Large-scale Parallelization of the Borg MOEA to Enhance the Management of Complex Environmental Systems

David Hadka

Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, 16802, USA

Patrick Reed

School of Civil and Environmental Engineering, Cornell University, Ithaca, NY, 14853, USA

Abstract

The Borg MOEA is a self-adaptive multiobjective evolutionary algorithm capable of solving complex, many-objective environmental systems problems efficiently and reliably. Water and environmental resources problems pose significant computational challenges due to their potential for large Pareto optimal sets, the presence of disjoint Pareto-optimal regions that arise from discrete choices, multi-modal suboptimal regions, and expensive objective function calculations. This work develops two large-scale parallel implementations of the Borg MOEA, the master-slave and multi-master Borg MOEA, and applies them to a highly challenging risk-based water supply portfolio planning problem. The performance and scalability of both implementations are compared on up to 16384 processors. The multi-master Borg MOEA is shown to scale efficiently on tens of thousands of cores while dramatically improving the reliability of attaining high-quality solutions. Our results dramatically expand the scale and scope of complex environmental systems that can be addressed using many-objective evolutionary optimization.

Keywords: Evolutionary algorithm, Borg MOEA, multiobjective optimization, large-scale parallelization

1. Introduction

The role of evolutionary computation in water resources systems planning and management as described in Maier et al. (In-Press) as well as the recent

Email addresses: dmh309@psu.edu (David Hadka), patrick.reed@cornell.edu (Patrick Reed)

review by Nicklow et al. (2010) is to advance our field's ability to address the mathematical complexities inherent to real decision contexts (i.e., multiple objectives, mixed real and discrete decisions, uncertainty, computational demanding simulations, etc.). Real water resources applications present substantial challenges given the evolving and highly uncertain impacts of climate change, rapid urbanization, and growing resource contention. These challenges present fundamental tradeoffs in water systems across time, space, and economic sectors (National Research Council, 2009, 2012). More formally from the optimization perspective, understanding the optimal balance between these tradeoffs requires evolutionary algorithms to approximate the Pareto set of solutions. These solutions encompass those water resources alternatives where improvement in one objective can only be improved by sacrificing performance in one or more other objectives (Cohon and Marks, 1975). The tradeoffs posed by managing water resources systems under change was directly discussed as a leading research challenge in the National Research Council's (NRC) recent vision for the future of global hydrology (National Research Council, 2012). The NRC highlights the need for translational science innovations that combine simulation, optimization, and high-performance computing to innovate real decision making. Complimentary to this vision, our core goals in this study are to advance: (1) a rigorous diagnostic framework for benchmarking massively parallel multiobjective evolutionary algorithms' (MOEAs) ability to discover the tradeoffs for highly challenging water resources problems and (2) fundamentally expand the capability of evolutionary multiobjective search to support decision making in computationally intensive applications. To address these goals, this study presents two parallel variants of the self-adaptive Borg MOEA (Hadka and Reed, 2013, 2012). Our parallelization study generalizes the Borg MOEAs auto-adaptive search to a massively parallel context to better discover effective search strategies that work cooperatively to tailor exploration for severely challenging water resources applications. Moreover, our parallel search diagnostics clarify the relative merits of the classical master-slave Borg MOEA versus a multi-master hierarchical parallelization architecture (multi-master Borg MOEA) that allows scalable search on emerging high-performance computing platforms. These innovations have value for water resources systems planning as well as other fields currently applying parallel evolutionary search. Alba et al. (2013) provide a detailed review of parallel metaheuristics that highlights there is at present a need for studies addressing auto-adaptivity in parallel search, that provide careful theoretical assessment of the scalability of multiobjective algorithm architectures, provide tools for minimizing serial bottlenecks in the algorithms' architectures, and contribute rigorous statistical diagnostics for multiple parallelization strategies. The present study makes important contributions in each of these areas.

Both parallel variants of the Borg MOEA are applied to a highly challenging water resources management application: a risk-based water supply portfolio planning problem focused on the Lower Rio Grande Valley (LRGV, (Kasprzyk et al., 2009, 2012, 2013)). This problem is many-objective, nonlinear, contains a mix of discrete and real decision variables, is severely-constrained,

and it has stochastic objectives with expensive function evaluation times. In the largest systematic benchmarking of serial MOEAs in the water resources literature to date (Reed et al., 2013), the LRGV application's complex disjoint Pareto optimal regions caused ten state-of-the-art MOEAs (including the Borg MOEA) to fail to solve this problem reliably. As noted by Tang et al. (2007), problem difficulty is critical for distinguishing alternative MOEA parallelization strategies. In simple master-slave approaches where only the function evaluations are parallelized, the internal search of the original serial algorithm is not modified. Consequently, master-slave strategies simply provide the ability to compute more function evaluations in a fixed wall clock period. Tang et al. (2007) show that often perceived serial MOEA failures are simply the result of not having enough function evaluations given users' limits on search time and not the mathematical difficulty of the underlying water resources applications. They demonstrated this effect for hydrologic model calibration and combinatorial groundwater monitoring design applications.

Alternatively, the LRGV test case selected in this study alternatively poses sufficient problem difficulty as will be shown in our results that the simple master-slave Borg MOEA implementation is fully statistically inferior to the proposed multi-master Borg MOEA parallelization that generalizes the algorithm's auto-adaptivity to modern leadership class supercomputing systems (i.e., thousands of processors). It is worth noting that virtually all of the recent parallel evolutionary computation efforts in the water resources systems literature employ simple master-slave variants of existing popular algorithms that have significant serial bottlenecks in their base algorithmic architectures (e.g., water distribution systems (Guidolin et al., 2012; Roshani and Filion, 2012; Zheng and Morad, 2012a,b); model calibration (Feyen et al., 2007; Tang et al., 2007; Vrugt et al., 2008; Zhang et al., 2013); and groundwater management (Kollat et al., 2011; Matott et al., 2006b,a; Reed and Kollat, 2013; Tang et al., 2007)). Moreover, a vast majority of these studies employed fewer than 500 processors and in the few instances of careful reporting of parallel scalability, their efficiencies decline severely with their maximum processor counts. Parallel efficiency refers to the ratio of actual speedup to theoretical speedup. For example, 100-percent efficiency when using 1000 processors requires a 1000-fold reduction in the wall clock time of a search application. The strong declines in parallel efficiency in the prior water resources literature are in fact theoretically expected given their use of master-slave architectures and existing MOEAs that have significant serial bottlenecks in their algorithmic architectures (e.g., generational selection, mating, mutation, and solution sorting). In master-slave architectures, an increasing number of workers increases communication costs and processing time at the master node, bounding the theoretical speedup attainable. Moreover, as described by Amdahl's law, the speedup of any program is limited by the serial portion of the program (Amdahl, 1967, 1988). Alternatively, this study contributes a self-adaptive multi-master Borg MOEA whose algorithmic architecture minimizes serial bottlenecks and exploits a parallelization strategy that reduces communication costs.

The master-slave and multi-master parallel variants of the Borg MOEA de-

veloped in this study make it possible to exploit leadership class computing systems with thousands or tens of thousands of processors to significantly improve convergence speed, solution quality, and reliability. As noted in the lead vision paper Maier et al. (In-Press), it is important to view the value of parallel metaheuristics in the broader continuum of advancing computing capabilities. Summary statistics for geographic and institutional access to increasingly powerful computing architectures shows the continued exponential growth in capability world-wide (Top 500 Supercomputer Sites, 2014). Moreover, these statistics clearly show that the supercomputers of today are the broadly available work stations of tomorrow. This study advances the water resources field's ability to exploit these trends in rapid and high-quality optimization.

Although this study focuses solely on parallelization, it should be noted that future studies can combine the master-slave and multi-master Borg MOEA advances with other efficiency enhancement strategies (e.g., response surface modeling (Razavi et al., 2012), emulation (Castelletti et al., 2012), global-local hybrid search (Sayeed and Mahinthakumar, 2005), pre-conditioning (Fu et al., 2013; Kollat and Reed, 2006), problem decomposition (Castelletti et al., 2012; Fu et al., 2013), etc.). Readers interested in these alternative efficiency enhancement strategies can reference the more detailed reviews by Nicklow et al. (2010); Maier et al. (In-Press). It is worth noting the relative concerns and consequences that users should consider when choosing parallelization and/or other commonly employed efficiency enhancement strategies for MOEAs. Beyond a simple focus on efficiency, an additional key question is whether or not a efficiency enhancement strategy precludes the exploration of important problem formulation hypotheses (Kasprzyk et al., 2013). The scope of problem formulations is fundamentally tied to the optimization strategies employed, and these interdependent choices are a significant potential source for negative decision biases if the system's complexities and uncertainties are overly simplified. Besides parallelization, response surface methods [see review for the water resources field in Razavi et al. (2012)] and their specific use in evolutionary algorithms termed fitness approximation [see general review for the metaheuristics field in Jin et al. (2002)] are the most popular strategies for reducing the computational demands. Jin et al. (2002) strongly demonstrate that these approximate evaluation techniques can actually severely degrade applications if not implemented carefully.

Response surfaces require offline or online training where the original computationally expensive model is used to evaluate the objectives for a statistical sampling of candidate decisions distributed throughout a problem's space of alternatives (Razavi et al., 2012). Response surface methods simply approximate the mapping from decisions to objectives. They do not provide insights into the spatial or temporal gradients of the states of the systems of interest (e.g., hydraulic heads or concentrations). This is problematic for data assimilation, uncertainty analysis, and state-based control optimization. Moreover in the multiobjective optimization context, every objective requires its own statistical design of experiments and a unique response surface, which poses a computational barrier onto itself. The water resources applications reviewed by

Razavi et al. (2012) highlights response surface methods have been limited in the number of decision variables that can be optimized (typically < 10) because training demands become severe. Even some of the most advanced work in this area (Regis and Shoemaker, 2007) bridging parallel search and response surface methods still struggles to address larger numbers of decisions, especially if they encompass a mix of real and integer decisions. Alternatively, dynamic emulation (Castelletti et al., 2012) and decomposition techniques (Fu et al., 2013) that exploit simplified model or problem variants to inform subsequent search are more flexible but still face fundamental challenges. To be successful they must capture the complex multivariate dependencies that can occur with increasing numbers of conflicting objectives even for simple yes/no decisions (Shah et al., 2011). Often emulation or decomposition approaches are strongly application specific and limited in their generalizable value. Moreover, there is a general bias in the published literature exploring MOEA efficiency enhancement schemes to date to consider typically two objective problem formulations [see discussions in Maier et al. (In-Press); Nicklow et al. (2010)]. Given the growing capabilities and interest in "many-objective", there have been several studies that have highlighted that traditional two objective formulations can strongly limit the identification of alternatives to extreme regions of the decision space whose performance is often considered strongly inferior when decision-makers are able to consider additional design relevant performance measures (Brill et al., 1990; Franssen, 2005; Woodruff et al., 2013). The master-slave and multi-master Borg MOEA variants introduced in this study seek to provide robust search across multiple competing problem formulation hypotheses without necessarily requiring a simplification of the evaluation model while also minimizing the amount of wall clock time required.

The remainder of this study is organized as follows. Section 2 overviews the serial Borg MOEA. Section 3 presents the master-slave and multi-master parallel extensions of the Borg MOEA. Section 4 describes the risk-based water supply portfolio planning problem used to benchmark the parallelization schemes. Section 5 overviews the experimental setup and Section 6 presents the results from this experiment. Conclusions and future work are presented in Section 7.

2. The Serial Borg MOEA

The Borg MOEA consists of three key components: (1) an ϵ-dominance archive to maintain a diverse set of Pareto approximate solutions, (2) an ϵ-progress restart mechanism triggered by search stagnation to avoid preconvergence to local optima, and (3) the use of multiple search operators that adapt to a given problem's landscape (Hadka and Reed, 2013). These components are adaptive in nature, allowing the Borg MOEA to adapt to local search conditions encountered in challenging problems. Key details of the algorithm are summarized below to facilitate readers in more fully understanding the auto-adaptive nature of the master-slave and multi-master Borg MOEA variants contributed in this study.

