspace method, prior to training the surrogate model, such that the surrogate model only sees the dimensionally-reduced decision space.

It may also be worthwhile to explore greedy selection of infills to prevent redundancies, thereby improving convergence and diversity of $\mathcal{P}$. Investigating the benefits of a greedy algorithm in this way is a topic of further research. Nevertheless, our results showed that using the presented fitness formula for selecting the infill ensemble still performed very well.

Appendix A: Benchmark Test Functions

Zitzler et al. (2000):

$$\begin{aligned} \min_{\mathbf{x}} \mathbf{\Phi}(\mathbf{x}) : &= \{\phi_1(\mathbf{x}), \phi_2(\mathbf{x})\} \\ \phi_1(\mathbf{x}) &= x_1 \\ \phi_2(\mathbf{x}) &= g(\mathbf{x}) h\big(\phi_1(\mathbf{x}), g(\mathbf{x})\big) \\ g(x) &= 1 + \frac{9}{29} \sum_{i=2}^{30} x_i \\ h(f(x_1), g(\mathbf{x})) &= 1 - \left(\frac{\phi_1(\mathbf{x})}{g(\mathbf{x})}\right)^2 \end{aligned} \tag{A1}$$

subject to:

$$\begin{aligned} 0 \le x_i \le 1 \\ 1 \le i \le 30 \end{aligned} \tag{A2}$$

Kursawe (1991):

$$\begin{aligned} \min_{\mathbf{x}} \mathbf{\Phi}(\mathbf{x}) : &= \{\phi_1(\mathbf{x}), \phi_2(\mathbf{x})\} \\ \phi_1(\mathbf{x}) &= \sum_{i=1}^{2} \left(-10 \exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right) \\ \phi_2(\mathbf{x}) &= \sum_{i=1}^{3} \left(|x_i|^{0.8} + 5\sin\left(x_i^3\right)\right) \end{aligned} \tag{A3}$$

subject to:

$$-5 \le x_1, x_2, x_3 \le 5 \tag{A4}$$

Osyczka and Kundu (1995):

$$\begin{aligned} \min_{\mathbf{x}} \mathbf{\Phi}(\mathbf{x}) : &= \{\phi_1(\mathbf{x}), \phi_2(\mathbf{x})\} \\ \phi_1(\mathbf{x}) &= -25(x_1 - 2)^2 - (x_2 - 2)^2 - (x_3 - 1)^2 - (x_4 - 4)^2 - (x_5 - 1)^2 \\ \phi_2(\mathbf{x}) &= \sum_{i=1}^{6} x_i^2 \end{aligned} \tag{A5}$$

subject to:

$$\begin{aligned} x_1 + x_2 - 2 &\geq 0 \\ 6 - x_1 - x_2 &\geq 0 \\ 2 - x_2 + x_1 &\geq 0 \\ 2 - x_1 + 3x_2 &\geq 0 \\ 4 - (x_3 - 3)^2 - x_4 &\geq 0 \\ (x_5 - 3)^2 + x_6 - 4 &\geq 0 \\ 0 \leq x_1, x_2, x_6 &\leq 10 \\ 1 \leq x_3, x_5 &\leq 5 \\ 0 \leq x_4 &\leq 6 \end{aligned} \tag{A6}$$

Appendix B: Seawater Intrusion Model Parameters

Characteristics of the model domain:

- Depth of 90 m and length of 8,000 m and discretized with 90 layers and 4,000 columns (1m × 2m for each cell) for a total of 360,000 cells in the entire domain;
- Ground surface is at layer 20 with elevation of 0 m above sea level (masl);
- General-head boundary located in column 4,000 from layer 20 to the bottom represent the sea with a salinity of 35.0 g/L and a head of 0 masl;
- Specified flux of 0.2 m^3/d per layer in the upgradient boundary column 1 from layer 6 to the bottom;
- Natural recharge (i.e., average precipitation less run-off) is uniformly applied only on the top of the domain at a rate of 0.82 mm/day;
- Horizontal and vertical conductivities for sand are 150m/d and 15m/d, respectively; horizontal and vertical conductivities for clay are 1.0 m/d and 0.01 m/d, respectively;
- The inland segment of the clay layer is located in layers 60–65 and spans from column 1 to 2,639 while the its coastal segment is in layers 75–78 and from columns 2,720 to 4,000;
- Four vertically aligned extraction wells are each located at columns 1,250 (W1–W4) and 1,800 (W6–W8); four of these are shallow wells tapping the unconfined sandy aquifer in layers 40 (W1, W5) and 55 (W2, W6) while the other four extracts water from the confined aquifer in layers 70 (W3, W7) and 85 (W4, W8);
- A GDE site is located in column 2,800 where drawdown is monitored;
- Two injection bores—one shallow (B1), located at layer 40 and column 2,750 near the GDE site, and one deep (B2), located at layer 85 and column 2,000—serve as artificial barriers against seawater intrusion.

