JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 44, No. 3, May–June 2007

Efficient and Accurate Evolutionary Multi-Objective Optimization Paradigms for Satellite Constellation Design

Matthew P. Ferringer,* Ronald S. Clifton,† and Timothy G. Thompson‡
The Aerospace Corporation, Chantilly, Virginia 20151

DOI: 10.2514/1.26747

Multi-objective evolutionary algorithms have been shown to be effective optimization tools to search the complex tradeoff spaces of satellite constellation design. Often, the metrics that make up the design tradeoff require lengthy function evaluation time, resulting in a decreased utility of serial multi-objective evolutionary algorithms. In this research, the authors implement two parallel processing multi-objective evolutionary algorithm paradigms, the master–slave and island models, on a heterogeneous system of processors and operating systems. The efficiency and effectiveness of each approach is studied in the context of a regional coverage design problem. The island scheme outperforms the master–slave model with respect to efficiency. A study of the search dynamics for each paradigm demonstrates that both reliably meet the goals of multi-objective optimization (progressing toward the Pareto-optimal front while maintaining a diverse set of solutions). A key conclusion of this research is that both paradigms provide excellent approximations of the true Pareto frontier using a single seed, and when combined across multiple trial runs, they find nearly the entire set of Pareto-optimal solutions.

Nomenclature

d_r	=	Euclidean distance
e_r	=	Boolean variable
i	=	inclination
j	=	number of processors available for a given simulation
$J2$	=	oblateness term
L	=	decision variable lower bound
M	=	mean anomaly
M_p	=	fastest processor in a heterogeneous system
m	=	total quantity of machines in the heterogeneous system
N	=	population size
n	=	binary bit-string length
n_p	=	total number of processors used by a given machine
P	=	parent population
Q	=	child population
q	=	index for a single machine in a heterogeneous system
R	=	combined parent and child population
T	=	elapsed time for a single simulation
U	=	decision variable upper bound
v	=	number of vectors in the known Pareto front
W_q	=	power weight of a given machine
x	=	decision variable resolution
α	=	number of chromosomes sent to each island
β	=	migration frequency
δ	=	degree of connectivity
Ω	=	right ascension of the ascending node

Presented as Paper 6015 at the AIAA/AAS Astrodynamics Specialist Conference, Keystone, CO, 21–24 August 2006; received 24 July 2006; revision received 27 November 2006; accepted for publication 30 November 2006.

*Member of the Technical Staff, Systems Engineering Division, 15049 Conference Center Drive, Mail Stop CH1-410. Member AIAA.

†Senior Engineering Specialist, Systems Engineering Division, 15049 Conference Center Drive, Mail Stop CH1-410. Member AIAA.

‡Senior Engineering Specialist, Systems Engineering Division, 15049 Conference Center Drive, Mail Stop CH1-410. Member AIAA.

I. Introduction

THE task of designing a satellite constellation requires decision-makers to evaluate potential solutions against mission requirements. When these requirements conflict with each other, there is not a single optimum design, but rather an optimal set of tradeoff solutions. The problem is complicated further by the high dimensionality and rugged landscape characteristic of constellation performance metrics. Optimization problems of this class challenge the effectiveness of traditional approaches (parametric, analytical, etc.). Multi-objective evolutionary algorithms (MOEAs) and their mimicry of the genetic mechanisms underlying natural selection are well suited for solving optimization problems in this domain.

Several researchers have used evolutionary multi-objective optimization (EMO) to approximate the Pareto fronts (solutions to a two-objective optimization problem) for a variety of conflicting metrics [1–7]. Although early efforts incorporated various interpretations of basic evolutionary algorithm principles into algorithm construction, recent efforts [8,9] have used the state of the art in open-source MOEAs produced by the evolutionary computation community to explore complex tradeoffs in the constellation design domain. All of these efforts have demonstrated the potential that MOEAs have to efficiently search the constellation design space and produce human-competitive results; however, the previous work has dealt with relatively small numbers of satellites and short (day-long) propagation times. Real-world applications often call for metrics based on lengthy propagation durations, complex constraints, and an ever-increasing number of satellites. These scenarios can result in extensive function evaluation time, limiting the utility of serial MOEAs. To retain or increase that utility for constellation design problems with computationally demanding function evaluations, parallel processing must be considered.