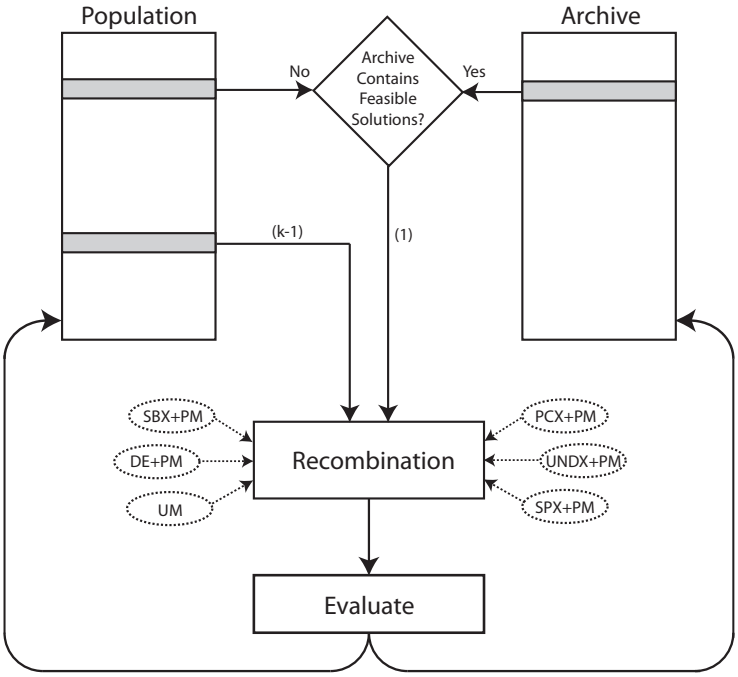

Figure 1: Flowchart of the Borg MOEA main loop adapted from Hadka and Reed (2013). First, one of the recombination operators is selected using the adaptive multi-operator procedure described in Section 2.1. For a recombination operator requiring k parents, $k-1$ parents are selected from the population using tournament selection. The remaining parent is selected randomly from the archive if the archive contains feasible solutions; otherwise it is selected randomly from the population. The offspring resulting from this operator are evaluated and then considered for inclusion in the population and archive.

2.1. Auto-Adaptive Multi-Operator Search

The Borg MOEA exploits the following six search operators:

1. Simulated Binary Crossover (SBX) (Deb and Agrawal, 1994)
2. Differential Evolution (DE) (Storn and Price, 1997)
3. Parent-Centric Crossover (PCX) (Deb et al., 2002)
4. Simplex Crossover (SPX) (Tsutsui et al., 1999)
5. Unimodal Normal Distribution Crossover (UNDX) (Kita et al., 1999)
6. Uniform Mutation (UM) applied with probability $1/L$

In addition, offspring produced by SBX, DE, PCX, SPX, and UNDX are mutated using polynomial mutation (PM) (Deb and Agrawal, 1994). It should be noted that these operators provide a variety of offspring distributions. For instance, SBX, PCX, and PM produce offspring near one of the parents. Such small perturbations helps fine-tune existing designs. SPX and DE result in larger perturbations, allowing the MOEA to translate across large landscapes efficiently. UNDX produces offspring about the centroid of the parents, quickly converging to valleys in the landscape. UM is the most disruptive of the operators, which aids in adding diversity to the population to prevent preconvergence.

Another key difference between these operators is rotational-invariance. In the ideal case, all decision variables are independent and can thus be optimized independently. However, it is common in complex environmental systems to encounter large amounts of interaction (epistasis) between decision variables. SBX and PM are tailored for problems with independent decision variables. PCX, SPX, UNDX, and DE are rotationally-invariant, and will often perform better on non-separable, epistatic problems. The Borg MOEA uses all six operators, but adapts the probability that each operator is applied based on the success of each operator from prior iterations.

2.2. ϵ-Progress Triggered Restarts

Since the ϵ-dominance archive is the set of all non-dominated solutions produced by the MOEA, Hadka and Reed (2013) propose monitoring the ϵ-dominance archive to detect search stagnation. If no new non-dominated solutions are accepted into the ϵ-dominance archive over a period of time, then the MOEA has stagnated. For instance, the MOEA may be stuck at a local optima. This mechanism of monitoring the ϵ-dominance archive for search stagnation is called ϵ-progress. In the Borg MOEA, if the entire population is evolved and the ϵ-dominance archive remains unchanged (no ϵ-progress), then a restart is triggered.

A restart involves several steps designed to help the algorithm escape local optima and introduce additional diversity into the search population. First, the population is emptied. Second, the population is resized relative to the ϵ-dominance archive. Several studies theoretically and experimentally demonstrate that maintaining a population size relative to the Pareto approximate set, as inferred by the ϵ-dominance archive size, helps avoid preconvergence (Horn, 1995; Mahfoud, 1995; Kollat and Reed, 2006; Hadka and Reed, 2013). Finally,

the population is filled with all solutions in the ϵ-dominance archive. Any remaining slots in the population are filled with randomly-selected ϵ-dominance archive members that undergo uniform mutation applied with probability $1/L$. This seeding reintroduces previously-discovered non-dominated solutions into the search population but also introduces additional diversity through the mutation operator.

In challenging real-world applications, the small perturbations introduced by a mutation probability of $1/L$ may not be sufficient to escape the local optima. Small perturbations also do not help discover other disjoint Pareto optimal regions. However, simply increasing the mutation probability is not straightforward. Larger perturbations are disruptive, and can slow search by introducing many sub-optimal solutions into the population. We propose in this study an adaptive restart strategy that identifies the smallest mutation probability required to escape the local optima.

The Borg MOEA starts with a mutation probability of $1/L$ (Hadka and Reed, 2013). Whenever a restart occurs that fails to escape the local optima, the mutation probability is increased. When a restart is successful, the mutation probability is decreased. The speed at which the probabilities change is controlled by a parameter called the "mutation index", m_{index}. This index starts with value 0 and is incremented or decremented when restarts are unsuccessful or successful, respectively. A restart is unsuccessful if there are two back-to-back restarts with no changes to the ϵ-dominance archive (i.e., the ϵ-progress count remains unchanged). The "maximum mutation index", $m_{\max}$, defines the maximum value of m_{index}. The minimum value is 0. Then, the uniform mutation rate is calculated by

$$\text{mutation rate} = \left[1 + \frac{(L-1)\, m_{\text{index}}}{m_{\max}}\right] / L$$

where L is the number of decision variables defined by the MOP. Hence, when m_{index} is 0, the mutation rate is $1/L$; when m_{index} is equal to $m_{\max}$, the mutation rate is 100%.

2.3. Controllability of the Borg MOEA

We conclude this section by discussing briefly the results of Hadka and Reed (2013) and Hadka and Reed (2012). It is commonly known that MOEAs are often strongly sensitive to their parameterizations (Purshouse and Fleming, 2003, 2007). Most contemporary MOEAs are flawed in this respect since their performance is tied to non-trivial parameterizations that are not consistent across problem domains (or even problems within the same domain). Hadka and Reed (2012) developed a rigorous statistical framework for assessing the sensitivity of MOEAs to their parameterization. MOEAs with highly-sensitive parameters are termed "uncontrollable", as the decision-maker is required to constantly tweak parameters to improve performance. Controllability is a fundamental requirement for MOEAs to have operational value. Our studies have shown for a wide variety of problems that traditional non-adaptive MOEAs often suffer from isolated islands of effective parameters that would be very difficult if not impossible

to discover in a real-world application context (Hadka et al., 2012; Reed et al., 2013; Woodruff et al., 2012). Moreover, the transition to massively parallel computing systems requires users to confront machine access constraints, making it of paramount importance that an MOEA lack sensitivity to its parameterization.

Hadka and Reed (2012) used this sensitivity analysis framework to rigorously confirm that the auto-adaptive features of the Borg MOEA drastically improves the algorithm's controllability. Several of our studies using test functions and real-world applications have confirmed that while the Borg MOEA typically meets or exceeds other MOEAs in is efficiency, NFE is the key controlling parameter (Hadka and Reed, 2012; Hadka et al., 2012; Woodruff et al., 2012; Reed et al., 2013). Furthermore, since NFE is its key controlling parameter, it is expected to benefit substantially from parallelization.

While the focus of this paper is on the parallelization of the Borg MOEA, we would like to briefly discuss the impact of the recency and adaptive restart enhancements to the original Borg MOEA introduced in this section. These enhancements were designed to improve the efficiency and reliability of the Borg MOEA on severely-constrained problems like the risk-based water supply portfolio planning problem explored in this study. Our prior MOEA diagnostics studies on severely-constrained real-world problems (Hadka et al., 2012; Woodruff et al., 2012; Reed et al., 2013) has shown that the improved constraint handling techniques described this section can substantially improve the overall performance of the Borg MOEA. Additionally, the results presented in Section 6 exceed the comprehensive benchmarks for attainment, efficiency, and reliability observed in our prior work on the same problem (Reed et al., 2013).

3. Parallelizing the Borg MOEA

This section describes two parallel implementations of the Borg MOEA. Both implementations are designed to remain faithful to the adaptive nature of the serial Borg MOEA described in Section 2. The master-slave Borg MOEA implementation in Section 3.1 is designed to scale to thousands of processors. The multi-master implementation in Section 3.2 expands on the master-slave implementation to scale on emerging Petascale high-performance computing architectures.

3.1. Master-Slave Implementation

The master-slave model for MOEAs is a straightforward extension of a serial MOEA to perform objective function evaluations in parallel (Cantú-Paz, 2000; Coello Coello et al., 2007). Modern parallel systems are typically comprised of many multi-core processors, each consisting of two or more processing cores (e.g., a quad-core processor). Throughout this paper, we refer to these individual processing cores as "processors". As shown in Figure 2, on a system with P processors, one of the processors is labeled the "master" and the remaining $P - 1$ processors are labeled "slaves". Internally, the master node runs the serial MOEA as-is; the only alteration is that objective function evaluations are

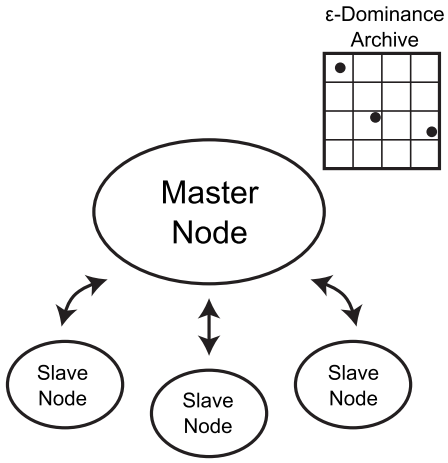

Figure 2: Diagram of the master-slave implementation of the Borg MOEA. The master node maintains the ϵ-dominance archive and runs the main loop of the serial Borg MOEA. The decision variables are transmitted to the slave nodes, and the evaluated objective function values and constraints are returned to the master node.

dispatched to one of the available slave nodes. The master sends the decision variable vector to an available slave node, the slave node evaluates the problem with the given decision variables, and when finished sends the evaluated objective values and constraints (if any) back to the master node.

Most MOEAs in use today are generational, meaning that the full population is replaced in a full loop through the selection, mating, and mutation operators. In a single generation, the population is evolved to produce offspring, the offspring are evaluated, and the offspring are added back into the population (possibly replacing existing members in the population). Parallelizing a generational MOEA using the master-slave approach is fairly straightforward (Cantú-Paz, 2000; Coello Coello et al., 2007). For the sake of simplicity, assume that the number of offspring is equal to the number of slave nodes, $P-1$. Then, when the algorithm reaches the point where it needs to evaluate the offspring, each member of the offspring is sent to its own slave node for evaluation. Once all slave nodes return the evaluated objective values, the algorithm resumes its serial loop. The need to completely evaluate all offspring before continuing to the next generation gives rise to the term "synchronous MOEA".

For completeness, we can remove our assumption that the number of offspring equals the number of slave nodes, $P-1$, by sending multiple offspring at a time to a single slave node. For instance, given 16 total processors, 15 would be slave nodes. For an offspring population size of 100, we can batch 6 or 7 offspring to be evaluated by a single slave node. When the offspring size does

not divide evenly by the number of slave nodes, then some nodes must process an additional offspring. As a result, some nodes have more work than the others, and will require more time to complete their evaluations. This potentially lowers efficiency as some of the slave nodes will sit idle while others continue processing. This problem also arises when the time to evaluate a solution is variable.

The Borg MOEA is a steady-state algorithm. Steady-state algorithms do not have defined generational boundaries; instead, each individual in the population evolves inside its own distinct evolutionary cycle. Since no boundary exists between generations, such algorithms are also called "asynchronous MOEAs". Additionally, the lack of a boundary often reduces overhead and increases the parallel performance of the algorithm. The remainder of this section describes the master-slave Borg MOEA implementation.