Appendix C: Hypervolume-Based Metrics

Relative hypervolume difference (RHD) (Wang et al., 2022) and hypervolume improvment (HVI; also known as hypervolume contribution) (Guerreiro et al., 2021; Yang et al., 2019) are only some of widely-used metrics to assess convergence and diversity of the Pareto front. These metrics may be used as a stopping criteria for outer iterations; however, it is still strongly advised to use these in conjunction with qualitative assessment of the front. These should never be used to assess the convergence and diversity of the Pareto cloud in inner iterations. Lastly, these should be used with caution because the numbers may look like convergence has been achieved, but solutions may not be diverse or the algorithm is still overcoming infeasibility or premature convergence in a local minima. Figure C1 illustrates the concept of hypervolume which is needed to compute RHD through Equation C1. HVI is also illustrated in Figure C1

$$\text{RHD} = \frac{\text{HV}'}{\text{HV}_{nd}} \tag{C1}$$

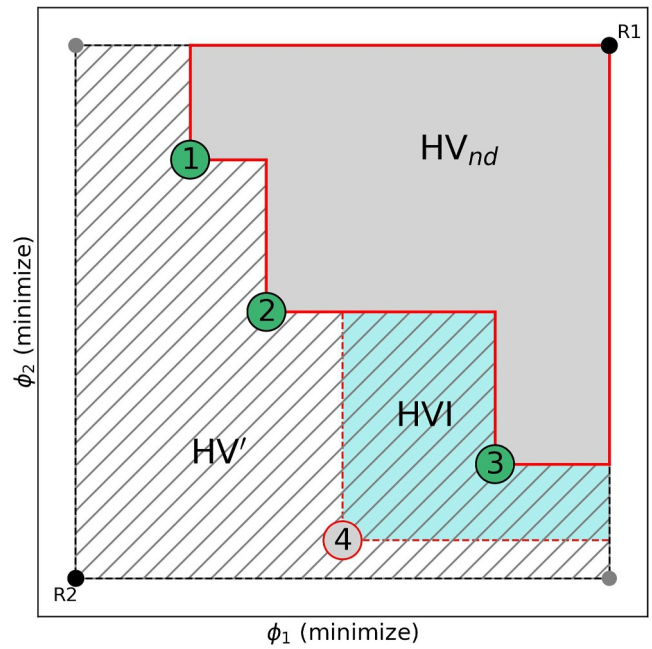

Figure C1. HV$_{nd}$ is the hypervolume enclosed by the reference point R1 and currently non-dominated positions (gray area); HV′ is the area within the rectangle bounded by R1 and R2 as opposite vertices that is outside HV$_{nd}$ (hatched area); HVI is the change in hypervolume when a new non-dominated position is added to the repository (light blue area).

Data Availability Statement

This research modified the source code of the original PESTPP-MOU (White et al., 2022), and the resulting full source code is provided in https://doi.org/10.5281/zenodo.13263937 (Macasieb, 2024a). All data required to execute the benchmarks and applications in this study are provided in https://doi.org/10.5281/zenodo.13262713 (Macasieb, 2024b).

