JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 5, November 2008

Parallel Evolutionary Multi-Objective Optimization on Large, Heterogeneous Clusters: An Applications Perspective

Patrick M. Reed[*] and Joshua B. Kollat[*†]
The Pennsylvania State University, University Park, PA 16802

Matthew P. Ferringer[‡] and Timothy G. Thompson[§]
The Aerospace Corporation, Chantilly, VA 20151

DOI: 10.2514/1.36467

Real-world operational use of parallel multi-objective evolutionary algorithms requires successful searches in constrained wall-clock periods, limited trial-and-error algorithmic analysis, and scalable use of heterogeneous computing hardware. This study provides a cross-disciplinary collaborative effort to assess and adapt parallel multi-objective evolutionary algorithms for operational use in satellite constellation design using large dedicated clusters with heterogeneous processor speeds/architectures. A statistical, metric-based evaluation framework is used to demonstrate how time-continuation, asynchronous evolution, dynamic population sizing, and epsilon dominance archiving can be used to enhance both simple master–slave parallelization strategies and more complex multiple-population schemes. Results for a benchmark constellation design coverage problem show that simple master–slave schemes that exploit time-continuation are often sufficient and potentially superior to complex multiple-population schemes.

Nomenclature

A	epsilon dominance archive
a	semimajor axis
e	eccentricity
$\boldsymbol{f}$	objective function vector
I	inclination
k	objective functions index
M	mean anomaly
m	inequality constraints
N	population size
P^*	Pareto optimal set
PF^*	Pareto front

Received 4 January 2008; accepted for publication 13 May 2008. Copyright 2008 by Patrick M. Reed, Joshua B. Kollat, Matthew P. Ferringer and Timothy G. Thompson. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1542-9423/08 $10.00 in correspondence with the CCC.

[*] Assistant Professor, Department of Civil Engineering, 212 Sackett Building.

[†] Ph.D. Candidate, Civil Engineering Department, 212 Sackett Building.

[‡] Senior Member of the Technical Staff and Ph.D. Candidate, Performance Modeling and Analysis Department, 15049 Conference Center Dr./CH1-410, AIAA Professional Member, matthew.ferringer@aero.org.

[§] Senior Engineering Specialist, Performance Modeling and Analysis Department, 15049 Conference Center Dr./CH1-410, AIAA Professional Member.

p	equality constraints
q	discrete points in target grid
r	visibility gaps
$\boldsymbol{x}$	decision variable vector
Λ	objective space
Ψ	decision space
ω	argument of perigee
Ω	right ascension of the ascending node

I. Introduction

THIS study provides a perspective on designing parallel multi-objective evolutionary algorithms (pMOEAs) for operational use in satellite constellation design, which can differ substantially from the analysis of artificial test functions [1–3]. The term ‘operational use’ in the context of the aerospace domain requires that pMOEAs should ideally

a) maximize successful search given a constrained wall-clock period
b) limit trial-and-error analysis for parameterization and algorithmic design for a domain’s multi-objective problems (MOPs)
c) reduce random seed variability for the run-time dynamics and end-of-run results
d) scale-well for evolving investments in computing where new hardware resources are being added over annual time-scales.

Addressing these issues is challenging given the dearth of guidance for real-world practitioners seeking to design and use pMOEAs [1,4].

Using the terminology of Cantu-Paz [5], this study focuses on the master–slave (MS) and multiple-population (MP) parallelization schemes. Prior pMOEA literature reviews [1,4] show a strong preferential focus on the MP scheme. Studies focusing on the MP scheme typically criticize the MS scheme’s inability to maintain diverse search as well as its inability to scale well for large clusters [1,6]. Another factor contributing to the popularity of the MP scheme is the existing literature’s treatment of “superlinear” speedup [1,5]. For parallel applications speedup is defined as the ratio of total serial wall-clock time to the parallel wall-clock time for completing an application [7]. Linear speedup occurs when this ratio is equal to the number of processors used. Superlinear speedup requires that the ratio of serial to parallel wall-clock times yield a number larger than the parallel processor count. Although superlinear speedups are theoretically possible for homogeneous and heterogeneous clusters [8], there has been significant debate and contradictory recommendations in the prior parallel evolutionary computation (EC) literature as noted in recent reviews [1,5]. Our earlier pMOEA work [2] has shown that when solution quality is reported carefully in conjunction with MP-based speedup, superlinear speedup is highly dynamic, difficult to sustain, and challenging to predict a priori.

From a real-world applications perspective, the parallel EC literature appears to predominantly focus on artificially constructed test problems that often represent problem difficulties that are limited in their value for designing and assessing pMOEAs for real-world applications [1,3]. Additionally, Van Veldhuizen et al. [1] highlight that the earlier pMOEA literature’s focus on MP-schemes is counterintuitive given their increased complexity relative to MS-strategies. These challenges were independently encountered by the authors’ previous work [2,9] and motivated our collaboration in this study. In the aerospace domain, Ferringer et al. [9] focused on developing MS and MP versions of the nondominated sorted genetic algorithm-II (NSGA-II) [10] for a benchmark satellite constellation design problem [11]. The study highlighted that the primary challenge for both MS and MP schemes resulted from the enumerative evolutionary algorithm (EA) parameter analysis required to attain high quality search on a heterogeneous cluster, particularly when considering population sizing and migration strategies.

The parameterization challenges faced by Ferringer et al. [9] motivated their collaborative interest in the parallelization work of Tang et al. [2]. Tang et al. [2] evaluated MS and MP versions of an epsilon-dominance archiving version of NSGA-II (ε-NSGA-II) designed for small, homogeneous LINUX workstation clusters (<16 processors).

The serial ε-NSGA-II was developed [12,13] to

a) maintain diverse representations of large real-world solution sets
b) enhance search with an auto-adaptive population sizing operator
c) use time-continuation [14] to maintain search for as long as is computationally feasible or sufficient
d) theoretically bound the solution set size and auto-adapted population size based on user specified epsilon values for each objective [15].

Tang et al. [2] generalized the auto-population sizing and time continuation components of ε-NSGA-II for both MS and MP implementations. The MS and MP versions of the algorithm were tested on a suite of test problems consisting of the DTLZ6 test function [16] and two benchmark water resources problems capturing continuous and discrete decision spaces. As will be discussed in more detail in Sec. II.B, the MS and MP versions of ε-NSGA-II were developed to minimize user interaction and to take advantage of time continuation (i.e., the injection of randomly generated solutions to enhance population diversity). Overall the MP version of ε-NSGA-II had superior performance for DTLZ6 and the less complicated MS version of the algorithm was significantly better for both water resources applications. Two separate pMOEA failure modes were identified: a) the absolute problem difficulty for DTLZ6 caused the serial and consequently the MS version of the ε-NSGA-II to have low success rates; and b) the water resources applications suffered from insufficient evolutionary search periods without parallelization. Tang et al. [2] used a statistical metrics-based evaluation to show that the MS version of the ε-NSGA-II maintains diverse search, has predictable speedup for small homogeneous clusters, and was able reliably to solve both water resources benchmark problems.

This study builds on work by Ferringer et al. [9] and Tang et al. [2] to provide a cross-disciplinary collaborative effort to comprehensively assess and adapt pMOEAs for the design of satellite constellations using large clusters with heterogeneous processor speeds and architectures (single, dual, and quad core processors). Van Veldhuizen et al. [1] has highlighted that there is a present need for real-world practitioners to collaboratively characterize pMOEA performance across MOP domains while clearly elucidating hardware considerations and algorithmic design rationale. Additionally, there is also a need to advance the pMOEA literature with real-world applications that report statistical, metrics-based evaluations of search effectiveness and parallel performance [1,4]. This paper tests the generality of the findings of Tang et al. [2] and provides detailed recommendations for adapting MS and MP schemes for larger, heterogeneous clusters with dedicated processing.

Section II of this paper introduces general multi-objective terminology, overviews the MS and MP versions of the ε-NSGA-II tested in this work, and provides a detailed introduction to the satellite constellation design application. Section III discusses the computational experiment used to test the parallelization schemes considered in terms of their parameterizations, the hardware details for The Aerospace Corporation's Fellowship cluster, and the metrics-based evaluative framework used to assess pMOEA performance. In Sec. IV comparative results for the original MS and MP-schemes of Tang et al. [2] designed for a small homogeneous cluster are contrasted to new versions adapted for larger-scale, heterogeneous parallel computing. Sections V and VI provide a discussion of the implications of this study and the overall conclusions of this work.