In this research, the authors implement two parallel processing paradigms: the master–slave and island models that use a heterogeneous system (HS) of processors and operating systems. The efficiency and effectiveness of each approach are studied in the context of a constellation design problem. The remainder of the paper is presented as follows: Sec. II reviews multi-objective optimization and provides a brief discussion of classical and evolutionary computation (EC) methods; Sec. III details the parallel processing paradigm designs and outlines the hardware used in this study; the constellation design test problem and parallel multi-objective evolutionary algorithm (PMOEA) performance metrics are defined in Sec. IV; the results and discussion are presented in Sec. V; and conclusions and recommendations for future research are in Sec. VI.

II. Fundamental Background

A. Multi-Objective Optimization

The solutions to the multi-objective problem are known as Pareto-optimal [10] or nondominated [11] and have the property that no other solution can be found that performs better for all objectives. The design tradeoff may be visualized as a front or a surface (two or three conflicting objectives, respectively) populated with the nondominated solutions. For example, it is desirable to maximize the life of a satellite while minimizing the cost. A design with a long life and a high cost can be Pareto-optimal, along with a solution that is less expensive and short-lived.

The Pareto front offers invaluable information to the decision-maker. In a single chart, the feasible performance bounds are established, and the degree to which the requirements conflict with each other is illustrated. Classically nondominated solutions are found by condensing multiple objectives into a single function for which the optimum is based on the extent of preference information available (weighted sum, ε constraint, weighted metric, Benson's value function, and interactive) [11]. These methods have significant limitations in that they require many simulation runs to approximate the Pareto front, fail to find solutions in nonconvex objective spaces, or require the decision-maker to apply subjective information (such as objective weighting) before performing the optimization. The methodology employed by multi-objective evolutionary computation (MOEC) offers an innovative approach that is not subject to these limitations.

B. MOEAs

There are several state-of-the-art open-source MOEAs [12–14] that may be modified to work on objective functions from a decision-maker's domain of interest. For this study, the nondominated sorting genetic algorithm 2 (NSGA-II) was selected for parallelization, because it has been shown (for most problems) to find a better spread of solutions and improved convergence near the true Pareto-optimal front when compared with its competitors [11]. As a result, the algorithm has been successfully used for a wide array of real-world applications [15–17]. The developer has several options for adapting a MOEA to use distributed processing.

III. Approach

A. PMOEA Paradigm Overview

The four major PMOEA paradigms include master–slave, island, diffusion, and hybrid (a combination of the other three forms) [18]. The master–slave model, as the name suggests, uses a single master processor to execute the MOEA and slave nodes to evaluate individual members of the population. This paradigm has a search-space exploration that is identical to that of a serial MOEA.

Modeled after Darwin's observations [19] of finch diversity from the Galapagos Archipelago, the island paradigm organizes the population into subpopulations called demes [18] that evolve independently, with periodic migration. This paradigm offers additional exploration that may improve convergence and result in the discovery of even better solutions when compared with the master–slave approach.

The reader is encouraged to reference Van Veldhuizen et. al. [20] for a thorough discussion of PMOEA development issues. The remainder of this section details the hardware architectures and paradigm implementations identified for study.

B. Hardware

Researchers and corporations alike are often bound by a limited set of existing resources consisting of a conglomeration of processor types, nodes, memory, operating systems, and communication networks. The paradigms developed for this study are designed to take advantage of current technology that allows simultaneous use of the HS for parallel execution. Heterogeneity can take on several definitions, depending on the resource characteristics. In this paper, HS refers to a collection of machines with varying processor types, number of processors, and/or operating systems connected by a local area network (LAN). All experiments are executed on any single/combination of the three HSs defined in Table 1–3.