The master-slave Borg MOEA maintains a queue of unevaluated solutions. Whenever a slave node is available for processing, it queries this queue for the next solution to evaluate. If the queue is empty, then the typical Borg operator selection and offspring generation mechanism is triggered to insert one or more offspring (unevaluated solutions) into the queue. Otherwise, the next unevaluated solution in the queue is sent to the slave node. The main Borg MOEA loop is this process of slave nodes querying the queue for solutions, and new solutions being generated and appended to the queue as needed.

When a slave node finishes evaluating a solution and sends the evaluated objective and constraint values to the Borg master node, these solutions are immediately added to the population and ϵ-dominance archive. The strategy for adding/replacing solutions in the population and archive are identical to the serial Borg MOEA. These newly-added solutions are now available as parents when the offspring generation mechanism is invoked next. The flowchart of these steps is shown in Figure 3. The other components of the Borg MOEA, such as ϵ-progress restarts, adaptive population sizing, etc., occur next and are identical to the serial Borg MOEA. The only difference is that any new solutions generated during a restart are appended to the queue.

Initialization works similarly to the serial Borg MOEA with one exception. As with offspring generation, the solutions generated during initialization are added to the queue and processed as described earlier. However, consider what happens when running on $N + 1$ processors, with 1 master node and N slave nodes, and an initial population size of N. All N initial solutions will be generated randomly and sent to the slave nodes for evaluation. The first solution to finish evaluation is added to the population, and the next offspring is immediately generated. At this point, the population has only 1 evaluated solution, which is problematic for multi-parent recombination operators and also lacks sufficient genetic diversity. To ensure that the population is filled with a sufficient number of solutions before applying the evolutionary operators to generate offspring, the master-slave Borg MOEA always generates at least $2N$ initial solutions, where N is the number of slave nodes. This ensures that at least N solutions have been added to the population prior to applying any evolutionary operators.

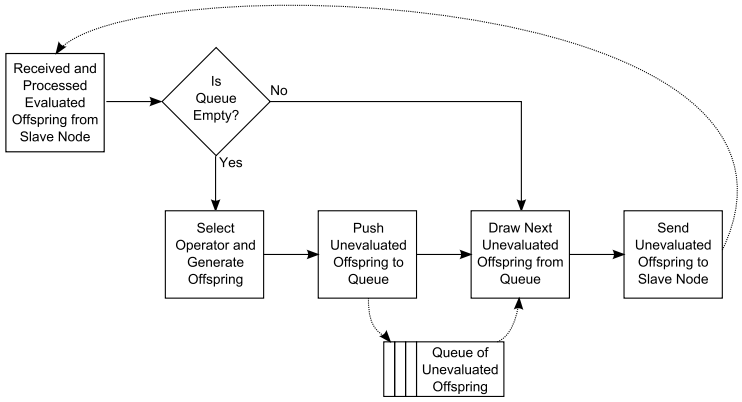

Figure 3: Flowchart of the main Borg MOEA loop running on the master nodes. A queue supports the asynchronous generation and evaluation of offspring. When a slave node is available (it returns an evaluated offspring), the master queries the queue for the unevaluated offspring. If the queue is empty, the algorithm invokes the operator selection and offspring generation steps from the serial Borg MOEA.

3.2. Multi-Master Implementation

The multi-master Borg MOEA hybridizes the master-slave implementation with the island-based model of parallelization representing a hierarchical parallelization scheme as defined by Cantú-Paz (2000). In an island-based model, each island runs a distinct MOEA with its own population evolved independently of other islands. Implementations of island-based MOEAs often include periodic migration events, wherein a small fraction of the population at each island is transmitted to one or more other islands. These migrations are intended to permit sharing of information between islands.

While the classic island-based model is a popular strategy for parallelizing MOEAs, it exacerbates the parameterization and algorithmic design challenges present in MOEAs. In order to run an island-based MOEA, one must select (1) the number of islands, (2) the number of processors per island, (3) the population size on each island, (4) operator selection and parameterization, (5) whether to run the same MOEA (homogeneous) or different MOEAs (heterogeneous) on each island, (6) migration policies; etc. Cantú-Paz (2000) developed theoretical models to determine problem-specific values for some of these settings, but in doing so also shows the complexities and non-linear relationships between the various settings that makes parameterization challenging. The effectiveness of an island-based MOEA is heavily dependent on such non-trivial design choices that must be tailored to individual problems. This is a limiting factor in the operational value of classic island-based MOEAs.

Our design of the multi-master Borg MOEA seeks to generalize its ease-of-

use and auto-adaptivity while maximizing its parallel efficiency on large-scale computing architectures. Several studies have shown that the Borg MOEA's auto-adaptivity eliminates parameterization concerns by allowing the algorithm to adapt and maximize its potential on a given problem (Hadka and Reed, 2012; Hadka et al., 2012). This eliminates issues (3), (4), and (5), since the dynamics of the Borg MOEA automatically configure the algorithm for the local conditions encountered during search. This additionally implies that each island running the Borg MOEA can assume drastically different search operators as needed to maximize performance. For instance, a struggling island can introduce heavy mutations to escape local optima while another island is fine-tuning near-optimal solutions using small, local perturbations. We address (6) by introducing an auto-adaptive migration mechanism based on the search progress made by each island. Unlike the unguided migration events in classic island-based models, in the multi-master Borg MOEA migrations only occur when an island is struggling and injects high-quality solutions and new search operator preferences to guide the struggling local population. Lastly, we answer (1) and (2) in Hadka et al. (2013) by contributing a discrete event simulation model to predict topologies for the multi-master Borg MOEA that maximize its parallel efficiency. The full details of the multi-master Borg MOEA implementation are given below.

As shown in Figure 4, the multi-master Borg MOEA introduces a new node, called the "controller", that has two responsibilities: (1) maintaining a global ϵ-dominance archive, and (2) providing guidance to master nodes when they need help. The global ϵ-dominance archive maintains the Pareto optimal solutions discovered by all master nodes. Identical to how each master node uses the ϵ-dominance archive to track the operators that contribute new, Pareto approximate solutions, the controller uses the global ϵ-dominance archive to track the operators that contribute globally Pareto approximate solutions. Note the term "global" as used here refers to the aggregate of all ϵ-dominant solutions from the full suite of searching master nodes. Each master node periodically sends an update to the controller. This update contains any new Pareto approximate solutions discovered by the master since its last update. In this study, this update is sent every 10000 NFE. This is small enough to ensure the global ϵ-dominance archive is updated frequently but large enough not to overwhelm the controller node (Hadka, 2013).

Since each master node is running an instance of the master-slave Borg MOEA, it includes all of the mechanisms to detect search stagnation and trigger restarts. In the event that these mechanisms are unsuccessful at escaping the local optima, the master node notifies the controller that it needs assistance. Upon receiving the help request, the controller seeds the master with the contents of the global ϵ-dominance archive and global operator probabilities. This in essence replaces the local ϵ-dominance archive that was stuck at a local optima with the global search state. Additionally, it provides the best-known global operator probabilities for contributing new Pareto approximate solutions. Upon receiving this guidance from the controller, the master updates its internal state and triggers a restart. Since the local archive of the master node is now set

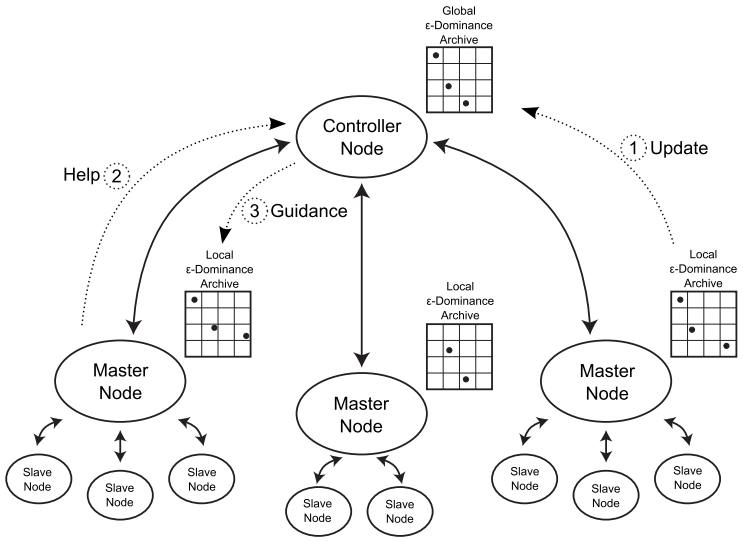

Figure 4: Diagram of the multi-master implementation of the Borg MOEA. The multi-master Borg MOEA consists of two or more master-slave instances. This diagram depicts three such instances. The multi-master consists of an additional controller node, which communicates with the masters using several messages. (1) Each master node periodically transmits its local ϵ-dominance archive to the controller to update the global archive. (2) When a master node is struggling, it sends a help message to the controller. (3) The controller responds with guidance, which includes the global ϵ-dominance archive and global operator probabilities.

to the global ϵ-dominance archive, the solutions injected during the restart are derived from the global search state, and the adaptive population sizing ensures the population is resized appropriately given the global search state.

The multi-master implementation also features a different style of initialization from the serial and master-slave Borg MOEA implementations. The original Borg MOEA generated the initial population by sampling the decision variables uniformly at random from their bounds. While uniform sampling is a common initialization strategy used in MOEAs, it has a major disadvantage: it makes no guarantee that the sampled points are representative of the actual distribution. In the context of MOEAs, this means that there is no guarantee that the initial population includes a representative sampling of all decision variable combinations. Instead, uniform sampling tends to result in areas with higher and lower densities, potentially introducing random bias into the initial search population.

It has been proposed in the literature to use other sampling techniques like Latin hypercube sampling (LHS) and Sobol's low-discrepancy sequence generator (Bäck et al., 1997). The improved quality of the samples by LHS and Sobol' sequence have been used in Monte Carlo simulations to improve convergence and reduce the number of required samples (Macdonald, 2009). In the context of MOEAs, LHS and Sobol' sequence help ensure that the initial population contains a representative sampling of the search space. In the multi-master Borg MOEA, we propose the global Latin hypercube initialization strategy. When the multi-master algorithm first starts, each master node notifies the controller of its desired initial population size. The sum total is the number of initial solutions generated by the controller using LHS. The controller then uniformly at random partitions these solutions into the initial populations for each master. Finally, the controller transmits the initial populations to the master.

For example, suppose we have 16 islands each using an initial population size of 100. Just like the master-slave Borg MOEA, the master node generates twice as many initial solutions as required to ensure that the population is filled prior to entering the main evolutionary loop. Thus, each island would request 200 initial solutions. Then, the controller would generate $16 * 200 = 3200$ initial solutions using LHS. Next, these 3200 solutions will then be randomly partitioned into 16 groups of 200. Finally, each group is sent to the corresponding island.

While this initialization strategy adds some additional overhead at startup, it has the benefit of ensuring that globally, the multi-master algorithm starts with a well-distributed, diverse set of initial solutions. Without this approach, the initial populations would have less diversity and likely subject to faster preconvergence.

4. Complex Environmental System: Water Supply Portfolio Planning

This section introduces a challenging risk-based urban water portfolio planning application to test the effectiveness, efficiency, and reliability of the parallel variants of the Borg MOEA. Urban water supply management is the act

of securing and allocating water resources to a locale under varying environmental and economic conditions. Population growth, increased urbanization, water scarcity due to droughts, and climate change are factors that challenge water supply management and increase the risk of critical water supply failures (Kundzewicz et al., 2007; Frederick and Schwarz, 1999; Lane et al., 1999; Vorosmarty et al., 2000; Milly et al., 2008; Brekke et al., 2009). A number of approaches can be taken to facilitate increases in demand and mitigate the impact of supply fluctuations. The municipality can undertake structural improvements, such as building new reservoirs, and non-structural adaptations, such as purchasing water on water markets (Anderson and Hill, 1997). Water markets aim to allocate water resources to their highest-value use by transferring volumes of water across regions or user sectors (Israel and Lund, 1995; Hadjigeorgalis, 2008).

In this case study, water supplies can be purchased using three market mechanisms: permanent rights, leases, and options. Permanent rights represent the purchase of a fixed percentage of the stream inflows to a reservoir. Leases facilitate short-term transfers of water from agricultural users to a city, but prices fluctuate with supply and demand. For instance, the onset of drought conditions can lead to a spike in prices. Alternatively, options reserve volumes of water at a fixed price that can be transferred later in the year. Options that remain unused at the end of the year are dropped, and can become costly if the city holds many unused options at the end of the year.