II. Methodology

A. Overview of Multi-objective Optimization Terms

Adapting the pMOEA notation and nomenclature of Van Veldhuizen et al. [1] and assuming minimization, a MOP can be generally defined using a vector function composed of up to k component functions

$$\text{Min}\ \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_k(\boldsymbol{x}) \end{bmatrix} \tag{1}$$

s.t.

$$g_i(\boldsymbol{x}) \geqslant 0.\, i = 1, 2, \ldots, m \tag{2}$$

$$h_i(\boldsymbol{x}) = 0.\, i = 1, 2, \ldots, p \tag{3}$$

In Eq. 1, the vector $\boldsymbol{x}$ is defined over the n-dimensional potential combination of binary, integer, real, or mixed decision variables that define the overall decision space ψ. The performance of each design/decision $\boldsymbol{x}$ vector is evaluated in terms of the k-dimensional $\Re^k$ objective space designated as Λ. As shown in Eqs. (1) to (3), the goal of multi-objective optimization is to identify a decision vector $\boldsymbol{x}^* = \left[x_1^*, x_2^*, \ldots, x_n^*\right]^T$ that satisfies m inequalities and p equality constraints while also attaining optimal values for the k-component objectives.

Unlike single objective optimization where a single optimal solution can be identified, MOPs solved using pMOEAs generally possess conflicts or tradeoffs among the k-component objectives, which means that optimality with respect to one objective may actually degrade system performance for one or more of the remaining objectives. The emerging popularity of MOEAs for solving real-world applications results from their ability to evolve the Pareto optimal $P^* \in \Psi$ set, composed of all decision vectors in which any component objective value can only be improved by reducing the optimality of at least one other objective. When mapped from the decision space to the objective, $\vec{f}$: $\Psi \rightarrow \Lambda$, the Pareto optimal set defines the Pareto front $PF^* \in \Lambda$ that composes up to a $k-1$ dimensional surface representing objective tradeoffs. An applications review as well as more formal mathematical introductions for MOPs, nondomination sorting, and Pareto optimality can be referenced in [4] and/or [17].

B. Satellite Constellation Design Problem

Several analytical solutions exist that provide the minimum number of satellites and their associated geometry required to achieve continuous global coverage [18,19]. In practice, a continuous coverage constellation may not be realizable due to resource limitations or desirable owing to program requirements. If a coverage gap is acceptable, then a challenging optimization problem emerges wherein a designer would like to minimize the maximum visibility gap, known as the maximum revisit time (MRT). However, by minimizing the MRT, the average coverage gap, or average revisit time (ART) will increase because resources used to revisit any one point more frequently than another are wasted [20]. For a discontinuous coverage constellation, a decision maker seeks out approximations of the Pareto-optimal set P^* that characterize this trade-off.

Before the development of MOEAs, constellation design tradeoffs were typically quantified using enumeration [21] owing to the nonlinear and discontinuous nature of the coverage objective functions. As constellations grow beyond one or two satellites, these analyses quickly become infeasible and/or inaccurate owing to computational resource limitations and the coarse resolution by which the design space is searched. During the previous decade, several researchers [9,11,21–24] have shown MOEAs to be effective tools for generating approximations of non-dominated sets in the constellation performance space. Several variations of the revisit time tradeoff reoccur in the literature, the most recent [9] of which provides the benchmark problem used for this study.

The benchmark problem addresses three-satellite constellation designs that minimize both the MRT and ART in a region of discrete points (made up of an equal area grid) overlaying the landmass of the conterminous United States. MRT is calculated using Eq. 4 where, for each discrete point, q, all visibility gaps, r, to that point are stored and the maximum gap is reported. ART is calculated using Eq. 5 by averaging all gaps to all points.

$$f_1(\boldsymbol{x}) = \max\{\max\{gaps_{i,j}(\boldsymbol{x})\}\}, \quad i = 1, 2, \ldots, q \quad \text{and} \quad j = 1, 2, \ldots, r \tag{4}$$

$$f_2(\boldsymbol{x}) = \sum_i^g \frac{\left[\sum_{j=1}^p gaps_{i,j}(\boldsymbol{x})\right]/r}{q}, \quad i = 1, 2, \ldots, q \quad \text{and} \quad j = 1, 2, \ldots, r \tag{5}$$

The objective functions are evaluated using coverage analysis software [25] developed by The Aerospace Corporation. For detailed information concerning the Astrodynamics assumptions for this benchmark problem see Ferringer et al. [9].

A satellite's orbit is defined by six variables: semimajor axis (a), eccentricity (e), inclination (I), right ascension of the ascending node (Ω), argument of perigee (ω), and mean anomaly (M). For the three-satellite constellation, there are a total of 18 potential design variables. Searching this entire design space is typically not done in practice. As any given program will typically define several requirements a priori that constrain the variables. For the benchmark problem these 18 potential decision variables are subject to a variety of pre-defined design specifications [9], which reduce the decision vector to the five variables shown in Eq. 6.

$$\boldsymbol{x}^* = [I, \Omega_2, \mathrm{M}_2, \Omega_3, \mathrm{M}_3]^T \tag{6}$$

The decision variables were enumerated via binary encoding to enable exact identification of the PF^* such that an exhaustive search of the decision space could be performed in a reasonable amount of time. The upper and lower bounds for each variable are shown in Eqs. 7 to 9.

$$0^\circ < I \leqslant 180^\circ \tag{7}$$

$$0^\circ < \Omega \leqslant 360^\circ \tag{8}$$

$$0^\circ < M \leqslant 360^\circ \tag{9}$$

Four bits are used to encode Ω and M while seven encode I. As the objective functions are most sensitive to the latter variable. After duplicate configurations are deleted, 3,171,952 unique designs are evaluated resulting in a PF^* with 262 solutions. Adding a single bit to Ω and M would increase the number of unique solutions to 58,920,960 rending exhaustive search to find PF^* impractical.

C. Overview of Epsilon-Dominance NSGA-II (ε-NSGA-II)

Previous studies have demonstrated that the ε-NSGA-II is generally superior to the NSGA-II parent algorithm [10] and compares favorably to other state-of-the-art multi-objective search algorithms for both test functions and a suite of real-world applications [12,13,26]. The ε-NSGA-II was developed with the intent of solving computationally intensive real-world applications with large Pareto optimal solution sets P^*. Operational use of the algorithm in time-constrained applications requires search to be reliable, effective, and efficient. Reliability and effectiveness can be discussed jointly in terms of maximizing search quality and repeatability across random seeds given user-defined computing/time constraints. Efficiency is related to both minimizing the number of evaluations per run as well as the overall number of runs (i.e., random seed analysis) required to provide approximate results for challenging problems.

In an operational context, the ε-NSGA-II seeks to enhance search reliability, effectiveness, and efficiency using three primary innovations relative to the parent NSGA-II algorithm

a) epsilon-dominance archiving [15]
b) auto-adaptation of population sizes
c) the use of time continuation [14] to maintain diverse search for as long as is computationally feasible or sufficient.

Figure 1 provides a schematic overview for the ε-NSGA-II algorithm. The epsilon-dominance archive shown in Fig. 1 serves several roles in the ε-NSGA-II. The user-specified epsilon grid for the objective space is important for real-world applications where objectives' values are often meaningless beyond certain levels of significant precision due to data uncertainties, numerical error, and design limits. As noted by Laumanns et al. [15] epsilon block non-domination sorting will promote an even search of the objective space and theoretically limits an MOEA's archive to a finite size. Kollat and Reed [27] demonstrated that epsilon-dominance archiving can be used dramatically to reduce the computational demands of real-world applications with large Pareto optimal solution sets by providing users with the ability to control the degree of approximation used in search.

Although the ε-NSGA-II maintains the basic representations, operators, and options of the original NSGA-II [10], Fig. 1 illustrates that epsilon-dominance archiving significantly changes the manner in which search progresses. The algorithm does not follow the "finite run" defined traditionally as the product of the number of generations and a static population size. Instead for the ε-NSGA-II, users simply specify either a maximum number of function evaluations or wall-clock time for search. The algorithm uses a series of "connected runs" that are initiated in run 1 with an arbitrarily small population ($\sim$12 individuals) that performs preliminary search at minimal computational expense. Evolution with the preliminary population continues for 250 generations as recommended by Deb et al. [10]. Figure 1 illustrates that after run 1 the epsilon-dominance archive A_1 is injected into the initial generation of run 2.