Table 1 X86/Solaris 10

Processors	CPU, MHz	Memory, GB
8	2190	32
2	2000	5
2	2000	5
2	2000	5
2	2000	5
2	2000	5

Table 2 Pentium 3/Linux Red Hat 9

Processors	CPU, MHz	Memory, GB
28	1000	0.256
6	1133	0.512

Table 3 3 SPARC/Solaris 9

Processors	CPU, MHz	Memory, GB
1	1015	1
8	1050	16
2	1056	5
1	1062	2
1	1062	2
8	1200	64
1	1200	1
2	1280	4
2	1280	2
2	1280	2
2	1280	2
2	1280	2
2	1280	2
1	1503	3

The network connecting the SPARC and X86 machines has a gigabit Ethernet backbone, with switches capable of supporting 10/100/1000 Mbps. The Pentium 3 processors are organized as a cluster on a subnet, with another Pentium 3 processor acting as a router to the backbone with a 100-Mbps channel. The communication library used is LAM message passing interface (MPI) Version 7.1.1 [21]. LAM/MPI is an open-source implementation of the MPI/MPI-2 standard developed and maintained by the Open Systems Lab at Indiana University.

C. Master–Slave Model

The master–slave model is implemented by executing the NSGA-II on the fastest node available for a given simulation run while evenly distributing population members across each slave. As the slaves finish evaluation of the objective functions, the master collects the results until all population members of a given generation have been processed. This implementation is asynchronous in that as soon as a slave finishes the evaluation of an individual, it receives another population member, until all have been evaluated for a given generation. However, generational synchronization is required before the NSGA-II can execute its evolutionary operators (EVOPs) (crossover, mutation, and crowded tournament selection) and other overhead functions (domination ranking and crowding distance sorting). Pseudocode for this implementation is provided in Algorithm 1. Postinitialization generational flow is shown in Fig. 1.

D. Island Model

The island implementation executes the NSGA-II on each processor while periodically migrating individuals to neighboring demes. Several new parameters include the number of chromosomes to ship to each island, the migration frequency (also known as the

Algorithm 1 Master–slave PMOEA pseudocode

```
Initialize parent population
  Randomly generate parents P_t of N individuals on processor 0
Determine initial parent population members fitness
  Loop until all n individuals in P_t are evaluated
    Ship one chromosome each to processor 1 to j − 1
    Slave processors conduct objective function evaluations
    Receive objective function values from slaves
  Perform nondominated sorting for each member in P_t on processor 0 to
    form domination fronts F
  Calculate crowding distance for each member in P_t on processor 0
Generate offspring population
  Form offspring population Q_t of N individuals on processor 0 via
    EVOPS
  Loop until all N individuals in Q_t are evaluated
    Ship one chromosome each to processor 1 to j − 1
    Slave processors conduct objective function evaluations
    Receive objective function values from slaves
  Determine combined population members fitness
    Perform nondominated sorting for combined population of 2N
      individuals R_t on processor 0
    Calculate crowding distance for each member in R_t on processor 0
  Report top N individuals from R_t to form P_{t+1}
Repeat generate offspring population until terminal generation
Report all nondominated individuals in the final generation on processor 0
```

epoch), the number of neighboring islands δ, and the topology [20]. The value given to each of these will impact the algorithm's efficiency and quality of solution. In an effort to eliminate some of the ad hoc trial-and-error experimentation that is typically required to optimize this set of parameters for a given problem, the authors use several important conclusions taken from parallel genetic algorithm (PGA) design theory developed by Cantu-Paz [18].

For a constant degree of connectivity, determined by δ, differing topologies produce nearly identical solutions (the differences in solution quality are statistically insignificant) [18]. As a result, the authors have designed an arbitrary topology detailed in Table 4 for δ. Figure 2 illustrates mapping for six demes at δ equal to 1 and 3. For example, the topology $+1$, -1, and $+3$ for processor 2 migrates individuals to processors 3, 1, and 5. An exception to this topology exists when combinations of migration parameters and topology mapping result in multiple migrations to identical processors. In this case, each duplicate migration is redirected to the nearest nonrepeating processor.

The island implementation, from the perspective of a single processor, is described with pseudocode in Algorithm 2. The NSGA-II executes independently on each island, forming offspring populations from the parent populations using EVOPs. Migrants are selected by randomly picking parent pairs to participate in crowded tournament selection. Each winning chromosome is selected only once, until the number of winners equals the desired number of migrants, at which time they are cloned and sent to other demes