Several studies considering only single-objective formulations of this problem have shown that water markets with both options and leases can reduce the overall cost associated with maintaining reliable urban water supplies (Lund, 1995; Wilchfort and Lund, 1997; Watkins Jr. and McKinney, 1999; Jenkins and Lund, 2000; Characklis et al., 2006; Kirsch et al., 2009). Kasprzyk et al. (2009) proposed the first many-objective formulation of this problem, allowing tradeoffs between cost, reliability, surplus water, cost variability, frequency of using leases, and unused transfers of water. They applied this problem to a city located in the Lower Rio Grande Valley (LRGV) in southern Texas with a 10-year planning horizon. A Monte Carlo simulation models the city using both thirty-three years of historical data from the region with additional factors like growing population demands, variable hydrologic conditions, and market pricing dynamics. In this study, we use the most challenging "Case D" variant of the problem from Kasprzyk et al. (2009) and refer to it as the LRGV problem.

The LRGV problem consists of 8 decision variables, 6 objectives, and 3 constraints. The 8 decision variables shown in Table 1 control the use of permanent rights, options, and leases by the simulation model. Several of these decision variables are discrete. Since the Borg MOEA uses real-valued operators, the decision variables are rounded to the nearest integer prior to invoking the simulation model. The simulation model outputs the 6 objectives shown in Table 2. The LRGV problem is thus defined by

$$\begin{aligned} F(x) =& (f_{\text{cost}}(x), f_{\text{rel}}(x), f_{\text{surplus}}(x), f_{\text{costvar}}(x), \\ & f_{\text{dropped}}(x), f_{\text{leases}}(x)) \end{aligned} \tag{1}$$

Table 1: Decision variables used by the LRGV problem.

Decision Variable	Type	Range	Description
N_R	Integer	30,000-60,000	Volume of permanent rights
$N_{O_{\text{low}}}$	Integer	0-20,000	Low-volume options contracts
$N_{O_{\text{high}}}$	Real	$N_{O_{\text{low}}} - 2N_{O_{\text{low}}}$	High-volume options contracts
ξ	Real	0.1-0.4	Low to high options threshold
$\alpha_{\text{May-Dec}}$	Real	0.0-3.0	Lease/options strategy for May-Dec ("when to acquire")
$\beta_{\text{May-Dec}}$	Real	$\alpha_{\text{May-Dec}}$-3.0	Lease/options strategy for May-Dec ("how much to acquire")
$\alpha_{\text{Jan-Apr}}$	Real	0.0-3.0	Lease/options strategy for Jan-Apr ("when to acquire")
$\beta_{\text{Jan-Apr}}$	Real	$\alpha_{\text{Jan-Apr}}$-3.0	Lease/options strategy for Jan-Apr ("how much to acquire")

Table 2: Objectives used by the LRGV problem.

Objective	Description	Direction	ϵ Search Precision
f_{cost}	Cost	Min	0.003
f_{rel}	Reliability	Max	0.002
f_{surplus}	Surplus	Min	0.01
f_{costvar}	Cost Variability	Min	0.001
f_{dropped}	Dropped Transfers	Min	0.002
f_{leases}	Number of Leases	Min	0.003

where

$$x = (N_R, N_{O_{\text{low}}}, N_{O_{\text{high}}}, \xi, \alpha_{\text{May-Dec}}, \beta_{\text{May-Dec}}, \alpha_{\text{Jan-Apr}}, \beta_{\text{Jan-Apr}}). \quad (2)$$

The 3 constraints ensure that potential solutions satisfy limits in cost variability, reliability, and critical reliability. Reliability measures small failures that can be mitigated by water conservation or other practices. Critical reliability measures larger failures where the city fails to meet more than 60% of the required demand in a given month. Formally, these constraints are defined by

$$f_{\text{costvar}} < 1.1 \quad (3)$$

$$f_{\text{rel}} > 0.98 \quad (4)$$

$$Pr[S_{i,j} > 0.6 d_{i,j}] = 1.0, \ \forall i \in [1, 12] \text{ and } j \in [1, T] \quad (5)$$

where $S_{i,j}$ is the simulated supply and $d_{i,j}$ is the simulated demand for month i in the year j, and $T = 10$ is the number of simulated years. Full details of the LRGV problem are available in Kasprzyk et al. (2009, 2012).

Since the LRGV simulation is stochastic, many Monte Carlo trials are performed when computing the expected values for its performance objectives. Increasing the number of Monte Carlo trials will improve the quality the estimates of the expected values for the objectives, but also significantly increases the evaluation time. In this study, 1000 samples are used, resulting in an evaluation time of approximately 0.14 seconds. This evaluation time is based on runs performed on the TACC Ranger supercomputer that operates 2.3 GHz AMD Opteron Quad-Core 64-bit processors (see Section 5 for details).

The first attempts to solve the LRGV problem used the ϵ-NSGA-II to discover the tradeoffs between various market strategies and their impact on cost and reliability when faced with the uncertainty and risks inherent in water portfolio planning (Kasprzyk et al., 2009). Reed et al. (2013) performed a rigorous assessment of several MOEAs on the LRGV problem, identifying that all of the top serial MOEAs struggled with their attainments and controllability, many of which completely failed on this problem.

These search failures are the result of several problem characteristics. First, the LRGV problem is a many-objective problem with a fully stochastic objective space. Many MOEAs are unable to cope with problems with four or more deterministic objectives as they are unable to effectively navigate and search high-dimensional spaces (Purshouse and Fleming, 2003, 2007; Hadka and Reed, 2012). Second, the problem is severely constrained. Reed et al. (2013) showed a random sampling baseline where the probability of randomly generating a feasible solution for the LRGV problem is approximately 1 in 500000. This implies the initial population will likely consist entirely of infeasible solutions, requiring the MOEA to direct search towards feasible regions. MOEAs unable to do so will fail to generate any Pareto approximate solutions. Third, as identified in Kasprzyk et al. (2009), the best-known reference set consists of three disjoint regions corresponding to vastly different water planning strategies. A successful

Table 3: The parallel MOEAs tested in this study and their salient characteristics.

Implementation	Islands	Initialization	Style	Operator
Master-Slave ϵ-NSGA-II	1	Uniform	Generational	SBX+PM
Master-Slave Borg	1	Latin	Steady-State	Multi-operator
Multi-Master Borg	2	Global Latin	Steady-State	Multi-operator
Multi-Master Borg	4	Global Latin	Steady-State	Multi-operator
Multi-Master Borg	8	Global Latin	Steady-State	Multi-operator
Multi-Master Borg	16	Global Latin	Steady-State	Multi-operator
Multi-Master Borg	32	Global Latin	Steady-State	Multi-operator

MOEA must be able to locate and diversify across all disjoint regions within the best known Pareto approximate set. Finally, the LRGV problem has an expensive function evaluation time. As mentioned previously, the objective function evaluation time in this study is approximately 0.14 seconds. This necessitates the use of parallel MOEAs in order to discover high-quality solutions in a minimum amount of wallclock time while being a sufficiently small evaluation time as to make parallel scalability challenging. Our parallel scalability as analyzed in this paper is conservative as many real-world applications would have substantially higher evaluation times making parallel scalability much easier to attain (Hadka et al., 2013).

5. Methodology

This study compared the master-slave and multi-master Borg MOEA implementations against the ϵ-NSGA-II algorithm originally used to explore the LRGV problem. ϵ-NSGA-II is one of the top-performing MOEAs on the LRGV problem (Reed et al., 2013). In this study, we are using the large-cluster master-slave ϵ-NSGA-II implementation from Reed et al. (2008). The master-slave and multi-master Borg MOEA implementations were written in high-performance C with the use of MPI to facilitate communication between nodes. This code was compiled and executed on the Texas Advanced Computing Center (TACC) Ranger system. TACC Ranger consists of 3,936 16-way symmetric multiprocessing (SMP) compute nodes, each containing four 2.3 GHz AMD Opteron Quad-Core 64-bit processors and 32 GBs of memory. Each core can perform 9.2 GFLOPS. In total, there are 62976 processing cores. Recall that in this paper we refer to these individual processing cores as "processors". Nodes are connected using two large Sun InfiniBand DataCenter switches.

The master-slave and multi-master Borg MOEA implementations were executed in a number of different configurations to compare their scalability and solution quality at large processor counts. On TACC Ranger, submissions are limited to 16384 cores. Therefore, the three implementations were each executed with 1024, 2048, 4096, 8192, and 16384 cores. Additionally, the multi-master

runs used different topologies with 2, 4, 8, 16 and 32 islands. A single run of an implementation was given 10 minutes of wallclock time, and allowed to evaluate as many objective function evaluations as it could manage. Each run was repeated 50 times with different initial random seeds so that the expected search quality and its deviation can be calculated. A summary of the algorithms tested in this study are given in Table 3.

The output of each run is the approximation set generated by the algorithm. This approximation set is stored in a database. After all runs have been executed, the aggregation of all approximation sets across all algorithms forms the reference set. This reference set contains all Pareto approximate solutions discovered in this study. Using this reference set, we can subsequently compute various performance indicators. Based our prior comprehensive assessment of the LRGV test case for a broad suite of MOEAs (Reed et al., 2013), we have selected to emphasize the hypervolume indicator. Our prior results have shown that the hypervolume is sensitive to the irregular Pareto approximate set geometry of the LRGV test case and that, in general, other measures are equivalent or easier to satisfy at high levels of performance. Hypervolume measures the volume of objective space dominated by an approximation set. Larger hypervolumes therefore correspond to approximation sets that dominate more space, which in turn indicates high-quality approximation sets.

Figure 5 shows an example of how hypervolume is computed in 2D space. A reference point is chosen based on the bounds of the reference set plus some additional delta. This delta ensures the boundary points contribute positive volume to the overall hypervolume. Hypervolume is normalized to the range $[0, 1]$ such that the best possible set, the reference set, has a hypervolume of 1. Approximation sets with hypervolumes near 1 are high-quality, have converged in proximity to the reference set, and are diversified across the entire Pareto front.

While hypervolume can be expensive to calculate, it offers several advantages over other performance indicators. Its results are scaling independent, it is compatible with the dominance relation, and its meaning is intuitive (Zitzler et al., 2002). Since the LRGV problem has six objectives, we elected to use the efficient WFG hypervolume algorithm to calculate exact hypervolume values (While et al., 2012).

In addition to recording the end-of-run approximation set, runtime data is collected every 10,000 NFE and stored in the database. The data includes a snapshot of the approximation set discovered by the algorithm at the current point in time, the operator probabilities used by the Borg MOEA's adaptive multi-operator mechanism, and local and global restart frequencies. Identical to how we compute hypervolume for the end-of-run approximation set, we also compute hypervolume for each snapshot. This provides a view into the dynamics of the algorithm. We can visualize the inner workings of the parallel Borg MOEA and its impact on solution quality.

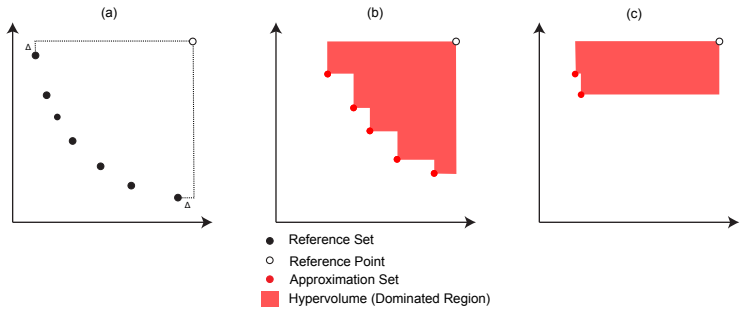

Figure 5: 2D demonstration of the hypervolume indicator. (a) The bounds of the reference set are used to calculate the reference point; this calculation typically adds a delta so that the boundary points contribute positive hypervolume. (b) Given an approximation set, the hypervolume is the volume of space dominated between the approximation set points and the reference point. (c) Demonstration of how an approximation set with good proximity but poor diversity results in a sub-optimal hypervolume.

6. Results

The LRGV problem described in Section 4 was solved using the large-cluster master-slave ϵ-NSGA-II (see Reed et al. (2008)), the master-slave Borg MOEA, and several configurations of the multi-master Borg MOEA as described in Section 5. This section presents the results from this experiment. First, Section 6.1 investigates the time required to converge to high-quality solutions, identifying the implementations which converged fastest and with the highest reliability. Second, we explore the end-of-run solution quality as a result of running each implementation for a fixed amount of time in Section 6.2, identifying the implementation that produced the highest-quality result. In Section 6.3, we analyze the operator dynamics introduced by the auto-adaptive multi-operator search mechanism used by the Borg MOEA. Finally, Section 6.4 calculates the parallel efficiency and speedup of the implementations, identifying the configurations that maximize their use of the underlying computing resources.