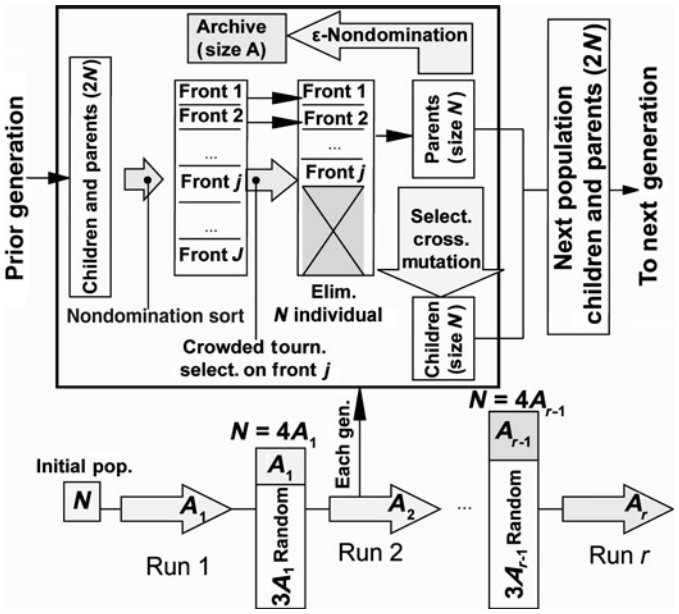

Fig. 1 Overview of the ε-NSGA-II algorithm adapted from Kollat and Reed [13].

In run 2, the population size is adapted to be $4 \times A_1$ in which 25% of the solutions are archived solutions and 75% are randomly generated using uniform distributions. This 25% injection strategy pre-conditions search using auto-adapted population sizes that grow linearly with the size of the epsilon-dominance archive. In the long term at run r, when the epsilon-dominance archive's size stabilizes the search population size also stabilizes. The injection of random solutions serves to maintain population diversity and "continue" search for the user-specified maximum search period. The use of time continuation in ε-NSGA-II is particularly important for real-world MOP's where search failures often result from insufficient search because of computational time constraints or resource limits for dedicated use of computing hardware. Extending Tang et al. [2], this paper seeks to develop parallelization schemes that exploit ε-NSGA-II's epsilon dominance archiving, auto-adaptive population sizing, and time continuation effectively on large, heterogeneous clusters.

D. Parallelization Schemes

There are several design considerations when shifting pMOEAs from small, homogeneous clusters (<16 processors) to larger, heterogeneous systems. From a hardware perspective it is important to characterize the degree of heterogeneity expected [8,28–30]. Bazterra et al. [28] decompose the influence of variable processor speeds, architectures, and processing memory into evaluation time effects and idle time effects. Evaluation time represents the wall-clock period for basic pMOEA search operations, of which function evaluations are often the primary computational concern. Idle time can be defined as the cumulative wall-clock time where processors are being poorly used owing to message communication, synchronization issues, I/O, and contention from other processes [28]. As noted in several earlier studies [5–8,28–30], the ratio of evaluation time to idle time asymptotically limits the potential speedup or scalability of pMOEAs. These issues guided our changes to the original MS and MP versions of the ε-NSGA-II [3].

In this study, four algorithm variants will be compared

a) a MS implementation designed for small, homogeneous clusters (MS_SC)
b) a MS variant designed for large, heterogeneous clusters (MS_LC)
c) a MP implementation designed for small, homogeneous clusters (MP_SC)
d) a MP variant designed for large, heterogeneous clusters (MP_LC).

The MS_SC and MP_SC schemes correspond to the initial parallelized versions of the ε-NSGA-II presented by Tang et al. [2]. The MS_LC and MP_LC configurations of ε-NSGA-II represent new contributions for this study. All parallel versions of the ε-NSGA-II where developed using the message passing interface (MPI) framework [31].

1. Master Slave Implementation for Small Clusters

The MS_SC version of the ε-NSGA-II represents the simplest possible parallelization strategy for MOEAs, in which a master processor controls the evolutionary operations of the code and slave processors simply perform function evaluations. In this parallelization scheme, the master processor initially sends a single individual to each slave for evaluation. If population members remain after this initial allocation to slaves, the master processor passes the remaining individuals as slave processors complete their evaluations. After all population members have been evaluated, the master processor performs the generational evolutionary operations shown in Fig. 1. In the MS_SC as implemented by Tang et al. [2], the master processor also performs function evaluations when it is not coordinating other evolutionary operations or processing messages.

The MS_SC strategy is simple to implement but limited in its scalability owing to the challenges posed by generational synchronization. Synchronization occurs at the end of each generation beginning when the master sends out the last individual of the population and ends when the final evaluation is completed. Note that the longest elapsed time will occur when the final individual is sent to the slowest processor. For computationally intensive applications, the time required to perform the evolutionary operations is very low compared with the generational synchronization costs. The search dynamics of the ε-NSGA-II exacerbate generational synchronization problems due to the algorithm's tendency to favor long run durations (i.e., high generation counts). The generational synchronization challenges for the MS_SC scheme cause processor idle times to grow quickly with increasing processor counts, especially for heterogeneous clusters [6,28].

2. Master–Slave Implementation for Large Clusters

The MS_LC version of the ε-NSGA-II uses the algorithm's variable population size to address synchronization problems through the use of asynchronous evolution. Generational synchronization is eliminated because the master processor does not wait for slow function evaluations within any given population before it evolves the next generation. Instead, these individuals are added to a subsequent child population, in which case their generation counts (or run numbers) can vary with heterogeneous processing speeds. Even though the child population size varies, it reaches a steady state where the number of individuals left-behind roughly equals the number of individuals from earlier generations whose function evaluations have finally completed.

The asynchronous evolution within the MS_LC scheme results in an interesting special case that occurs with large processor counts. When the initial population size is smaller than the number of available processors, the idle processors are passed new randomly generated solutions to evaluate. The new random solutions provide load balance and support the time-continuation of search. An additional performance boost is obtained because the master does not participate in performing function evaluations. The master is ready to respond as quickly as possible to a slave's request for more work and it asynchronously evolves the next generation while the slaves are processing individuals.

3. Multiple Population Implementation for Small Clusters

The MP_SC version of the ε-NSGA-II represents a far more complex algorithm in terms of its message passing interface implementation and overall design. Figure 2 shows that each of the W processors has a fully functional version of the algorithm exploring the full decision space. In general MP-schemes require developers to specify the size and number of populations (or demes), processor topologies (or connectivity), and migration policies. Cantu-Paz's work [5] analyzing MP-schemes highlights the following general conclusions: a) minimal deme sizes maximize the potential for parallel speed ups; b) problem difficulty heavily influences deme sizing requirements and consequently the potential for parallel speedups; and c) selection pressure and convergence dynamics are heavily controlled by migration policies. Migration policies are defined in terms of the frequency of solution exchange, the rate or number of solutions exchanged, and the strategies governing how selected migrants from sending populations

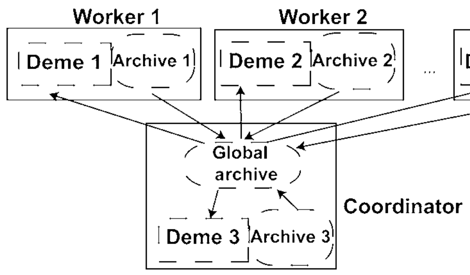

Fig. 2 Illustration of multiple population version of ε-NSGA-II designed for small homogeneous clusters adapted from Tang et al. [2].

will replace members of receiving populations. For example a greedy search would employ frequent migrations of several individuals with a best replaces worst selection/replacement policy.

Cantu-Paz [5] and Ferringer et al. [9] both highlight that designing and parameterizing MP schemes typically requires enumerative trial-and-error analysis, which may have to be repeated for MOP formulation changes or new applications. Both MP versions of the ε-NSGA-II have been designed to minimize trial-and-error analysis and limit parameterization challenges. The MP_SC version of the ε-NSGA-II exploits global and local epsilon-dominance archives adaptively to control deme sizes and solution migration with minimal user inputs. The MP_SC scheme exploits epsilon-dominance archiving when selecting and storing migrant solutions. The epsilon-dominance archives ensure that large numbers of well distributed solutions can be exchanged without adversely impacting demes' search capabilities.