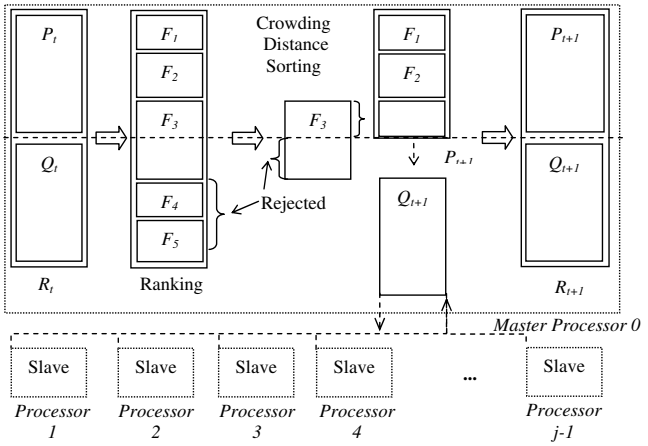

Fig. 1 Master–slave generational flow.

Table 4 Topology mapping

Connectivity (number of neighbors δ)	Topology
0	0
1	$+1$
2	$+1, -1$
3	$+1, -1, +3$
4	$+1, -1, +3, -3$
5	$+1, -1, +3, -3, +5, -5$
6	$+1, -1, +3, -3, +5, -5, +7$
7	$+1, -1, +3, -3, +5, -5, +7, -7$

Algorithm 2 Island PMOEA pseudocode

```
On each island
  Initialize parent populations
    Randomly generate parent population P_t of N individuals on each
      processor
  Determine initial parent population members fitness
    Loop until all N individuals in P_t are evaluated
    Perform nondominated sorting for each member in P_t to form
      domination fronts F
    Calculate crowding distance for each member in P_t
  Generate offspring population
    Form offspring population Q_t of N individuals via EVOPs
    Loop until all N individuals in Q_t are evaluated
  Migrate chromosomes
    Perform crowded tournament selection to form pool of α migrants
    Ship α migrants to the receiving buffer R_t for each of δ islands
  Fitness evaluation
    Perform nondominated sorting for combined population of
      (P_t + 1 + Q_t + 1 + R_t + 1)
    Calculate crowding distance for each member in combined population
    Report top N individuals to form new parent population
  Repeat generate offspring/migrate chromosomes/fitness evaluation until
    terminal generation
  Report all nondominated individuals
Report all nondominated individuals across all islands
```

determined by the neighborhood topology. Migrants received are added to each deme's parent and offspring populations just before ranking and crowding distance sorting.

The execution of the NSGA-II on each processor eliminates generational synchronization required by the master–slave model but introduces a question of how to migrate individuals with minimal impact to efficiency. Migration is implemented asynchronously using a buffering system. Each deme has a receiving buffer capable of accepting shipments of chromosomes from connected islands, such that deme sizes expand at a rate dependent on the heterogeneity of the system. However, the parent population is always reduced to N individuals after ranking and crowding distance sorting.

The generational flow for the deme on processor 0, with a degree of 3 and three chromosomes per shipment, is shown in Fig. 3. Each boat $(X.Y)_Z$ originates on processor X with selected clones of the parent population of generation Y and migrates to processor Z. In the case in which a heterogeneous system of processors is used, the faster processors may ship multiple boats, packets $(2.1)_0$ and $(2.0)_0$, to

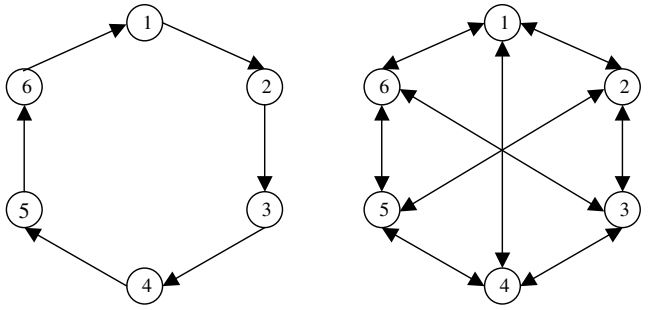

Fig. 2 Left, six deme $\delta = 1$/right, six deme $\delta = 3$.