6.1. Convergence Speed and Reliability

Figure 6 shows the speed and reliability of the different parallel MOEA implementations tested in this study. These results show the cumulative distribution functions (CDFs) for generating high-quality approximation sets with respect to wallclock time. Here, an algorithm generates a high-quality approximation set if its hypervolume is $\geq 90\%$ of the best-known, reference set hypervolume. Each of the subplots in Figure 6 shows the results for different processor counts. Each of the line series corresponds to one of the implementations in Table 3.

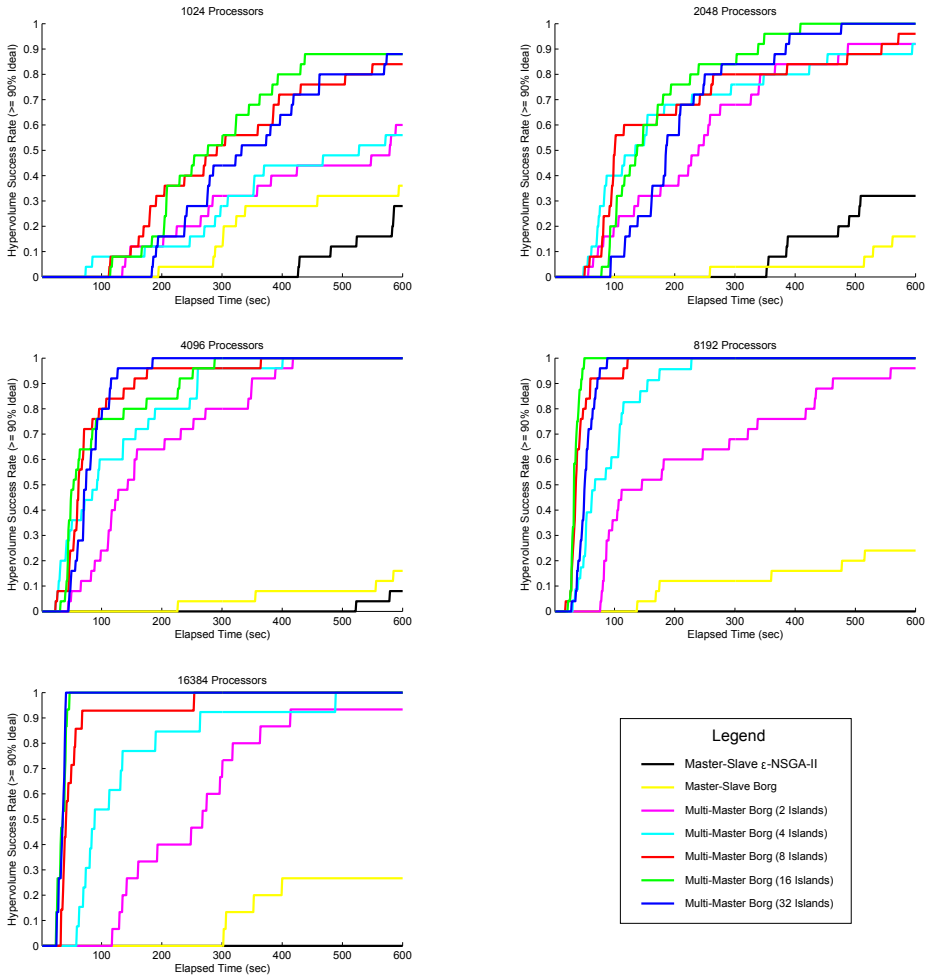

Figure 6: Probability of each parallel implementation of attaining a hypervolume >= 90% of the reference set hypervolume on the LRGV problem. Each subplot shows the results for different processor counts, from 1024 up to 16384 processors.

These line series plot at each point in time the probability that the implementation generated high-quality approximation sets exceeding the 90% hypervolume threshold. Ideal performance on these plots are vertical CDFs (i.e., no random seed variability) at a minimum wallclock.

Starting with the 1024 processor subplot, we observe that none of the implementations had a 100% probability of attaining the 90% hypervolume threshold within the wallclock allowed. The closest results were provided by the 16 and 32 island multi-master Borg MOEA implementations, which reached the hypervolume threshold with 90% probability. This is followed closely by the 8 island multi-master Borg MOEA implementation with 85% probability, and more distantly by the 2 and 4 island multi-master Borg MOEA implementations with 60% and 55% probability, respectively. The high failure rates for several configurations of the parallel Borg MOEA confirm the difficulty of the LRGV case study as has been observed in prior work (Reed et al., 2013). All of the multi-master Borg MOEA implementations significantly exceeded the reliability of the master-slave Borg MOEA and ϵ-NSGA-II implementations. Note that the slopes of all of the success rate CDFs show strong random seed variability in the time required to attain high-quality approximations of the LRGV case study's tradeoffs.

Additionally, by observing the position along the x-axis where the line series reached their maximum, we can determine the convergence speed of the algorithm. Continuing with our analysis, we observe in the 1024 processor subplot that the 16 and 32 island multi-master Borg MOEA implementations converged in 450 and 560 seconds, respectively. In general, we desire MOEAs that produce the highest-quality results. As with this case, when the quality attained by two different implementations are equivalent, we then look at the speed of convergence. For the 1024 processor case, the 16 island multi-master Borg MOEA implementation produced the best result.

As the processor count increases, we observe that many implementations are able to reach the 90% hypervolume threshold with 100% probability. With 2048 processors, the 16 island multi-master Borg MOEA implementation converged fastest with 100% probability in 410 seconds. With 4096 processors, the 32 island multi-master Borg MOEA implementation dominates, converging with 100% probability in 190 seconds. With 8192 processors, the 16 and 32 island multi-master Borg MOEA implementations perform similarly, converging with 100% probability in 50 and 80 seconds, respectively. Finally, at 16384 processors, the 16 and 32 island multi-master Borg MOEA implementations have nearly identical convergence speeds of approximately 40 seconds. Note at 8192 and 16384 processor counts, the top performing instances of the multi-master Borg MOEA have virtually no random variability. Any given trial of the algorithm is 100% reliable in both solution quality and wall clock time required. This a major benefit for operational use of the algorithm on large parallel architectures where compute hours are often strongly constrained.

From these results, it is clear that the multi-master implementations provide significant improvements in terms of speed and reliability over the master-slave implementations. The master-slave Borg MOEA and ϵ-NSGA-II implementa-

tions never converged with 100% probability, regardless of how many processors were available. This failure is attributed to the inefficiency of the master-slave implementations, which quickly become congested trying to receive messages from so many slave nodes (Hadka et al., 2013). Furthermore, the ability of struggling islands to request help from the controller node also is a contributor to the superior performance of the multi-master implementations.

At higher processor counts, inefficiencies due to congestion can also be seen in the 2 and 4 island multi-master implementations. For instance, compare the 2 island multi-master Borg MOEA for the 4096, 8192, and 16384 processor subplots. With 4096 processors, the 2 island multi-master Borg MOEA implementation is performing reasonably well. However, its performance declines significantly with 8192 and 16384 processors. This is a result of each island becoming congested, and it is simply unable to evaluate as many NFE as the implementations with more islands. This shows that selecting a topology appropriate for the processor count is critical. Our discrete simulation-based approach for determining the optimal topology for the multi-master Borg MOEA will be discussed later in Section 6.4.

6.2. End-of-Run Quality

In the previous section, we analyzed the results in terms of the 90% hypervolume threshold. We fixed the performance threshold and observed the time required to reach this threshold. In this section, we instead fix time and look at the performance of each implementation. As described in Section 5, each implementation was run for 10 minutes. The end-of-run hypervolume is calculated from the approximation set produced by each MOEA after 10 minutes.

Table 4 shows the median and standard deviation of the end-of-run hypervolume from all 50 seeds for each implementation. Recall that a hypervolume of 1 is optimal. At 1024 processors, the multi-master Borg MOEA improvement is marginal. The hypervolume increases approximately 2% when switching from the master-slave ϵ-NSGA-II to the 32 island multi-master Borg MOEA. At larger processor counts, the improvement is more significant. With 16384 processors, the 32 island multi-master Borg implementation produces a hypervolume 29% better than master-slave ϵ-NSGA-II. This implies a significant improvement in solution quality when switching to the multi-master Borg MOEA implementation.

Across all topologies, the 16384 processor runs of 32 island multi-master Borg MOEA resulted in the best end-of-run hypervolume. Combined with the speed and reliability results from Section 6.1, this shows concretely that the multi-master Borg MOEA with a larger number of islands produces the highest-quality results efficiently and reliably. Furthermore, the results significantly exceed the quality of the master-slave ϵ-NSGA-II and Borg MOEA implementations.

Table 4 also provides results from the Kruskal-Wallis and Mann-Whitney U tests. Both tests determine whether differences in the medians of two sampled populations are statistically significant or occurred due to random chance (Sheskin, 2004). The Kruskal-Wallis test is first applied to all medians in the table to determine if there is a statistical difference in the entire table. Since the

Table 4: Table showing the median and standard deviation of the end-of-run hypervolume results. The Kruskal-Wallis and Mann-Whitney U tests were used to test the statistical significance of the medians. The significant column contains a ✓ if the median from that row is significantly different than the best result, 16384 processor multi-master Borg MOEA (32 islands), with 95% confidence. The row containing the best result is highlighted. The final column contains the corresponding p-value from the Mann-Whitney U test.

Processors	Implementation	Median	Stdev	Significant	p-value
1024	Master-Slave ϵ-NSGA-II	0.88889	0.013124	✓	1.75×10^{-7}
	Master-Slave Borg	0.89146	0.015297	✓	1.75×10^{-7}
	Multi-Master Borg (2 Islands)	0.89892	0.015105	✓	1.75×10^{-7}
	Multi-Master Borg (4 Islands)	0.89512	0.010933	✓	1.75×10^{-7}
	Multi-Master Borg (8 Islands)	0.90447	0.015395	✓	5.71×10^{-7}
	Multi-Master Borg (16 Islands)	0.90786	0.011394	✓	1.75×10^{-7}
	Multi-Master Borg (32 Islands)	0.90796	0.012429	✓	2.03×10^{-7}
2048	Master-Slave ϵ-NSGA-II	0.89667	0.013536	✓	1.75×10^{-7}
	Master-Slave Borg	0.88374	0.013262	✓	1.75×10^{-7}
	Multi-Master Borg (2 Islands)	0.90897	0.014425	✓	3.18×10^{-7}
	Multi-Master Borg (4 Islands)	0.91225	0.013274	✓	3.18×10^{-7}
	Multi-Master Borg (8 Islands)	0.91526	0.014061	✓	2.74×10^{-7}
	Multi-Master Borg (16 Islands)	0.92074	0.015761	✓	3.08×10^{-6}
	Multi-Master Borg (32 Islands)	0.91621	0.012114	✓	2.36×10^{-7}
4096	Master-Slave ϵ-NSGA-II	0.87477	0.014715	✓	1.75×10^{-7}
	Master-Slave Borg	0.88124	0.013009	✓	1.75×10^{-7}
	Multi-Master Borg (2 Islands)	0.92561	0.012299	✓	2.36×10^{-7}
	Multi-Master Borg (4 Islands)	0.92572	0.015114	✓	5.27×10^{-6}
	Multi-Master Borg (8 Islands)	0.92695	0.013407	✓	7.82×10^{-6}
	Multi-Master Borg (16 Islands)	0.92601	0.015314	✓	1.49×10^{-5}
	Multi-Master Borg (32 Islands)	0.9332	0.013837	✓	4.01×10^{-5}
8192	Master-Slave ϵ-NSGA-II	0.8163	0.014652	✓	3.61×10^{-7}
	Master-Slave Borg	0.88813	0.015637	✓	1.75×10^{-7}
	Multi-Master Borg (2 Islands)	0.91815	0.015299	✓	5.71×10^{-7}
	Multi-Master Borg (4 Islands)	0.93421	0.011551	✓	0.000149
	Multi-Master Borg (8 Islands)	0.93698	0.016602	✓	0.010163
	Multi-Master Borg (16 Islands)	0.94167	0.010124	✓	0.005836
	Multi-Master Borg (32 Islands)	0.94194	0.012687	✓	0.025419
16384	Master-Slave ϵ-NSGA-II	0.73672	0.14131	✓	3.39×10^{-6}
	Master-Slave Borg	0.8907	0.017862	✓	3.39×10^{-6}
	Multi-Master Borg (2 Islands)	0.91252	0.014744	✓	5.05×10^{-6}
	Multi-Master Borg (4 Islands)	0.92989	0.01303	✓	0.000464
	Multi-Master Borg (8 Islands)	0.94489	0.01707		0.21356
	Multi-Master Borg (16 Islands)	0.94534	0.013617		0.53383
	Multi-Master Borg (32 Islands)	0.94814	0.014137		

Kruskal-Wallis test indicated differences were significant, the Mann-Whitney U test is applied to each pair to determine which specific cases are significant. Since the 32 island multi-master Borg MOEA implementation produced the

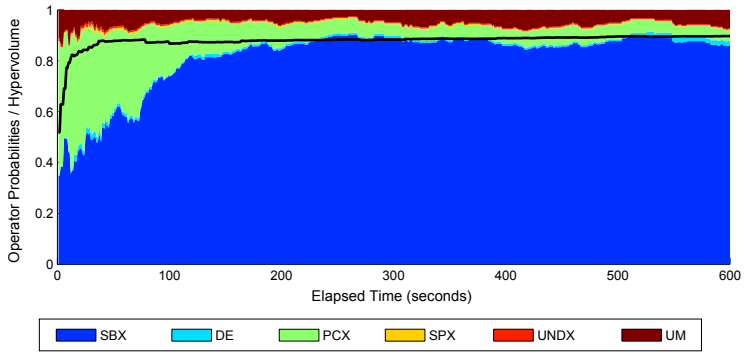

Figure 7: The operator probability runtime dynamics from a single run of the master-slave Borg MOEA with 1024 processors. The solid black line traces the hypervolume of the approximation set at each point in time.

best end-of-run hypervolume, we compare the significance of this result with all other topologies. In Table 4, the "significant" column contains a check mark if the end-of-run hypervolume from that row was statistically different from the 32 island multi-master Borg MOEA result with 95% confidence. Additionally, the p-value from the Mann-Whitney U test is shown. With 95% confidence, a p-value ≤ 0.05 rejects the null hypothesis and implies that the results are statistically significant.