The MP_SC scheme uses dynamic messaging between processors to facilitate asynchronous population sizing and search dynamics. Following the same basic algorithmic steps illustrated in Fig. 1, the MP_SC version of the ε-NSGA-II initiates search on *W* processors, each of which possess a randomly generated deme of 12 members. Run 1 then proceeds on each processor for 250 generations (the maximum epoch). After completing run 1, worker processors send requests to the coordinator for a copy of the current global archive. As all processors exchange solutions via the global archive, the MP version of the ε-NSGA-II has a fully connected topology and an epsilon-dominance-based selection/replacement policy for migration. The coordinator processor assembles the global epsilon-dominance archive from all of the local processors' archives. Each worker is allowed to complete their current generation before sending their local archives to the coordinator. Once collected, the global archive is then used to adapt deme sizes on all processors. The auto-adaptation of deme sizes follows the same strategy as the serial version of the ε-NSGA-II: 25% of the new populations are composed of global archive members and 75% of the new individuals are generated randomly.

After run 2 is initiated on each processor, their individual migration rates and frequencies are determined based on each deme's search progress. A deme can request the global archive if a) it fails to improve 10% of its local epsilon dominance archive members in 10 generations or b) it reaches its maximal search epoch of 250 generations. Asynchronous search continues until the user-specified maximum wall-clock time is reached. Once the termination criterion is reached, the coordinator sends termination messages to all of the workers.

4. Multiple Population Implementation for Large Clusters

The MP_LC version of the ε-NSGA-II minimizes processor idle time by enhancing the coordinator processor's ability to handle dynamic, asynchronous messaging with larger processor counts and heterogeneous resources. For small homogeneous clusters, the dual use of a processor as both the coordinator and a worker helps to exploit the system's limited computational resources. For larger heterogeneous clusters, the benefits of using the coordinator processor for search decrease rapidly as interprocessor synchronization becomes more challenging. In the MP_SC scheme, population sizing requests can cause processors to be idle while waiting for other processors to complete their current generation. Often the demes on the fastest processors will make the most frequent population sizing requests. This idle time effect is compounded when the coordinator also has to complete its current generational search before processing requests.

In the MP_LC scheme, the coordinator processor is solely dedicated to maintaining the global archive and facilitating archived-based migrations. Workers send their local archives to the coordinator after every generation to reduce the synchronization problems that can occur with large numbers of population-sizing requests. These changes balance synchronization costs vs communication costs. As the computational demands of a real-world application increase, synchronization issues have a larger impact on message-based parallel performance. By sending local archives after every generation, the coordinator in the MP_LC scheme actively maintains the global archive and can share solutions as needed for adapting deme-sizes on a large number of heterogeneous processors.

III. Computational Experiment

The computational experiment used to evaluate the MS_SC, MS_LC, MP_SC, and MP_LC versions of the ε-NSGA-II represents a cross-disciplinary collaboration to better understand real-world operational use of pMOEAs. The MS_SC and MP_SC versions of the ε-NSGA-II serve as the control for this computational experiment. These versions of the algorithm were originally developed for small, homogeneous clusters and tested on applications in the water resources application domain [2]. This is the first study to test the ε-NSGA-II for satellite constellation design [9] from the aerospace domain. As noted in Sec. I, this computational experiment has been designed to test if the alternate parallel versions of the ε-NSGA-II can a) maximize search performance given a constrained wall-clock period for search, b) limit trial-and-error analysis for parameterization and algorithmic design, c) reduce random seed variability for the run-time dynamics and end-of-run results, and d) scale well on large, heterogeneous clusters.

A. Operators, Parameterization, and Random Trials

This study used a binary representation for the satellite constellation design problem to enable enumeration of the decision space and quantification of the true Pareto optimal solution set. The decision variable resolution (7 bits for I and 4 bits for Ω and M) were selected such that an exhaustive search could be completed in a reasonable amount of time. The inclination I was resolved with a longer bit string because previous work [9] has shown that the MRT and ART objectives are highly sensitive to this decision variable. Permutation analysis of the decision variables (after duplicate configurations are removed) resulted in 3,171,952 function evaluations. The full resolution Pareto optimal set is composed of 262 points. The epsilon-nondominated reference set used to evaluate search performance in this study was computed using epsilon settings of 0.1 min (or 6 s). As both objective functions attain values that are the same order of magnitude. These epsilon settings reduced the full resolution 262 solution Pareto optimal set to 20 epsilon nondominated reference solutions. From an application perspective, the epsilon settings were selected because objective differences below 6 s are not meaningful. As the dynamical models used during the objective function evaluations do not model physical reality perfectly. The epsilon-nondominated reference set is used in this study to comprehensively compare the parallelization schemes for the ε-NSGA-II using a metrics-based evaluation of run-time dynamics and end-of-run results.

Table 1 provides a summary of the parameters specified for the binary operators used in all algorithm runs. The operators and parameters used in this study are based on the previous parametric sensitivity analysis and recommendations of Ferringer et al. [9] for the satellite constellation benchmark problem. Prior real-world applications in both

Table 1 Summary of parameters for the MS and MP schemes

Parameter	Setting
Initial population sizes	12
Objective precisions used in reference set	0.1 min, 0.1 min
Epsilons for archiving	0.1 min, 0.1 min
Epsilons for metric calculations	0.1 min, 0.1 min
Problem For two-point crossover	1.0
ProbLem for jump mutation	0.2
Wall-clock time for run termination	12 h

the water resources and aerospace domains [2,9,11,12] have found that population size and run duration have the largest impact on search performance for the NSGA-II and the ε-NSGA-II. The four parallel configurations of the ε-NSGA-II were evaluated using 50 random seed trial runs, each of which terminated after 12 h of wall-clock time. In the MP-schemes' trial runs, each deme has its own unique random seed. The overall rationale in this experiment is that parallelization provides the opportunity to perform more search (i.e., more function evaluations) for increasing processor counts and a fixed wall-clock time for termination. In both the aerospace and water resources domains, operational use of pMOEAs would typically require users to provide the best answer possible within constrained time periods.

B. Fellowship Cluster Configuration

The construction of the Fellowship high-performance computing cluster was primarily motivated by the need to solve computationally expensive problems for a diverse set of users at The Aerospace Corporation. Figure 3 shows how the Fellowship cluster has evolved from a cluster of eight dual Intel Pentium III systems.

Currently, the Fellowship cluster contains six servers, 352 dual-processor nodes, a network switch, and a suite of support software [32]. The servers provide job scheduling, shell access, directory service for user accounts, shared temporary file storage, network boot services, and backups. Table 2 provides a summary of the Fellowship cluster's hardware resources. Both nodes and compute cores share a private nonrouted network and run the Free Berkeley Software Distribution (BSD) 6.2 operating system, which is derived from the AT&T UNIX Time-Sharing System. A Cisco Catalyst 6509 switch and Gigabit Ethernet connects all nodes. The Fellowship cluster is a shared corporate resource that manages job scheduling with the Sun Grid Engine (SGE) batch queuing system.

The SGE scheduler uses a first-in-first-out policy wherein the job submitted first will be attempted to be scheduled first. The SGE searches for unoccupied nodes and schedules the execution when the profile requirements are satisfied. If the first job fails to find a suitable free resource then the SGE will attempt to schedule the next job. When the system is fully loaded, node selection proceeds by using the first available nodes that satisfy the requirements profile for a given job. When there are multiple suitable combinations of nodes (this occurs when the cluster is not 100% used) then SGE uses system load information on the machines to select the least loaded nodes for the jobs.

The distribution of the Fellowship cluster's core count given in Table 2 is used by SGE to schedule jobs on different processor groups. The processor groups with the highest compute core counts get the largest number of jobs. For each of the four algorithm variants (MS_SC, MS_LC, MP_SC, MP_LC) a total of ten W-core job (where W is the number of processors required) were submitted to the SGE. Each of the jobs consisted of five random seed trials. The overall likelihood that SGE selected members of the processor groups listed in Table 2 for executing the 40 submissions for each W-core job is based on their frequency in the overall distribution of processing cores. Any

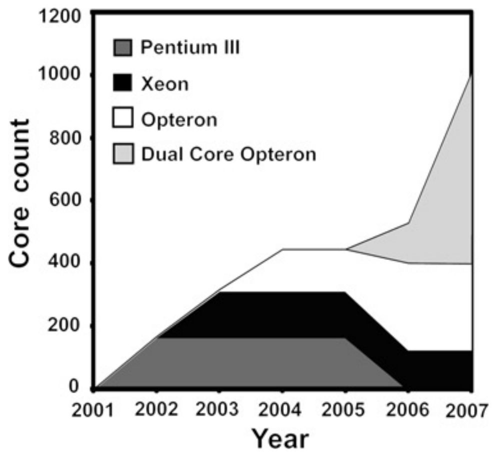

Fig. 3 Temporal changes in the Fellowship high-performance cluster's processor types and core count since its initiation.