Acknowledgments
Funding for this work was provided by the Department of Water and Environmental Regulation (DWER) of Western Australia through the project titled, "Perth Regional Aquifer Model version 3.6 (PRAMS 3.6): Construction, history matching and predictive uncertainty." Scientific computing resources were provided by the Commonwealth Scientific and Industrial Research Organisation (CSIRO) through the Petrichor HPC cluster. Open access publishing facilitated by The University of Western Australia, as part of the Wiley - The University of Western Australia agreement via the Council of Australian University Librarians.

References

Ahlfeld, D. P., & Mulligan, A. E. (2000). *Optimal management of flow in groundwater systems*. Academic Press.

Alaviani, F., Sedghi, H., Asghari Moghaddam, A., & Babazadeh, H. (2018). Adopting GMS–PSO Model to reduce groundwater withdrawal by integrated water resources management. *International Journal of Environmental Research*, *12*(5), 619–629. https://doi.org/10.1007/s41742-018-0115-x

Asher, M. J., Croke, B. F. W., Jakeman, A. J., & Peeters, L. J. M. (2015). A review of surrogate models and their application to groundwater modeling. *Water Resources Research*, *51*(8), 5957–5973. https://doi.org/10.1002/2015WR016967

Baltar, A. M., & Fontane, D. G. (2008). Use of multiobjective particle swarm optimization in water resources management. *Journal of Water Resources Planning and Management*, *134*(3), 257–265. https://doi.org/10.1061/(asce)0733-9496(2008)134:3(257)

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Brochu, E., Cora, V. M., & de Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Retrieved from http://arxiv.org/abs/1012.2599

Cheng, R., Jin, Y., Olhofer, M., & Sendhoff, B. (2016). A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, *20*(5), 773–791. https://doi.org/10.1109/TEVC.2016.2519378

Coello, C., Pulido, G., & Lechuga, M. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 256–279. https://doi.org/10.1109/TEVC.2004.826067

Dantzig, G. B., Ordern, A., & Wolfe, P. (1955). The generalized simplex method. *Pacific Journal of Mathematics*, *5*(2), 183–195.

Datta, R., & Regis, R. G. (2016). A surrogate-assisted evolution strategy for constrained multi-objective optimization. *Expert Systems with Applications*, *57*, 270–284. https://doi.org/10.1016/j.eswa.2016.03.044

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Ceur workshop proceedings*, *1133*, 849–858. https://doi.org/10.1007/3-540-45356-3{_}83

Deb, K., Roy, P. C., & Hussein, R. (2020). Surrogate modeling approaches for multiobjective optimization: Methods, taxonomy, and results. *Mathematical and Computational Applications*, *26*(1), 5. https://doi.org/10.3390/mca26010005

Doherty, J., & Moore, C. (2020). Decision support modeling: Data assimilation, uncertainty quantification, and strategic abstraction. *Groundwater Series*, *58*(3), 327–337. https://doi.org/10.1111/gwat.12969

Emmerich, M., Yang, K., Deutz, A., Wang, H., & Fonseca, C. M. (2016). A multicriteria generalization of Bayesian global optimization. In P. Pardalos, A. Zhigljavsky, & J. Žilinskas (Eds.), *Advances in stochastic and deterministic global optimization. Springer optimization and its applications* (Vol. 107, pp. 229–242). Springer. https://doi.org/10.1007/978-3-319-29975-4_12

Emmerich, M. T. M., & Deutz, A. H. (2018). A tutorial on multiobjective optimization: Fundamentals and evolutionary methods. *Natural Computing*, *17*(3), 585–609. https://doi.org/10.1007/s11047-018-9685-y

Emmerich, M. T. M., Giannakoglou, K. C., & Naujoks, B. (2006). Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, *10*(4), 421–439. https://doi.org/10.1109/TEVC.2005.859463

Fieldsend, J. E., & Everson, R. M. (2005). Multi-objective optimisation in the presence of uncertainty. In *2005 IEEE congress on evolutionary computation*. https://doi.org/10.1109/CEC.2005.1554691

Forrester, A. I. J., & Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, *45*(1–3), 50–79. https://doi.org/10.1016/j.paerosci.2008.11.001

Forrester, A. I. J., Sóbester, A., & Keane, A. J. (2008). Engineering design via surrogate modelling. https://doi.org/10.1002/9780470770801