These statistical tests show that there is not a statistically significant difference between the 8, 16, and 32 island multi-master Borg MOEA implementations with 16384 processors. However, these three implementations are statistically better than all other runs.

6.3. Operator Dynamics

The Borg MOEA bases its selection of search operators on archive membership and recency as discussed in Section 2. Adapting its search operators at runtime allows the Borg MOEA to favor operators that contribute more Pareto approximate solutions, leading to faster convergence and diversification. In this section, we explore the operator dynamics on the LRGV problem. The results in this section are based on a single, typical run. We have confirmed that the trends observed in these results are consistent with general trends.

Figure 7 shows the operator probabilities from a single run of the master-slave Borg MOEA on the LRGV problem with 1024 processors. At each point in time along the x-axis, this plot shows the combination of search operators using the shaded regions. Large shaded regions corresponding to heavier use of that operator. Additionally, the black solid line traces the hypervolume of the approximation set at each point in time. Although it would be expected that the specific operator probabilities and search dynamics will vary, we have

found that they are generally consistent making these results reflective of typical search behavior. The run shown in Figure 7 begins with significant use of simulated binary crossover (SBX), parent-centric crossover (PCX), and uniform mutation with probability $1/L$ (UM). These four operators facilitate rapid identification and convergence to the Pareto approximate front. SBX takes over in diversifying along the Pareto front, since SBX with a large distribution index (as with prior studies, this study uses a distribution index of 15) introduces only small perturbations resulting in small, local improvements. Also note that there is no single activated operator, but instead there exists cooperation between several search operators. This cooperation allows the Borg MOEA to combine the qualities of multiple search operators when generating offspring, and can significantly improve the quality of search (Vrugt and Robinson, 2007; Vrugt et al., 2009).

As demonstrated in this example, the use of multiple search operators significantly improves the search dynamics of an MOEA. Membership and recency allow the MOEA to quickly identify the search operators that are beneficial. We also observe that two operators, differential evolution (DE) and unimodal normal distribution crossover (UNDX), had minimal use. While DE and UNDX were not used heavily on the LRGV problem, they have been actively used on other problems (Hadka et al., 2012). Allowing the MOEA to determine the appropriate selection of search operators is a significant advantage when using the Borg MOEA for real-world complex engineered systems applications.

We can also explore the operator dynamics on the multi-master Borg MOEA. Recall that each island maintains its own operator probabilities, but they can request help from the controller. When receiving help, the island also receives updated operator probabilities that are derived from the global ϵ-dominance archive. Figure 8 shows the operator dynamics for a single run of the 16 island multi-master Borg MOEA with 1024 processors. Each of the subplots shows the operator probabilities from a single island. The vertical black lines indicate when the island requests help from the controller. Like Figure 7, the solid black line traces the hypervolume of the approximation set at each point in time.

Many islands, as expected, only require help at the end of the run once the initial convergence and diversification is complete. However, we observe that several islands benefit from receiving help earlier in runs. For instance, Island 12 started with significant use of uniform mutation (UM). This selection of operator probabilities was ineffective; the algorithm quickly determined that it was no longer making improvements and immediately asked the controller for help. Upon receiving help, as indicated by the left-most vertical black line, the guidance provided by the controller corrected the operator probabilities to allow search to progress. Thereafter, the algorithm made continuous progress as indicated by the lack of additional help messages until much later in the run. Other islands, such as Island 15, do not require any help during a run.

This example demonstrates how the Borg MOEA can avoid bad initial seeds by relying on the global knowledge gained by running multiple concurrent instances of the Borg MOEA. As we saw with Island 12 in Figure 8, an initial bad seed can be quickly detected and corrected without wasting significant com-

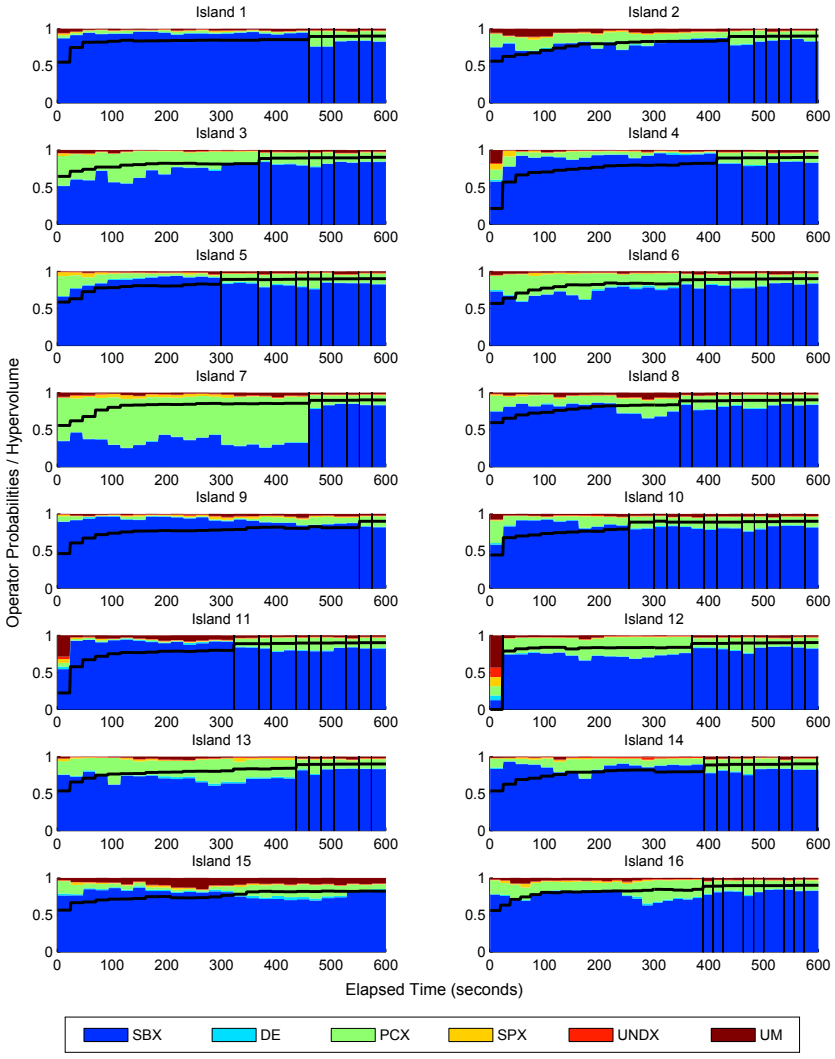

Figure 8: The operator probability runtime dynamics from a single run of the 16 island multi-master Borg MOEA with 1024 processors. Each subplot shows the operator probabilities for an island. The vertical black lines indicate when the island requested help from the controller. Like Figure 7, the solid black line traces the hypervolume of the approximation set at each point in time.

puting resources. This contributes to the overall improvement in search quality observed when running the multi-master Borg MOEA with many islands.

We lastly turn to look at the improvement in search quality resulting from the island-based model in the multi-master Borg MOEA. Both Figure 7 and Figure 8 plot the hypervolume of the approximation set at each point in time with solid black lines. Recall that hypervolumes near 1 indicate high-quality results. The master-slave Borg MOEA search dynamics in Figure 7 show that the hypervolume quickly levels off around 0.85 and makes no further improvements. The master-slave Borg MOEA is simply unable to attain high-quality results. However, by running multiple islands and sharing solutions and operator probabilities between islands as done in the multi-master Borg MOEA, hypervolume is increased substantially. Figure 8 shows this effect. While individual islands tend to converge slower than the master-slave run in Figure 7, they attain substantially better hypervolume results later in the run.

6.4. Parallel Efficiency and Speedup

Finally, we explore the parallel efficiency and speedup of the various master-slave and multi-master Borg MOEA configurations explored in this study. Since each implementation was run for a fixed wallclock time (10 minutes), efficiency is based on the total NFE in each run. Thus, if NFE_S is the total NFE for a serial run and NFE_P is the total NFE for a parallel run with P processors, efficiency is calculated by

$$\text{efficiency} = \frac{\text{NFE}_P}{P \cdot \text{NFE}_S}. \tag{6}$$

The total NFE of the serial algorithm running for 10 minutes is $\text{NFE}_S = 4285$. Table 5 shows the total NFE expended by each parallel implementation and the calculated efficiency.

With only 1024 processors, all of the configurations have very high efficiency. As expected, as the number of islands increases, the efficiency drops slightly due to the overhead introduced by having additional master nodes, the controller node, and the additional communication between these nodes.

When the number of processors increases beyond 1024, the efficiency of the master-slave Borg MOEA rapidly declines. With 16384 processors, the master-slave Borg MOEA is running with an efficiency of 0.064. At this point, the increased overhead and communication burden overloads the single master node and reduces the overall NFE. Increasing the number of islands reduces the workload on individual master nodes, spreading the NFE across multiple islands. Looking at the 16384 processor case in Table 5, switching from the master-slave to a 2 island multi-master configuration increases the efficiency from 0.064 to 0.205. Increasing the number of islands improves the efficiency further, reaching an efficiency of 0.964 with 32 islands.

In Hadka (2013), we developed a discrete event simulation model for accurately predicting the efficiency of the master-slave and multi-master Borg MOEA. Table 5 shows the actual and the predicted efficiency from this model for

Table 5: Table showing the median NFE expended by each implementation and the parallel efficiency.

Processors	Implementation	Total NFE	Efficiency	Predicted Efficiency
1024	Master-Slave Borg	4293080	0.978	0.98
	Multi-Master Borg (2 Islands)	4301767	0.98	0.99
	Multi-Master Borg (4 Islands)	4291951	0.978	0.99
	Multi-Master Borg (8 Islands)	4277744	0.975	0.98
	Multi-Master Borg (16 Islands)	4242323	0.967	0.97
	Multi-Master Borg (32 Islands)	4166046	0.949	0.96
2048	Master-Slave Borg	7755607	0.884	0.91
	Multi-Master Borg (2 Islands)	8610209	0.981	0.97
	Multi-Master Borg (4 Islands)	8609865	0.981	0.98
	Multi-Master Borg (8 Islands)	8588290	0.979	0.98
	Multi-Master Borg (16 Islands)	8552526	0.975	0.97
	Multi-Master Borg (32 Islands)	8478679	0.966	0.97
4096	Master-Slave Borg	7681163	0.438	0.47
	Multi-Master Borg (2 Islands)	16496460	0.94	0.91
	Multi-Master Borg (4 Islands)	17174236	0.979	0.97
	Multi-Master Borg (8 Islands)	17207637	0.98	0.97
	Multi-Master Borg (16 Islands)	17142685	0.977	0.97
	Multi-Master Borg (32 Islands)	17129074	0.976	0.96
8192	Master-Slave Borg	7160437	0.204	0.23
	Multi-Master Borg (2 Islands)	17057671	0.486	0.46
	Multi-Master Borg (4 Islands)	32469898	0.925	0.92
	Multi-Master Borg (8 Islands)	34009570	0.969	0.97
	Multi-Master Borg (16 Islands)	34139711	0.973	0.98
	Multi-Master Borg (32 Islands)	34121055	0.972	0.98
16384	Master-Slave Borg	4470551	0.064	0.08
	Multi-Master Borg (2 Islands)	14385033	0.205	0.23
	Multi-Master Borg (4 Islands)	32373010	0.461	0.47
	Multi-Master Borg (8 Islands)	64639837	0.921	0.91
	Multi-Master Borg (16 Islands)	67101524	0.956	0.96
	Multi-Master Borg (32 Islands)	67661785	0.964	0.97

the LRGV problem. Timing collected from the LRGV runs determined the inputs to the simulation model. These inputs included estimates for the algorithm overhead, $T_A = 0.000105$ seconds, the communication overhead, $T_C = 0.000006$ seconds, and objective function evaluation time, $T_F = 0.14$ seconds. All of these timings were collected on TACC Ranger. From Table 5, we see that the simulation model can very accurately predict the parallel efficiency of the multi-master Borg MOEA.