Table 2 Fellowship node details

Node count	Type	Speed (GHz)	RAM (GB)	Core count
64	Intel Xeon	2.4	1	128
64	Operton 244	1.8	2	128
24	Operton 246	2	2	48
68	Operton 246	2	2	136
116	Opteron 270	2	2	464
32	Opteron 275	2.2	2	128

particular assignment of processors by the SGE for any given run will generally match the overall core distribution for the Fellowship's resources. Replicating the core count distribution ensures that every processor speed is used and that processor groups with the high core counts complete the largest number of jobs.

C. Evaluation Framework

The evaluation of the parallelization schemes tested in this study is decomposed into a) intra-paradigm analysis of the small and large cluster versions of the MS and MP schemes and b) inter-paradigm analysis of the MS and MP versions of the ε-NSGA-II developed in this study. The SC versions of the ε-NSGA-II are used as the controls in the intra-paradigm analysis to demonstrate that the LC changes proposed in this study improved their performance on the Fellowship cluster. As part of the intra-paradigm analysis, the epsilon-performance metric [12,13] is used to demonstrate run-time dynamics and speedup for MS and MP versions of the ε-NSGA-II. The epsilon-performance was selected because the trends demonstrated were reflective of other run-time measures and the metric captures both diversity and convergence concerns for approximation sets.

To support inter-paradigm analysis a tabular summary of end-of-run results are presented for Deb's diversity and convergence metrics [33], Kollat and Reed's epsilon-performance metric [12], and Zitzler et al.'s epsilon indicator and hypervolume metrics [34,35]. Readers interested in detailed descriptions for these metrics should reference their original citations. The metrics are calculated from 50-random seed trials for each parallelization scheme tested across 1, 2, 4, 8, 16, 32, and 64 processor counts. The 64-processor count was considered sufficient to demonstrate large-scale parallelization given the severe computational demands posed by the 1400 trial runs used in this study (each of which required 12 h of dedicated wall-clock time) as well as the Fellowship cluster's queue demands.

For all of the metrics used in this study, the Mann–Whitney test [36] was used to distinguish the statistical significance of performance differences and to clarify the algorithmic rankings. Statistical tests were performed when comparing two separate parallelization schemes and when comparing a single parallelization scheme's performance for increasing processor counts. The null hypothesis for the Mann-Whitney tests assumed that the two input distributions being compared are identical. The p-values attained from the test where used to judge the significance of the differences in metric distributions and in judging which version of the ε-NSGA-II was superior. Only the Mann–Whitney results for comparing parallelization schemes will be shown in Sec. IV. As increasing processor counts always yielded improved performance at greater than the 95% confidence level in all cases.

IV. Results

Section IV.A compares the MS_SC and MS_LC versions of the ε-NSGA-II to demonstrate how time-continuation and asynchronous evolution can enhance search dynamics and parallel speedups on the Fellowship cluster. Section IV.B compares the MP_SC and MP_LC schemes' performances in terms of run-time dynamics, processor idle time, and speedup dynamics. Section IV.C presents an inter-paradigm analysis that focuses on the end-of-run results for the MS and MP versions of the ε-NSGA-II.

A. Search Dynamics for Master Slave Variants

The success rates of the MS_SC and MS_LC schemes are shown in Fig. 5 (see later) in the form of cumulative distribution functions. Within a given wall-clock time, the distribution functions define the percent of 50 trials that were able to meet or exceed an epsilon performance metric value of 0.8. Figure 4 and Table 3, show that an epsilon-performance metric threshold value of 0.8 is a very challenging performance goal for the MS_SC and MS_LC

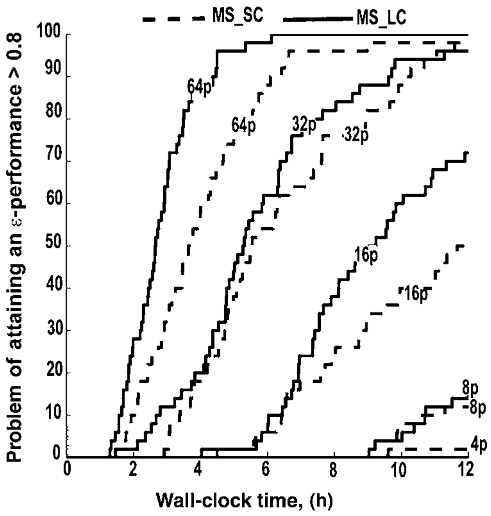

Fig. 4 Dynamic success rates for the MS_SC and MS_LC versions of the ε-NSGA-II. The cumulative probabilities are computed as the percentage of 50 trials that were able to attain an ε-performance value of 0.8 or better for a given wall-clock time and processor count.

schemes. In Fig. 4, the slope or steepness of each distribution provides a visual measure of the run-time variance across random seed trials. For example, a vertical distribution would signify that all 50 trials have identical run times (i.e., random seed independence). The serial and two processor results are not shown in Fig. 4. As all of these runs failed to satisfy the 0.8 performance threshold in 12 h of wall-clock time.

The epsilon-performance metric was selected to illustrate search dynamics because it captures both approximation set convergence and diversity [12,13]. Additionally, the metric is attractive because it has a very intuitive interpretation where the 0.8 performance threshold requires that 80% of the reference set is found within the user specified epsilons (see Table 1). Success rate plots for the other metrics showed very similar trends. For increasing processor counts, the MS_LC scheme has significantly better time efficiency and success rates relative to the MS_SC scheme. Generational synchronization and processor heterogeneity combine to dramatically reduce the effectiveness of the MS_SC scheme as the processor count increases. The MS_LC configuration differs from the MS_SC scheme by not using the master processor for function evaluations and in its use of asynchronous evolution. From an operational use perspective, the MS_LC scheme exploits increased processor counts to maximize search performance in minimum wall-clock periods. As an example, shifting MS_LC's processor count from 32 to 64 processors enables a success rate greater than 95% in approximately half the wall-clock time. At the 64 processor count, the MS_SC scheme requires approximately 10 h to attain a 95% success rate and its run-time variation is significantly higher.

Figure 5 provides a more detailed analysis of processor idle time for the MS_LC and MS_SC schemes. In the MS_SC configuration of the ε-NSGA-II, the overall percentage of wall-clock time spent evaluating designs ranges from nearly 100% for small processor counts to approximately 50% for 64 processors. This result explains why 95% success rates for the 64-processor MS_SC runs take nearly the same wall-clock time as the 32-processor MS_LC runs in Fig. 4. A detailed analysis of processor idle times for both MS schemes showed that communication and processor core heterogeneities (demands, speed, architecture, and so on) contributed on average less than 20 s per run. For the MS_SC scheme, generational synchronization contributed up to 6 h of processor idle time for increasing processor counts. The MS_LC scheme has minimal synchronization issues, increasing its potential for effective speedups [30].