Frazier, P. I. (2018). A tutorial on Bayesian optimization (No. Section 5). Retrieved from http://arxiv.org/abs/1807.02811

Gaur, S., Dave, A., Gupta, A., Ohri, A., Graillot, D., & Dwivedi, S. B. (2018). Application of artificial Neural networks for identifying optimal groundwater pumping and piping network layout. *Water Resources Management*, *32*(15), 5067–5079. https://doi.org/10.1007/s11269-018-2128-9

Gelbart, M. A., Snoek, J., & Adams, R. P. (2014). Bayesian optimization with unknown constraints. In *Uncertainty in artificial intelligence - proceedings of the 30th conference, uai 2014* (pp. 250–259). Retrieved from http://arxiv.org/abs/1403.5607

Gorelick, S. M., & Zheng, C. (2015). Global change and the groundwater management challenge. *Water Resources Research*, *64*(46), 929. https://doi.org/10.1029/eo064i046p00929-04

Gramacy, R. B. (2016). LaGP: Large-scale spatial modeling via local approximate Gaussian processes in R. *Journal of Statistical Software*, *72*(1). https://doi.org/10.18637/jss.v072.i01

Gu, Q., Wang, Q., Li, X., & Li, X. (2021). A surrogate-assisted multi-objective particle swarm optimization of expensive constrained combinatorial optimization problems[Formula presented]. *Knowledge-Based Systems*, *223*, 107049. https://doi.org/10.1016/j.knosys.2021.107049

Guerreiro, A. P., Fonseca, C. M., & Paquete, L. (2021). The hypervolume indicator: Computational problems and algorithms. *Association for Computing Machinery*, *54*(6), 1–42. https://doi.org/10.1145/3453474

Heidari, M., Chow, V. T., Kokotović, P. V., & Meredith, D. D. (1971). Discrete differential dynamic programing approach to water resources systems optimization. *Water Resources Research*, *7*(2), 273–282. https://doi.org/10.1029/WR007i002p00273

Hughes, E. J. (2001). Evolutionary multi-objective ranking with uncertainty and noise. In E. Zitzler, L. Thiele, K. Deb, C. A. Coello, & D. Corne (Eds.), *Evolutionary multi-criterion optimization. emo 2001. lecture notes in computer science* (Vol. 1993). Springer. https://doi.org/10.1007/3-540-44719-9{_}23

Jin, R., Chen, W., & Simpson, T. W. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, *23*(1), 1–13. https://doi.org/10.1007/s00158-001-0160-4

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, *21*(4), 345–383. https://doi.org/10.1023/A:1012771025575

Khosravi, F., Borst, M., & Teich, J. (2018). Probabilistic dominance in robust multi-objective optimization. In *2018 ieee congress on evolutionary computation (cec)* (pp. 1–6). https://doi.org/10.1109/CEC.2018.8477676

Khosravi, F., Rass, A., & Teich, J. (2021). Efficient computation of probabilistic dominance in multi-objective optimization. *ACM Transactions on Evolutionary Learning and Optimization*, *1*(4), 1–26. https://doi.org/10.1145/3469801

Kim, M., Hiroyasu, T., Miki, M., & Watanabe, S. (2004). SPEA2+: Improving the performance of the strength pareto evolutionary algorithm 2. *Lecture Notes in Computer Science*, *3242*, 742–751. https://doi.org/10.1007/978-3-540-30217-9{_}75

Kleijnen, J. P. (2009). Kriging metamodeling in simulation. *A review*, *192*(3), 707–716. https://doi.org/10.1016/j.ejor.2007.10.013

Kollat, J. B., Reed, P. M., & Wagener, T. (2012). When are multiobjective calibration trade-offs in hydrologic models meaningful? *Water Resources Research*, *48*(3). https://doi.org/10.1029/2011WR011534

Kursawe, F. (1991). A variant of evolution strategies for vector optimization. *Lecture Notes in Computer Science*, *496*(LNCS), 193–197. https://doi.org/10.1007/BFb0029752

Langevin, C. D., Hughes, J. D., Banta, E. R., Niswonger, R. G., Panday, S., & Provost, A. M. (2017). *Documentation for the MODFLOW 6 Groundwater Flow Model: U.S. Geological survey techniques and methods* (Vol. Book 6, p. 197). Modeling Techniques. https://doi.org/10.3133/tm6A55