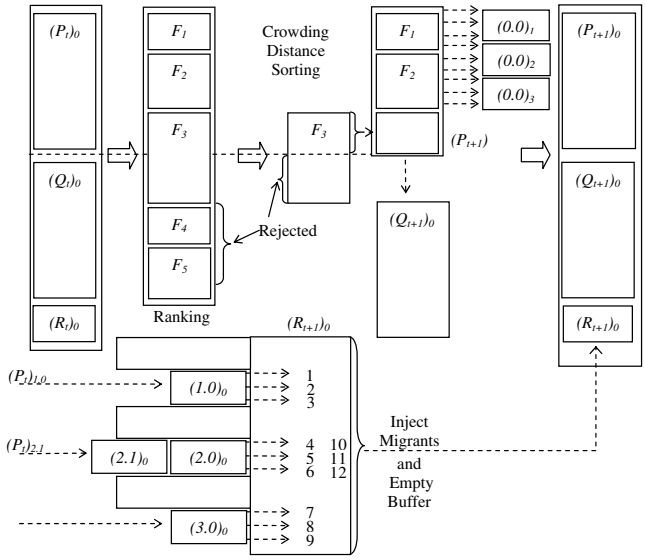

Fig. 3 Island generational flow from the perspective of processor 0.

demes that occupy slower processors before the buffer occupants are combined with P_{t+1} and Q_{t+1}. The buffer is locked during nondominated sorting when R_{t+1}, consisting of individuals 1–12, is combined with P_{t+1} and Q_{t+1}. During this time, incoming chromosomes wait until the buffer is cleared and unlocked, ensuring that individuals are not lost during migration. The simulation terminates when a maximum number of function evaluations, accumulated across all islands, has been reached.

IV. Test Problem and Metric Formulation

The NSGA-II is capable of modeling the decision variables as continuous (real-valued representation) or discrete (binary representation). In both cases, a computational grid is placed over the decision space. A real-valued chromosome has a much finer grid fidelity (only limited by the precision of the computer), giving the representation the ability to reach more points in the design space. A binary representation reduces the size of the search space to a resolution determined by the length of the binary bit strings of each decision variable and allows for a quantitative comparison between the true Pareto front PF_{true} (to a specific resolution), found using an exhaustive search, and the Pareto fronts PF_{known}, found with various PMOEA implementations [22]. The following section details a constellation design test problem, the parametric analysis constructed to determine PF_{true}, and the metrics used to measure PMOEA efficiency and effectiveness.

A. Constellation Design Test Problem

The constellation design test problem is constructed such that an exhaustive search of the decision variables, to some coarse resolution, may be completed with a practical amount of computational effort. Further guidance is found in the literature, in which a well-known multiple-objective tradeoff is analyzed.

George [23] notes, "If your goal is to minimize the maximum revisit time (MRT) that occurs at any point, coverage used to revisit any point more frequently than any other is wasted." George's observation suggests that there is a multiple-objective tradeoff between the sparse coverage metrics of MRT and average revisit time (ART). The MRT, also referred to as the maximum gap interval, is the longest period of time that any ground point in the region of interest remains out of sight of the constellation during the length of the propagation. The ART is defined as the average of the average gap intervals to all points in the region of interest.

The test problem seeks three-satellite constellation designs that minimize both the MRT and ART to a region of discrete points (forming an equal area grid) overlaying the landmass of the conterminous United States. The chromosome consists of a common inclination across all satellites and independent values for the right ascension of the ascending node and mean anomaly for two of the three satellites. Both the Ω and M of the remaining satellite are arbitrarily set to zero. All satellites occupy a constant altitude of 400 n mile (740.8 km) and have horizon-to-horizon visibility. The satellite motion is simulated via a Keplerian propagator with J_2 secular effects for 1 year. The fixed global reference frame for the Earth and physical constants such as radius, rotation rate, gravitation parameter, and flattening are taken from World Geodetic Survey 84 [24].

Coverage analysis software, Revisit-C, is integrated into the NSGA-II to perform the objective function evaluations. Revisit-C, developed by The Aerospace Corporation, uses an aerospace-developed astrodynamics and math library called Astrolib [25] to support a wide range of coverage analysis problems. Revisit-C is written in C to provide seamless integration with NSGA-II. Revisit-C is based on a legacy coverage application, called Revisit, that was written in FORTRAN and has been used for the past two decades to support many of Aerospace's customers. Revisit models satellite visibility to ground locations and computes coverage statistics for multiconstellation, multisensor, multiconstraint satellite architectures. Coverage constraints are applied sequentially to compute a set of constrained visibility intervals for each target. Two types of constraints are modeled in Revisit: availability constraints and visibility constraints. Availability constraints limit when a satellite is available to provide coverage (e.g., relay and ground station constraints). Visibility constraints limit the coverage provided by a satellite (e.g., sun angle and solar exclusion constraints). Targets available in Revisit include individual ground locations, areas, and grids. Performance metrics computed by Revisit include revisit time, response time, access interval duration, daily visibility time, and daily number of accesses.