The MS_SC and MS_LC speedup performance in Fig. 6 carefully considers solution quality and speedup jointly. Earlier studies have highlighted that any parallel EA study reporting speedups must explicitly guarantee that the wall-clock timings are for the same level of solution quality [1,2,5]. In pMOEA studies assessing solution quality is challenging. As users want both convergence to the Pareto optimal set as well as a diverse approximation to the full extent of its tradeoffs. This study follows the approach of Tang et al. [2] in reporting speedup and MOEA metrics

Table 3 End of run results for all metrics

Processor count	Scheme	Conv. ($\times 10^{-3}$) Ideally = 0 AVG	STD	Div. Ideally = 1 AVG	STD	Eperf. Ideally = 1 AVG	STD	Eind. Ideally = 0 AVG	STD	Hyper. Ideally maximized AVG	STD
4P	MS_SC	3.14	2.06	0.856	0.05	0.585**	0.10	0.541	0.15	3357	5.08
	MS_LC	3.87	2.91	0.870	0.06	0.554	0.08	0.532	0.12	3357	4.63
	MP_SC	3.00++	1.46	0.870++	0.05	0.612++	0.08	0.527++	0.13	3358++	4.25
	MP_LC	4.53	2.75	0.895	0.05	0.514	0.09	0.651	0.27	3354	6.50
8P	MS_SC	2.27	1.49	0.888	0.04	0.710	0.09	0.367	0.15	3363	3.11
	MS_LC	**1.65****	**1.30**	**0.891**	**0.05**	**0.746****	**0.08**	**0.336**	**0.14**	**3364****	**3.48**
	MP_SC	2.29	1.33	0.891	0.05	0.688	0.09	0.432++	0.15	3362++	3.58
	MP_LC	2.62	1.48	0.870	0.05	0.659	0.08	0.524	0.21	3360	5.39
16P	MS_SC	1.38	0.95	0.924	0.03	0.827	0.08	0.259	0.14	3366	2.20
	MS_LC	**0.77****	**1.11**	**0.919**	**0.03**	**0.861****	**0.07**	**0.218**	**0.12**	**3367****	**2.12**
	MP_SC	1.96	1.09	0.902	0.04	0.746	0.07	0.339	0.14	3364	2.47
	MP_LC	1.63	1.10	0.896	0.05	0.783	0.08	0.351	0.26	3364	5.39
32P	MS_SC	0.65	0.63	0.932	0.02	0.923	0.05	0.154	0.09	3368	1.42
	MS_LC	**0.32****	**0.39**	**0.926**	**0.02**	**0.937**	**0.05**	**0.128****	**0.07**	**3368****	**0.67**
	MP_SC	1.09	0.76	0.915	0.03	0.863	0.06	0.221	0.13	3367	2.15
	MP_LC	0.77++	0.79	0.926++	0.03	0.909++	0.06	0.151++	0.07	3368++	1.34
64P	MS_SC	0.39	0.46	0.937	0.03	0.960	0.05	0.112	0.06	3368	0.68
	MS_LC	**0.27****	**0.27**	**0.931**	**0.01**	**0.983****	**0.03**	**0.093****	**0.03**	**3368****	**0.46**
	MP_SC	0.49	0.59	0.925	0.02	0.936	0.05	0.123	0.07	3368	0.91
	MP_LC	0.50	0.56	0.926	0.02	0.939	0.06	0.138	0.09	3368	1.38

The averages (AVG) and standard deviations (STD) for the convergence (Conv), diversity (Div), epsilon performance (Eperf), epsilon indicator (Eind), and hypervolume (Hyper) metrics were computed using 50 random trials. The best overall parallelization scheme at each processor count is highlighted in underlined bold if Mann–Whitney inter-paradigm tests yield at least 95% confidence levels for a majority of the metrics.
**Superior metric values for MS schemes based on intra-paradigm comparisons that yield at least a 95% confidence level.
++Superior metric values for MP schemes based on intra-paradigm comparisons that yield at least a 95% confidence level.

jointly. Figure 6 uses the epsilon performance metric due to its ability to measure both convergence and diversity. The speedup results show that for low processor counts the MS_LC version of the ε-NSGA-II is similar and sometimes inferior to the MS_SC scheme. The MS_LC configuration distinguishes itself in terms of speedup for processor

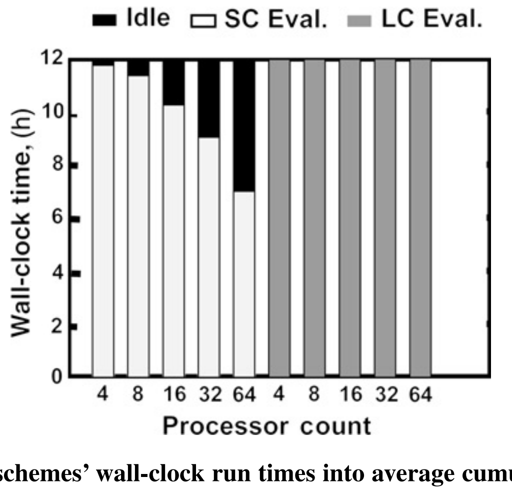

Fig. 5 Decomposition of the MS schemes' wall-clock run times into average cumulative processor idle time vs the average time spent performing function evaluations. The cumulative averages were computed for each processor count using 50 trial runs of the MS_SC and MS_LC versions of the ε-NSGA-II.

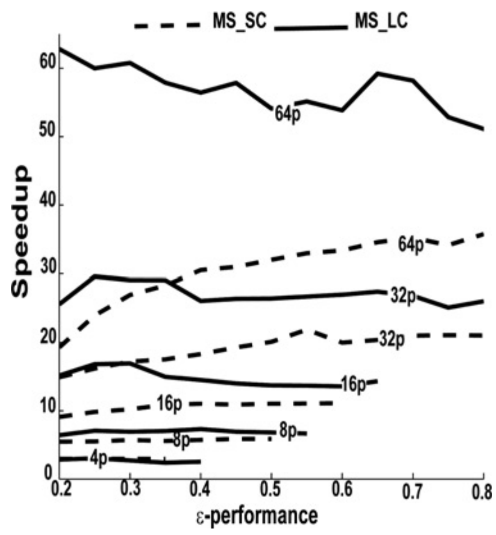

Fig. 6 Average speedup vs solution quality for the MS_SC and MS_LC versions of the ε-NSGA-II. Speedups were computed as the ratio of the average serial wall-clock time on the fastest fellowship processors vs the average parallel wall-clock time required to attain each level of the ε-performance. Speedups are shown only when all 50 trial runs attained a given value of the ε-performance.

counts greater than 16. Figure 6 clearly demonstrates that both MS schemes exploit wall-clock speedup to increase search evaluations and approximate a high percentage of reference set solutions. End-of-run parallel efficiencies for the MS_LC configuration are typically 80% or higher. The epsilon performance speedup results are similar to those found using the other run-time metrics in Table 3.

B. Search Dynamics for the Multiple Population Variants

Figure 7 shows success rates plots for the MP versions of the ε-NSGA-II. Again, the success rate plots show time-based cumulative distribution functions that define the percent of 50 trials that were able to meet or exceed an epsilon performance metric value of 0.8. Figure 7 shows that both MP schemes struggle to attain the performance

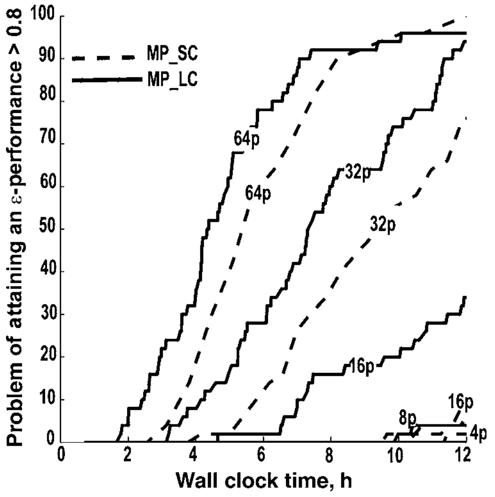

Fig. 7 Dynamic success rates for the MP_SC and MP_LC versions of the ε-NSGA-II. The cumulative probabilities are computed as the percentage of 50 trials that were able to attain an ε-performance value of 0.8 or better for a given wall-clock time and processor count.

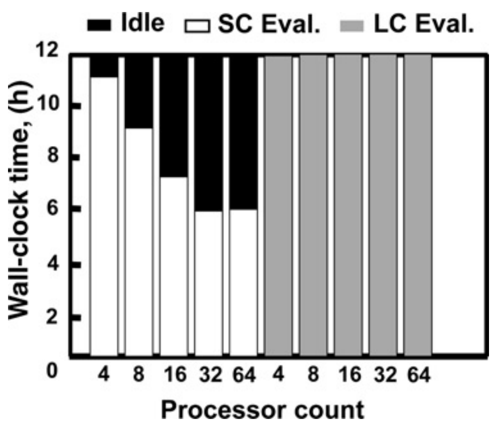

Fig. 8 Decomposition of the MP schemes' wall-clock run times into average cumulative processor idle time vs the average time spent performing function evaluations. The cumulative averages were computed for each processor count using 50 trial runs of the MP_SC and MP_LC versions of the ε-NSGA-II.

threshold for low processor counts. The MP_SC scheme is superior to the LC implementation for small processor counts (<16), which makes sense given that Tang et al. [2] designed this version of the ε-NSGA-II for small clusters. When using small processor counts the MP_SC scheme's cumulative idle times are small (Fig. 8) and allowing the coordinator processor to perform search in tandem with worker processors maximizes search progress.