Langevin, C. D., Provost, A. M., Panday, S., & Hughes, J. D. (2022). *Documentation for the MODFLOW 6 groundwater transport model: U.S. Geological survey techniques and methods*. (Vol. Book 6). Modeling Techniques. https://doi.org/10.3133/tm6A61

Li, H., Nichols, P. G., Han, S., Foster, K. J., Sivasithamparam, K., & Barbetti, M. J. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359. https://doi.org/10.1023/A:1008202821328

Lim, W. J., Jambek, A. B., & Neoh, S. C. (2015). Kursawe and ZDT functions optimization using Hybrid Micro Genetic Algorithm (HMGA). *Soft Computing*, *19*(12), 3571–3580. https://doi.org/10.1007/s00500-015-1767-5

Macasieb, R. (2024a). Modified PESTPP-MOU to accommodate surrogate-assisted multi-objective optimization [Software]. *Zenodo*. https://doi.org/10.5281/zenodo.13263937

Macasieb, R. (2024b). Scripts and template files for benchmark tests and application problem used for SAMOO development [Dataset]. *Zenodo*. https://doi.org/10.5281/zenodo.13262713

Mathai, A. M., & Provost, S. B. (1992). Moments of a quadratic form in the nonsingular normal case. In *Quadratic forms in random variables: Theory and application* (pp. 51–54). Marcel Dekker, Inc.

Miettinen, K. (1998). *Nonlinear multiobjective optimization* (Vol. 12). Springer US. https://doi.org/10.1007/978-1-4615-5563-6

Mo, S., Lu, D., Shi, X., Zhang, G., Ye, M., Wu, J., & Wu, J. (2017). A taylor expansion-based adaptive design strategy for global surrogate modeling with applications in groundwater modeling. *Water Resources Research*, *53*(12), 10802–10823. https://doi.org/10.1002/2017WR021622

Mockus, J., Tiesis, V., & Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum. *Towards Global Optimisation*, *2*(2), 117–129.

Moradi, A. M., & Dariane, A. B. (2009). Particle swarm optimization: Application to reservoir operation problems. In *2009 IEEE international advance computing conference, IACC 2009(March)* (pp. 1048–1051). https://doi.org/10.1109/IADCC.2009.4809159

Mozaffari, S., Javadi, S., Moghaddam, H. K., & Randhir, T. O. (2022). Forecasting groundwater levels using a hybrid of support vector regression and particle swarm optimization. *Water Resources Management*, *36*(6), 1955–1972. https://doi.org/10.1007/s11269-022-03118-z

Nishikawa, T., Siade, A. J., Reichard, E. G., Ponti, D. J., Canales, A. G., & Johnson, T. A. (2009). Stratigraphic controls on seawater intrusion and implications for groundwater management, Dominguez Gap area of Los Angeles, California, USA. *Hydrogeology Journal*, *17*(7), 1699–1725. https://doi.org/10.1007/s10040-009-0481-8

Osyczka, A., & Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, *10*(2), 94–99. https://doi.org/10.1007/BF01743536

Rajanayaka, C., Weir, J., Kerr, T., & Thomas, J. (2021). Sustainable water resource management using surface-groundwater modelling: Motueka-Riwaka Plains, New Zealand. *Watershed Ecology and the Environment*, *3*, 38–56. https://doi.org/10.1016/j.wsee.2021.08.001

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Razavi, S., Tolson, B. A., & Burn, D. H. (2012). Review of surrogate modeling in water resources. *Water Resources Research*, *48*(7). https://doi.org/10.1029/2011WR011527

SaberChenari, K., Abghari, H., & Tabari, H. (2016). Application of PSO algorithm in short-term optimization of reservoir operation. *Environmental Monitoring and Assessment*, *188*(12), 667. https://doi.org/10.1007/s10661-016-5689-1

Sawyer, C. S., & Lin, Y.-f. (1998). Mixed-integer chance-constrained models for ground-water remediation. *Journal of Water Resources Planning and Management*, *124*(5), 285–294. https://doi.org/10.1061/(asce)0733-9496(1998)124:5(285)