We expect the multi-master Borg MOEA to be able to efficiently scale to very large processor counts by increasing the number of islands as needed to remain efficient. Using the simulation model, we can predict the efficiency of the multi-master Borg MOEA at larger processor counts. Figure 9 shows the predicted efficiency for the LRGV problem. Note the linear relationship

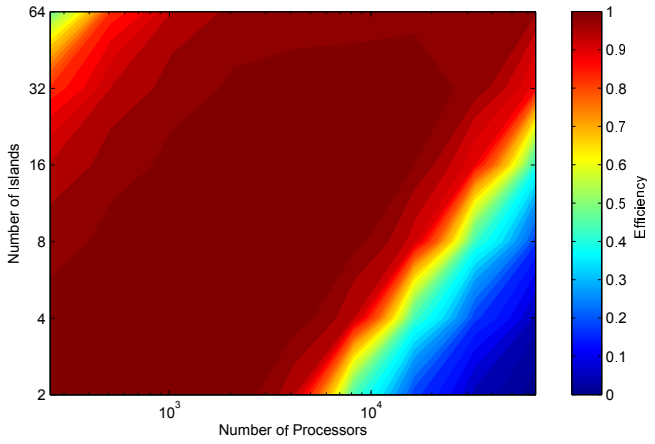

Figure 9: Predicted efficiency for the multi-master Borg MOEA on the LRGV problem from 1024 up to 65536 processors.

between the number of processors and the number of islands. To maintain high efficiency, doubling the number of processors requires the number of islands to double. This maintains a fixed number of processors per island, which is chosen to yield the maximum efficiency. We can use this simulation model to determine the optimal topology for maximizing efficiency.

Maximizing efficiency will increase NFE, but this does not necessarily correspond to increased search quality. It is also necessary to consider how parallelization improves overall search quality. Figure 10 shows the comparative speedup attained when switching from the master-slave to the multi-master Borg MOEA. Each subplot corresponds to a different processor count. The lines within each subplot trace the speedup of that implementation. The baseline is the master-slave Borg MOEA. Results are averaged over 50 random seed trials. The speedup measures how many times faster (or slower) the multi-master Borg MOEA is in attaining the same hypervolume. For example, if the master-slave Borg MOEA reached a hypervolume of 0.8 in 300 seconds, and the multi-master Borg MOEA reached the same hypervolume in 150 seconds, it would show a speedup of 2. Since the master-slave is the baseline, it appears as a flat line with a speedup of 1. Note that these speedup measurements are provided between runs with the same processor count — the computing power is fixed. Thus, any speedup observed is a result of the improved convergence and diversity of a given implementation of the parallel Borg MOEA, and is not a result of more computing power.

With 1024 processors, we see that at low hypervolume thresholds, the multi-master Borg MOEA implementations have lower convergence speeds than the maser-slave Borg MOEA. Only as we increase the hypervolume threshold do the multi-master Borg MOEA implementations begin to converge faster. The

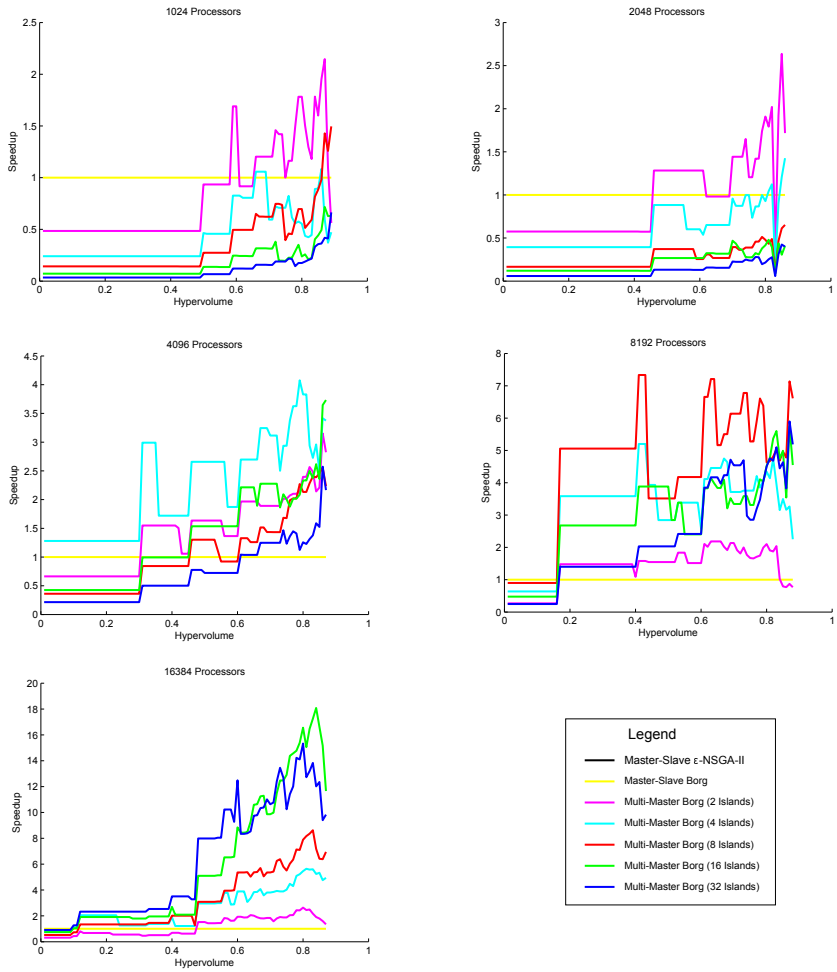

Figure 10: Hypervolume speedup of the multi-master Borg MOEA implementations compared to the baseline master-slave Borg MOEA. These results are averaged over the 50 random seed trials.

master-slave Borg MOEA converges very fast, but it is limited to attaining lower hypervolume than the multi-master Borg MOEA. Note in Figure 10 given that the master-slave Borg MOEA baseline never attains the highest levels of hypervolume, the multi-master Borg MOEA speedup results are conservative. At the largest tested processor count, 16384, we see that the 16 and 32 island multi-master Borg MOEA runs reach a speedup of $10-18$ times faster than the master-slave Borg MOEA. This means these multi-master runs are converging in 1/10th the wallclock time as the master-slave Borg MOEA, even though the master-slave and multi-master are given the same number of processors. This speedup is therefore a result of algorithmic improvements in the multi-master paradigm, allowing the algorithm to better exploit high processor counts and capture the same solution quality in less time. This combined with the global restarts and guidance provided by the controller help improve the speed, effectiveness, and reliability of the multi-master Borg MOEA.

7. Conclusion

The Borg MOEA was originally introduced to solve many-objective, multimodal, non-separable problems. The success of the Borg MOEA has been demonstrated in several studies (Hadka and Reed, 2013, 2012; Hadka et al., 2012; Reed et al., 2013). Application of the Borg MOEA is limited by its serial implementation, which is unable to rapidly solve large-scale problems with expensive objective function evaluations.

To address this limitation, this study developed two parallel versions of the Borg MOEA. The master-slave Borg MOEA runs a parallelized version of the serial Borg MOEA where objective function evaluations are performed in parallel. This provides direct speedup, but is limited by inefficiencies due to the communication overhead that limits its ability to attain very high levels of performance. The multi-master Borg MOEA is a hierarchical extension where two or more islands run instances of the master-slave Borg MOEA in parallel. Additionally, a global controller node maintains the global search state of the algorithm and provides guidance to masters when they preconverge. This guidance extends the restart mechanism and the adaptive selection of search operators of the serial Borg MOEA, allowing for global restarts and sharing of the global search state.

Applying these parallel implementations of the Borg MOEA to a risk-based water supply portfolio planning problem, we observed that the master-slave and multi-master Borg MOEA produced high-quality solutions when compared to another state-of-the-art parallel MOEA, ϵ-NSGA-II. The multi-master Borg MOEA with 32 islands produced the highest-quality results. This is attributed to the ability of the multi-master implementation to quickly detect preconvergence in islands and provide guidance in the form of the global ϵ-dominance archive and global operator probabilities.

The efficiency, reliability, and search quality of the multi-master Borg MOEA have been demonstrated running on up to 16384 processors with over 95% efficiency. We contribute an accurate discrete event simulation of the multi-master

Borg MOEA's parallel efficiency that shows the algorithm has the strong potential for use on emerging Petascale and planned Exascale computing architectures (> 100000 processors). The ability to scale efficiently to high processor counts makes the Borg MOEA a viable tool for solving extremely large-scale, complex environmental problems. For the LRGV problem explored in this study, the 32 island multi-master Borg MOEA solved the problem with the highest-quality results in 10 minutes using 16384 processors. If running in serial, this would require over 109 days of computation. This opens the possibility for further research to explore the value of the multi-master Borg MOEA for addressing more complex environmental systems effectively while providing decision-makers with the ability to rapidly evaluate their tradeoffs, formulations, and potential design solutions.

Acknowledgments

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing high-performance computing resources that have contributed to the research results reported within this paper. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

References

Alba, E., Luque, G., Nesmachnow, S., 2013. Parallel metaheuristics: recent advances and new trends. In: International Transactions in Operational Research. Vol. 20. pp. 1–48.

Amdahl, G. M., 1967. Validity of the single processor approach to achieving large scale computing capabilities. In: American Federation of Information Processing Societies Spring Computer Conference.

Amdahl, G. M., 1988. Limits of expectation. International Journal of High Performance Computing Applications 2, 88–94.

Anderson, T. L., Hill, P. J., 1997. Water Marketing: The Next Generation. Rowman and Littlefield, Lanham, MD.

Bäck, T., Fogel, D. B., Michalewicz, Z., 1997. Handbook of Evolutionary Computation. Taylor & Francis, New York, NY.

Brekke, L. D., Maurer, E. P., Anderson, J. D., Dettinger, M. D., Townsley, E. S., Harrison, A., Pruitt, T., 2009. Assessing reservior operations risk under climate change. Water Resources Research 45 (4).

Brill, E. D., Flach, J., Hopkins, L., Ranjithan, S., 1990. MGA: A decision support system for complex, incompletely defined problems. IEEE Transactions on Systems, Man, and Cybernetics 20 (4), 745–757.

Cantú-Paz, E., 2000. Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Norwell, MA.

Castelletti, A., Galelli, S., Ratto, M., Soncini-Sessa, R., Young, P., 2012. A general framework for dynamic emulation modelling in environmental problems. Environmental Modelling & Software 34, 5–18.

Characklis, G., Kirsch, B. R., Ramsey, J., Dillard, K., Kelley, C. T., 2006. Developing portfolios of water supply transfers. Water Resources Research 42 (5).

Coello Coello, C. A., Lamont, G. B., Van Veldhuizen, D. A., 2007. Evolutionary Algorithms for Solving Multi-Objective Problems. Springer Science+Business Media LLC, New York, NY.

Cohon, J., Marks, D., 1975. A review and evaluation of multiobjective programming techniques. Water Resources Research 11 (2), 208–220.

Deb, K., Agrawal, R. B., Nov. 1994. Simulated binary crossover for continuous search space. Tech. Rep. IITK/ME/SMD-94027, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India.

Deb, K., Joshi, D., Anand, A., 2002. Real-coded evolutionary algorithms with parent-centric recombination. Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002) 1, 61–66.

Feyen, L., Vrugt, J. A., Nuallain, B. O., van der Knijff, J., Roo, A. D., 2007. Parameter optimisation and uncertainty assessment for large-scale streamflow simulation with the lisflood model. Journal of Hydrology 332 (3-4), 276–289.