As processor counts increase Figs. 7 and 8 show that the benefits of allowing the coordinator to perform search rapidly decline while synchronization issues become the dominant control of search success, allowing the MP_LC scheme to distinguish itself. Results for the MS and MP schemes demonstrate the SC implementations are often more effective for small processor counts. Generalizing pMOEA performance to scale well on large, heterogeneous systems requires a concomitant reduction in performance when using small numbers of processors. Comparison of the success rate plots in Figs. 4 and 7 shows that in general the MP schemes have a much higher variation in wall-clock time and they require substantially more wall-clock time to achieve high success rates. The 64-processor runs for MP_LC scheme require approximately twice the wall-clock time as the 64-processor MS_LC runs to attain a 90% success.

Figure 8 shows that the MP_SC version of the ε-NSGA-II does have slightly higher processor idle times relative to the MS_SC version of the algorithm. Both the MS_LC and the MP_LC schemes successfully limit processor idle time and maximize the time spent performing function evaluations as shown in Figs. 5 and 8. The asynchronous, dynamic archive-based migration strategies used in the MP_LC and MP_SC strategies are quite effective and strongly limit trial-and-error design analysis. Table 3 shows that the end-of-run metrics for the MP schemes are comparable to the MS-based results. Figures 7 and 9 show that the MP-schemes' limited speedups required them to attain high quality solutions using fewer design evaluations. Although improved search utilizing fewer function evaluations is a major benefit, redundancy in the MP-schemes search and their reduced speedups combine to limit their effectiveness in terms of actual wall-clock time, which is the key consideration for parallelization efforts [7]. Another interesting observation in Fig. 8 is that the 32 and 64 processor cases for the MP_SC have similar idle times which is counterintuitive. As has been noted in earlier literature, successful MP-schemes must balance the benefits of increasing search diversity with more populations and the communication costs associated with their migration strategies. In the Fig. 8, the 64 processor instance of the MP_SC code has an overall very large, diverse search that permits less migration (less idle time).

C. Analysis of End-of-Run Results for All Configurations

This section focuses on statistically assessing which parallelization scheme has the best end-of-run effectiveness. The four parallelization schemes have been characterized using a total of 1400 trial runs (50 random trials per processor count $\times 7$ processor counts per parallelization scheme $\times 4$ parallelization schemes). For all of the metrics used in this study, the Mann-Whitney test [36] was used to distinguish the statistical significance of performance differences and to clarify algorithmic rankings. Table 3 provides the end-of-run mean values and standard deviations for all

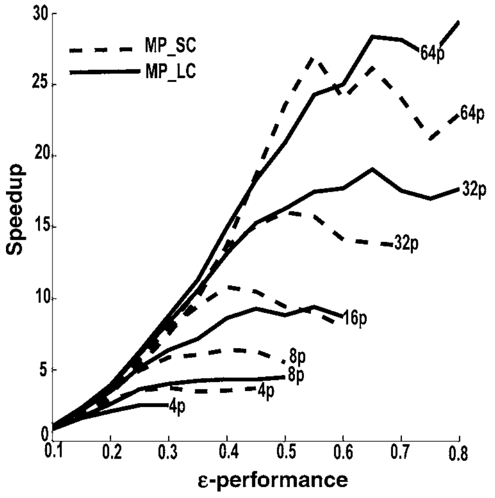

Fig. 9 Average speedup vs solution quality for the MP_SC and MP_LC versions of the ε-NSGA-II. Speedups were computed as the ratio of the average serial wall-clock time on the fastest fellowship processors vs the average parallel wall-clock time required to attain each level of the ε-indicator. Speedups are shown only when all 50 trial runs attained a given value of the ε-performance.

of evaluation metrics used in this study. Results for less than four processors where statistically indistinguishable and were therefore not included in Table 3. The results shown in Table 3 provide a baseline to judge the dynamic performance plots presented in Secs. IV.A and IV.B. In Table 3, the $**$ superscript designates when either the MS_SC or MS_LC versions of the ε-NSGA-II attain superior metric values to at least the 95% confidence level. Likewise for intra-paradigm analysis of the MP schemes, the $++$ superscript designates superior metric values to at least the 95% confidence level. The best overall parallelization scheme at each processor count is highlighted in Table 3 using underlined bold font when Mann–Whitney inter-paradigm tests confirmed superior performance for a majority of the metrics at the 95% confidence level.

Analysis of the metric moments in Table 3 for increasing processor counts, shows that all four parallelization strategies were able to exploit the Fellowship cluster to attain consistent improvement of end-of-run metrics. Increasing processor counts always yielded improved performance at greater than a 95% confidence level in all cases. At the 4 processor count, the MP_SC scheme is significantly superior to the MP_LC configuration as was noted in the prior results shown in Fig. 7. This again confirms that at small processor counts allowing the MP_SC scheme's coordinator processor to perform search in tandem with worker processors maximizes search progress. Table 3 and Fig. 7 show that although MP_LC scheme minimizes processor idle time, its end-of-run results are not substantially different from those of the MP_SC scheme until 32 or 64 processors are used. Problem difficulty and the diversity of the MP schemes search populations appear to play a more important role in their performance than synchronization bottlenecks and speedup.

For the MS schemes, the asynchronous evolution of the LC implementation dramatically improved speedup and allowed the algorithm to perform significantly more search in the 12 h wall-clock period. Table 3 shows that the MS_LC version of the ε-NSGA-II was the best overall performer when using processor counts of 8 or more. The small variances attained by the MS_LC scheme at high processor counts again highlight its seed independence or reliability in solving the benchmark aerospace application as was noted in Fig. 4. Table 3 shows that the MS_SC scheme is the second best performing parallelization scheme overall. The MS_SC scheme is able to attain competitive to superior metric results relative to both MP schemes for all processor counts.

V. Discussion and Conclusions

This study uses a statistical, metrics-based evaluation framework to demonstrate how time-continuation, asynchronous evolution, dynamic population sizing, and epsilon archiving can be used to dramatically enhance both MS and MP versions of the ε-NSGA-II. Comparison of the MS and MP schemes' speedups highlights that MP-based search fundamentally changes evolutionary dynamics as has been routinely recognized in the EA literature [1,5]. Although the MP versions of the ε-NSGA-II have been shown to be effective, their speedup and search dynamics are far more complex than those of the MS schemes. Speedups for both of the MP schemes tested shows a strong increasing trend with improving metric values for search quality. The increasing trends correspond to increases in problem difficulty because it is far more challenging to attain 80% of the Pareto optimal set than it is to attain 20%. This research confirms the results of Tang et al. [2] showing that for extremely challenging problems where a serial algorithm fails, MP-schemes can dramatically enhance search.

This research was motivated by the question, "Should real-world pMOEA applications consider simple MS strategies or complex MP strategies?" This study demonstrates that in many instances, the answer to this question will likely be "simple MS strategies". The large cluster MS implementation of the ε-NSGA-II has been demonstrated to be the best overall parallel scheme tested in this study. The MS_LC scheme maintains diverse search using adaptive population sizing and time-continuation while exploiting asynchronous evolution to minimize processor idle times. By minimizing processor idle time the MS_LC scheme is highly scalable for large, heterogeneous processor counts, maintaining nearly linear speedups over the entire wall-clock period of search.

Search failures for the benchmark satellite constellation design application solved in this study were not the result of severe problem difficulty (which is often the case for studies focused on artificial test functions). If this were the case, the MS schemes analyzed in this work would not have appreciably reduced search failures. The MS schemes were able to reduce search failures using longer active search periods facilitated by the ε-NSGA-II's time-continuation and sustained speedups. This study contributes a detailed example of how to design scalable, MS strategies for large heterogeneous clusters that are capable of attaining statistically superior results relative to high quality MP schemes. Despite the earlier pMOEA literature bias towards MP research, this study highlights that real-world practitioners should first consider MS schemes that exploit time-continuation to maintain diverse search for as long as is feasible or necessary. Scalable MS schemes are easy-to-implement, limit trial-and-error parameterization, and can exploit predictable speedups to enhance the solution of computationally demanding real-world applications.