Siade, A. J., Cui, T., Karelse, R. N., & Hampton, C. (2020). Reduced-dimensional Gaussian process machine learning for groundwater allocation planning using swarm theory. *Water Resources Research*, *56*(3). https://doi.org/10.1029/2019WR026061

Siade, A. J., Hall, J., & Karelse, R. N. (2017). A practical, robust methodology for acquiring new observation data using computationally expensive groundwater models. *Water Resources Research*, *53*(11), 9860–9882. https://doi.org/10.1002/2017WR020814

Siade, A. J., Rathi, B., Prommer, H., Welter, D., & Doherty, J. (2019). Using heuristic multi-objective optimization for quantifying predictive uncertainty associated with groundwater flow and reactive transport models. *Journal of Hydrology*, *577*(July), 123999. https://doi.org/10.1016/j.jhydrol.2019.123999

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. Retrieved from http://arxiv.org/abs/1206.2944

Vahdat-Aboueshagh, H., Tsai, F. T., Habib, E., & Clement, T. P. (2022). Multi-objective optimization of aquifer storage and recovery operations under uncertainty via machine learning surrogates. *Journal of Hydrology*, *612*, 128299. https://doi.org/10.1016/j.jhydrol.2022.128299

Vanderzalm, J., Page, D., Dillon, P., Gonzalez, D., & Petheram, C. (2022). Assessing the costs of managed aquifer recharge options to support agricultural development. *Agricultural Water Management*, *263*(July 2021), 107437. https://doi.org/10.1016/j.agwat.2021.107437

Wang, L., Yao, Y., Zhang, T., Adenutsi, C. D., Zhao, G., & Lai, F. (2022). A novel self-adaptive multi-fidelity surrogate-assisted multi-objective evolutionary algorithm for simulation-based production optimization. *Journal of Petroleum Science and Engineering*, *211*, 110111. https://doi.org/10.1016/j.petrol.2022.110111

White, J. T., Knowling, M. J., Fienen, M. N., Siade, A., Rea, O., & Martinez, G. (2022). A model-independent tool for evolutionary constrained multi-objective optimization under uncertainty. *Environmental Modelling & Software*, *149*, 105316. https://doi.org/10.1016/j.envsoft.2022.105316

Wilson, J. T., Hutter, F., & Deisenroth, M. P. (2018). Maximizing acquisition functions for Bayesian optimization. *Advances in Neural Information Processing Systems, 2018-Decem(NeurIPS)*, 9884–9895.

Xiaohui, H., & Eberhart, R. (2002). Multiobjective optimization using dynamic neighborhood particle swarm optimization. *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002* (Vol. 2, 1677–1681). https://doi.org/10.1109/CEC.2002.1004494

Yang, K., Emmerich, C., Deutz, A., & Bäck, T. (2019). Multi-Objective Bayesian global optimization using expected hypervolume improvement gradient. *Swarm and Evolutionary Computation*, *44*(September 2017), 945–956. https://doi.org/10.1016/j.swevo.2018.10.007

Zare, M., & Koch, M. (2021). Hybrid signal processing/machine learning and PSO optimization model for conjunctive management of surface–groundwater resources. *Neural Computing & Applications*, *33*(13), 8067–8088. https://doi.org/10.1007/s00521-020-05553-8

Zhao, M., Zhang, K., Chen, G., Zhao, X., Yao, C., Sun, H., et al. (2020). A surrogate-assisted multi-objective evolutionary algorithm with dimension-reduction for production optimization. *Journal of Petroleum Science and Engineering*, *192*(March), 107192. https://doi.org/10.1016/j.petrol.2020.107192

Zheng, N., Wang, H., & Yuan, B. (2022). An adaptive model switch-based surrogate-assisted evolutionary algorithm for noisy expensive multi-objective optimization. *Complex & Intelligent Systems*, *8*(5), 4339–4356. https://doi.org/10.1007/s40747-022-00717-6

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, *8*(2), 173–195. https://doi.org/10.1162/106365600568202

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. https://doi.org/10.1109/4235.797969