Franssen, M., 2005. Arrow's theorem, multi-criteria decision problems and multi-attribute preferences in engineering design. Research in Engineering Design 16 (1), 42–56.

Frederick, K. D., Schwarz, G., 1999. Socioeconomic impacts of climate change on u.s. water supplies. Journal of the American Water Resources Association 35 (6), 1563–1583.

Fu, G., Kapelan, Z., Kasprzyk, J., Reed, P. M., 2013. Optimal design of water distribution systems using many-objective visual analytics. ASCE Journal of Water Resources Planning & Management 139 (6), 624–633.

Guidolin, M., Fu, G., Reed, P., 2012. Parallel evolutionary multiobjective optimization of water distribution system design. In: 14th Water Distribution Systems Analysis Conference. Adelaide, Australia.

Hadjigeorgalis, E., 2008. Managing drought through water markets: Farmer preferences in the rio grande basin. Journal of the American Water Resources Association 44 (3), 594–605.

Hadka, D., 2013. The foundations, parallelization, and application of the borg multiobjective evolutionary algorithm. Ph.D. thesis, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA.

Hadka, D., Madduri, K., Reed, P., 2013. Scalability analysis of the asynchronous, master-slave borg multiobjective evolutionary algorithm. In: IEEE International Parallel and Distributed Processing Symposium (IPDPS), Nature Inspired Distributed Computing Workshop (NIDISC). Cambridge, MA.

Hadka, D., Reed, P., 2012. Diagnostic assessment of search controls and failure modes in many-objective evolutionary optimization. Evolutionary Computation 20 (3), 423–452.

Hadka, D., Reed, P., 2013. Borg: An auto-adaptive many-objective evolutionary computing framework. Evolutionary Computation 21 (2), 231–259.

Hadka, D., Reed, P., Simpson, T., June 2012. Diagnostic assessment of the borg moea for many-objective product family design problems. In: WCCI 2012 World Congress on Computational Intelligence, Congress on Evolutionary Computation (CEC 2012). Brisbane, Australia, pp. 986–995.

Horn, J., 1995. The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations. Ph.D. thesis, University of Illinois, Urbana-Champaign, Illinois.

Israel, M., Lund, J. R., 1995. Recent california water transfers: Implications for water management. Natural Resources Journal 35 (1), 1–32.

Jenkins, M. W., Lund, J. R., 2000. Integrating yield and shortage management under multiple uncertainties. Journal of the American Water Resources Association 126 (5), 288–297.

Jin, Y., Olhofer, M., Sendhoff, B., 2002. A framework for evolutionary optimization with approximate fitness function. IEEE Transactions on Evolutionary Computation 6 (5), 451–494.

Kasprzyk, J., Nataraj, S., Reed, P. M., Lempert, R. J., 2013. Many-objective robust decision making for complex environmental systems undergoing change. Environmental Modelling & Software 42, 55–71.

Kasprzyk, J., Reed, P., Kirsch, B., Characklis, G., 2012. Many-objective de Novo water supply portfolio planning under deep uncertainty. Environmental Modelling & Software 34, 87–104.

Kasprzyk, J. R., Reed, P. M., Kirsch, B. R., Characklis, G. W., 2009. Managing population and drought risks using many-objective water portfolio planning under uncertainty. Water Resources Research 45 (12).

Kirsch, B. R., Characklis, G. W., Dillard, K., Kelley, C. T., 2009. More efficient optimization of long-term water supply portfolios. Water Resources Research 45 (3).

Kita, H., Ono, I., Kobayashi, S., 1999. Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC 1999). Washington, DC, pp. 1581–1588.

Kollat, J., Reed, P., Maxwell, R., 2011. Many-objective groundwater monitoring network design using bias-aware ensemble kalman filtering, evolutionary optimization, and visual analytics. Water Resources Research 47 (2).

Kollat, J. B., Reed, P. M., 2006. Comparison of multi-objective evolutionary algorithms for long-term monitoring design. Advances in Water Resources 29 (6), 792–807.

Kundzewicz, Z., Mata, L., Arnell, N., Doll, P., Kabat, P., Jimenez, B., Miller, K. A., Oki, T., Sen, Z., Shiklomanov, I., 2007. Freshwater resources and their management. In: Climate Change 2007: Impacts, Adaptation and Vulnerability. Cambridge University Press, Cambridge, England, pp. 173–210.

Lane, M. E., Kirshen, P. H., Vogel, R. M., 1999. Indicators of impacts of global climate change on u.s. water resources. Journal of Water Resources Planning and Management 125 (4), 194–204.

Lund, J. R., 1995. Derived estimation of willingness to pay to avoid probabilistic shortage. Water Resources Research 31 (5), 1367–1372.

Macdonald, I., 2009. Comparison of sampling techniques on the performance of monte-carlo based sensitivity analysis. In: 11th International IBPSA Conference (Building Simulation 2009). Glasgow, Scotland, pp. 992–999.

Mahfoud, S. W., 1995. Niching methods for genetic algorithms. Ph.D. thesis, Department of Computer Science, University of Illinois, Urbana-Champaign, IL.

Maier, H. R., Kapelan, Z., Kasprzyk, J., Kollat, J., Matott, L. S., da Conceição Cunha, M., Dandy, G. C., Gibbs, M. S., Keedwell, E., Marchi, A., Ostfeld, A., Savic, D., Solomatine, D., Vrugt, J. A., Zecchin, A. C., Minsker, B. S., Barbour, E., Kuczera, G., Pasha, F., Castelletti, A., Reed, P., In-Press. Evolutionary algorithms and other metaheuristics in water resources: Current status, research challenges and future directions.

Matott, L. S., Bartelt-Hunt, S. L., Rabideau, A. J., Fowler, K., 2006a. Application of heuristic optimization techniques and algorithm tuning to multilayered sorptive barrier design. Environmental science & technology 40 (20), 6354–6360.

Matott, L. S., Rabideau, A. J., Craig, J. R., 2006b. Pump-and-treat optimization using analytic element method flow models. Advances in Water Resources 29 (5), 760–775.

Milly, P. C. D., Betancourt, J., Falkenmark, M., Hirsch, R., Kundzewicz, Z. W., Lettenmaier, D. P., Stouffer, R. J., 2008. Stationarity is dead: Whither water management? Science 319, 573–574.

National Research Council, 2009. Informing decisions in a changing climaterep. The National Academies Press.

National Research Council, 2012. Challenges and opportunities in the hydrologic sciences. The National Academies Press.

Nicklow, J., Reed, P., Savic, D., Dessalegne, T., Harrell, L., Chan-Hilton, A., Karamouz, M., Minsker, B., Ostfeld, A., Singh, A., Zechman, E., 2010. State of the art for genetic algorithms and beyond in water resources planning and management. Journal of Water Resources Planning and Management 136, 412–432.

Purshouse, R. C., Fleming, P. J., Dec. 2003. Evolutionary many-objective optimisation: An exploratory analysis. In: Congress on Evolutionary Computation (CEC 2003). Canberra, Australia, pp. 2066–2073.

Purshouse, R. C., Fleming, P. J., 2007. On the evolutionary optimization of many conflicting objectives. IEEE Transactions on Evolutionary Computation 11 (6), 770–784.

Razavi, S., Tolson, B., Burn, D., 2012. Review of surrogate modeling in water resources. Water Resources Research 48 (W07401).

Reed, P. M., Hadka, D. M., Herman, J. D., Kasprzyk, J. R., Kollat, J. B., 2013. Evolutionary multiobjective optimization in water resources: The past, present, and future. Advances in Water Resources 51, 438–456.

Reed, P. M., Kollat, J. B., 2013. Visual analytics clarify the scalability and effectiveness of massively parallel many-objective optimization: A groundwater monitoring design example. Advances in Water Resources 56, 1–13.

Reed, P. M., Kollat, J. B., Ferringer, M. P., Thompson, T. G., Nov. 2008. Parallel evolutionary multi-objective optimization on large, heterogeneous clusters: An applications perspective. Journal of Aerospace Computing, Information, and Communication 5 (11), 460–478.

Regis, R. G., Shoemaker, C. A., 2007. Parallel radial basis function methods for the global optimization of expensive functions. European journal of operational research 182 (2), 514–535.

Roshani, E., Filion, Y., Sept. 2012. Using parallel computing to increase the speed of water distribution network optimization. In: WDSA 2012: 14th Water Distribution Systems Analysis Conference. Adelaide, Australia.

Sayeed, M., Mahinthakumar, G. K., 2005. Efficient parallel implementation of hybrid optimization approaches for solving groundwater inverse problems. Journal of Computing in Civil Engineering 19 (4), 329–340.

Shah, R., Reed, P. M., Simpson, T., 2011. Many-objective evolutionary optimization and visual analytics for product family design. In: Wang, L., Ng, A. H. C., Deb, K. (Eds.), Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing. Springer, London, England, pp. 137–159.

Sheskin, D. J., 2004. Handbook of Parametric and Nonparametric Statistical Procedures. Chapman & Hall/CRC, Boca Raton, FL.

Storn, R., Price, K., 1997. Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11 (4), 341–359.

Tang, Y., Reed, P. M., Kollat, J. B., 2007. Parallelization strategies for rapid and robust evolutionary multiobjective optimization in water resources applications. Advances in Water Resources 30, 335–353.

Top 500 Supercomputer Sites, 2014. Statistics of Efficiency, Power, Cores 1993-2014. http://www.top500.org/statistics/efficiency-power-cores/ [Online, Accessed 14 August 2014].

Tsutsui, S., Yamamura, M., Higuchi, T., 1999. Multi-parent recombination with simplex crossover in real coded genetic algorithms. In: Genetic and Evolutionary Computation Conference (GECCO 1999). Orlando, FL, pp. 657–664.

Vorosmarty, C. J., Green, P., Salisbury, J., Lammers, R. B., 2000. Global water resources: Vulnerability from climate change and population growth. Science 289, 284–288.

Vrugt, J. A., Braak, C. J. T., Clark, M. P., Hyman, J. M., Robinson, B. A., 2008. Treatment of input uncertainty in hydrologic modeling: Doing hydrology backward with markov chain monte carlo simulation. Water Resources Research 44 (12).

Vrugt, J. A., Robinson, B. A., 2007. Improved evolutionary optimization from genetically adaptive multimethod search. Proceedings of the National Academy of Sciences 104 (3), 708–711.

Vrugt, J. A., Robinson, B. A., Hyman, J. M., April 2009. Self-adaptive multimethod search for global optimization in real-parameter spaces. IEEE Transactions on Evolutionary Computation 13 (2), 243–259.

Watkins Jr., D. W., McKinney, D. C., 1999. Screening water supply options for the edwards aquifier region in central texas. Journal of Water Resources Planning and Management 125 (1), 14–24.

While, L., Bradstreet, L., Barone, L., 2012. A fast way of calculating exact hypervolumes. IEEE Transactions on Evolutionary Computation 16 (1), 86–95.

Wilchfort, O., Lund, J. R., 1997. Shortage management modeling for urban water supply systems. Journal of the American Water Resources Association 123 (4), 250–258.

Woodruff, M., Hadka, D., Reed, P., Simpson, T., Sept. 2012. Auto-adaptive search capabilities of the new borg moea: A detailed comparison on product family design problems. In: 12 AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. Indianapolis, IN, aIAA Paper AIAA-2012-5442.

Woodruff, M., Reed, P. M., Simpson, T., 2013. Many-objective visual analytics: Rethinking the design of complex engineered systems. Structural and Multidisciplinary Optimization 48, 201–219.

Zhang, X., Beeson, P., Link, R., Manowitz, D., Izaurralde, R. C., Sadeghi, A., Thomson, A. M., Sahajpal, R., Srinivasan, R., Arnold, J. G., 2013. Efficient multi-objective calibration of a computationally intensive hydrologic model with parallel computing software in python. Environmental Modelling & Software 46, 208–218.

Zheng, Y. W., Morad, B., Sept. 2012a. Comparing methods of parallel genetic optimization for pump scheduling using hydraulic model and gpu-based ann meta-model. In: WDSA 2012: 14th Water Distribution Systems Analysis Conference. Adelaide, Australia.

Zheng, Y. W., Morad, B., Sept. 2012b. Real-time pump scheduling using genetic algorithm and artificial neural network based on graphics processing unit. In: WDSA 2012: 14th Water Distribution Systems Analysis Conference. Adelaide, Australia.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., da Fonseca, V. G., 2002. Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7 (2), 117–132.