Acknowledgments

The authors would like to thank Yong Tang for his valuable contributions in the design and development of the SC parallel version of the ε-NSGA-II. All trademarks, service marks, and trade names are the property of their respective owners.

References

[1] Van Veldhuizen, D., Zydallis, J., and Lamont, G., "Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 2, 2003, pp. 144–172.
doi: 10.1109/TEVC.2003.810751

[2] Tang, T., Reed, P., and Kollat, J. B., "Parallelization Strategies for Rapid and Robust Evolutionary Multiobjective Optimization in Water Resources Applications," *Advances in Water Resources,* Vol. 30, No. 3, 2007, pp. 335–353.
doi: 10.1016/j.advwatres.2006.06.006

[3] Holland, J. H., "Building blocks, cohort genetic algorithms, and hyper-plane-defined functions," *Evolutionary Computation*, Vol. 8, No. 4, 2000, pp. 373–391.
doi: 10.1162/106365600568220

[4] Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B., *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York, 2002.

[5] Cantu-Paz, E., *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publishers, Norwell, MA, 2000.

[6] Alba, E., and Troya, J., "Analyzing synchronous and asynchronous parallel distributed genetic algorithms," *Future Generation Computer Systems*, Vol. 17, No. 4, 2001, pp. 451–465.
doi: 10.1016/S0167-739X(99)00129-6

[7] Crowl, L. A., "How to Measure, Present, and Compare Parallel Performance," *IEEE Parallel and Distributed Technology: Systems and Technology,* Vol. 2, No. 1, 1994, pp. 9–25.
doi: 10.1109/88.281869

[8] Donaldson, V., Berman, F., and Paturi, R., "Program Speedup in a Heterogeneous Computing Network," *Journal of Parallel and Distributed Computing,* Vol. 21, 1994, pp. 316–322.
doi: 10.1006/jpdc.1994.1062

[9] Ferringer, M., Clifton, R., and Thompson, T., "Efficient and Accurate Evolutionary Multi-objective Optimization Paradigms for Constellation Design," *Journal of Spacecraft and Rockets,* Vol. 44, 2007, pp. 682–691.
doi: 10.2514/1.26747

[10] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* Vol. 6, No. 2, 2002, pp. 182–197.
doi: 10.1109/4235.996017

[11] Ferringer, M., and Spencer, D., "Satellite Constellation Design Tradeoffs Using Multiple-Objective Evolutionary Computation," *Journal of Spacecraft and Rockets,* Vol. 43, No. 6, 2006, pp. 1404–1411.
doi: 10.2514/1.18788

[12] Kollat, J. B., and Reed, P., "The Value of Online Adaptive Search: A comparison of NSGA-II, ε-NSGAII, and εMOEA," *Evolutionary Multi Criterion Optimization: Third International Conference (EMO 2005)*, Guanajuato, Mexico, 2005, Lecture Notes in Computer Science, Springer-Verlag, pp. 386–398.

[13] Kollat, J. B., and Reed, P., "Comparing State-of-the-Art Evolutionary Multi-Objective Algorithms for Long-Term Groundwater Monitoring Design," *Advances in Water Resources,* Vol. 29, No. 6, 2006, pp. 792–807.
doi: 10.1016/j.advwatres.2005.07.010

[14] Goldberg, D. E., *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic Publishers, Norwell, MA, 2002.

[15] Laumanns, M., Thiele, L., Deb, K., and Zitzler, E., "Combining Convergence and Diversity in Evolutionary Multiobjective Optimization," *Evolutionary Computation,* Vol. 10, No. 3, 2002, pp. 263–282.
doi: 10.1162/106365602760234108

[16] Deb, K., "Scalable test problems for evolutionary multi-objective optimization," Technical Report, KanGAL Report No. 2001001, Indian Institute of Technology 2001.

[17] Deb, K., *Multi-Objective Optimization using Evolutionary Algorithms*, Wiley, New York, 2001.

[18] Walker, J. G., "Some Circular Orbit Patterns Providing Continuous Whole Earth Coverage," Technical Report 70211, Royal Aircraft Establishment, Farnborough, UK, 1970.

[19] Draim, J., "Three and Four-Satellite Continuous Coverage Constellations," *Journal of Guidance, Control and Dynamics,* Vol. 8, No. 6, 1985, pp. 725–730.

[20] George, E., "Optimization of Satellite Constellations for Discontinuous Global Coverage via Genetic Algorithms," *Advances in the Astronautical Sciences*, Vol. 97, August, 1997, pp. 333–346.

[21] Lang, T., "A Parametric Examination of Satellite Constellations to Minimize Revisit Time for Low Earth Orbits Using a Genetic Algorithm," *Advances in the Astronautical Sciences*, Vol. 109, August, 2001, p. 625.

[22] Williams, E., Crossley, W., and Lang, T., "Average and Maximum Revisit Time Trade Studies for Satellite Constellations Using a Multi-Objective Genetic Algorithm," *Advances in the Astronautical Sciences,* Vol. 105, January, 2000, p. 653.

[23] Martin, E., Hassan, R., and Crossley, W., "Comparing the N-branch Genetic Algorithm and the Multi-Objective Genetic Algorithm," *AIAA Journal*, Vol. 42, No. 7, 2004, pp. 1495–1500.
doi: 10.2514/1.756

[24] Mason, W., Coverstone-Carroll, V., and Hartmann, J. W., "Optimal Earth Orbiting Satellite Constellations via a Pareto Genetic Algorithm," Aug. 1998, AIAA Paper 98-4381.

[25] Elitzur, R., Gurlitz, T., Sedlacek, S., and Senechal, K., "*ASTROLIB Users Guide*," Navagation and Geopositioning Systems Department, Systems Engineering Division, Engineering and Technology Group, The Aerospace Corporation, Los Angeles, CA, 1993.

[26] Tang, Y., Reed, P., and Wagener, T., "How efficient and effective are evolutionary multiobjective algorithms at hydrologic model calibration?," *Hydrology and Earth System Sciences,* Vol. 10, 2006, pp. 289–307.

[27] Kollat, J. B., and Reed, P., "A Computational Scaling Analysis of Multiobjective Evolutionary Algorithms in Long-Term Groundwater Monitoring Applications," *Advances in Water Resources,* Vol. 30, No. 3, 2007, pp. 408–419.
doi: 10.1016/j.advwatres.2006.05.009

[28] Bazterra, V. E., Cuma, M., Ferraro, M., and Facelli, J., "A general framework to understand parallel performance in heterogeneous clusters: analysis of a new adaptive parallel genetic algorithm," *Journal of Parallel and Distributed Computing,* Vol. 65, No. 1, 2005, pp. 48–57.

[29] Al-Jaroodi, J., Mohamed, N., Jiang, H., and Swanson, D., "Modeling Parallel Applications Performance on Heterogeneous Systems," *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, 2003, doi:10.1109/IPDPS.2003.1213298.

[30] Alba, E., Nebro, A., and Troya, J., "Heterogeneous Computing and Parallel Genetic Algorithms," *Journal of Parallel and Distributed Computing,* Vol. 62, No. 9, 2002, pp. 1362–1385.
doi: 10.1006/jpdc.2002.1851

[31] Gropp, W., *Using MPI: Portable Parallel Programming with Message-Passing Interface*, 2nd ed., MIT Press, Cambridge, MA, 1999.

[32] Davis, B., AuYeung, M., Clark, J., Lee, C., Palko, J., and Thomas, M., "Reflections on Building a High-performance Computing Cluster using FreeBSD," *BSDCan*, University of Ottawa, 2007, pp. 1–13.

[33] Deb, K., and Jain, S., "Running Performance Metrics for Evolutionary Multi-Objective Optimization," Indian Institute of Technology Kanpur, Kanpur, India, Technical Report 2002004, 2002.

[34] Zitzler, E., and Thiele, L., "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation,* Vol. 3, No. 4, 1999, pp. 257–271.
doi: 10.1109/4235.797969

[35] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Grunert da Fonseca, V., "Performance Assessment of Multiobjective Optimizers: An Analysis and Review," *IEEE Transactions on Evolutionary Computation,* Vol. 7, No. 3, 2003, pp. 117–132.
doi: 10.1109/TEVC.2003.810758

[36] Conover, W. J., *Practical Nonparametric Statistics,* 3rd ed., Wiley, New York, 1999.

Christopher Rouff
Associate Editor