B. Parametric Analysis

The decision variables are enumerated to conduct a discrete search of the revisit tradeoff space. Propagation time, number of satellites, and number of ground points drive computational cost for a function evaluation. These parameters and the preceding decision-variable step sizes are selected such that the exhaustive search is completed in a reasonable amount of time on an eight-processor AMD Opteron machine, 2190 MHz each, with 32 GB of RAM. Variable resolution with upper bound and lower bound is determined by the length of their binary strings using Eq. (1). The upper and lower bounds for each of the decision variables are defined in Table 5, and the discrete increments for several bit-string lengths are shown in Table 6.

$$x = L + \frac{(U - L)}{2^n - 1} \tag{1}$$

For this study, four bits are used to represent Ω and M, and seven bits represent i. A permutation analysis of the Ω and M decision variables, after duplicate configurations are deleted, yields 24,976 combinations for every i. Each of these combinations is explored for $\Delta i = 1.417$ deg from 0–180 deg, resulting in a total of 3,171,952

Table 5 Decision variable ranges

Bound	i, deg	Ω, deg	M, deg
L	0	0	0
U	180	360	360

Table 6 Decision variable resolution

n	i, deg	Ω, deg	M, deg
4	12.000	24.000	24.000
5	5.806	11.613	11.613
6	2.857	5.714	5.714
7	1.417	2.835	2.835

function evaluations. Taking a smaller step in any of the decision variables results in an impractical number of combinations. For example, adding one bit to Ω and M would increase the total amount of function evaluations required to 58,920,960, rendering an exhaustive parametric search infeasible. Because of these limitations for obtaining a measure of truth, no attempt is made to address scalability issues, and the results and conclusions outlined are valid for the problem size considered. The interested reader is encouraged to reference recent work [26] in the water resources community, in which a computational scaling analysis of a NSGA-II variant is performed.

C. PMOEA Performance Analysis

The no-free-lunch (NFL) theorem [27] suggests that there is a lack of universal metrics that allow for a direct comparison of different paradigm implementations that solve an identical problem. With NFL in mind, the authors analyze the efficiency and effectiveness by giving the same search opportunity to each paradigm. This is accomplished by terminating each simulation after a fixed number of function evaluations. The metrics outlined in what follows provide useful insight into the strengths and weaknesses of each of the PMOEA schemes.

On a homogeneous parallel computing system (defined as a collection of identical processors and operating environments connected by a LAN), speedup is measured as the elapsed simulation time on one processor divided by the time to complete execution on multiple processors. Efficiency is calculated as a ratio of the speedup to the number of processors [20]. These definitions do not accurately measure performance for the HS with varying processor capability and machine configurations.

A more sophisticated model [28] that accommodates the characteristics of the HS can be described as follows. Elapsed time is directly measured for each simulation and used to calculate the power weight [Eq. (2)] of each machine for a given combination of processors. In other words, W_q is the amount of work that q can do relative to the fastest processor in the HS. The speedup [Eq. (3)] is the ratio of the execution time of the fastest processor in the HS to that of the parallel run time. The denominator in the efficiency ratio, given in Eq. (4), represents the ideal speedup by taking into account the W_q of each q in the HS. Because W_q is based on T and given that the simulation runs are executed on shared resources, each T is reviewed before the reported results are accepted.

$$W_q = \frac{\min_{p=1}^{m}\{T(M_p)\}}{T(M_q)} \tag{2}$$

$$SP = \frac{\min_{p=1}^{m}\{T(M_p)\}}{T(\mathrm{HS})} \tag{3}$$

$$E = \frac{SP}{\sum_{p=1}^{m} W_q * n_p} \tag{4}$$

A Perl script was written to monitor the load on each processor during the simulation runs. When a load imposed from a shared user interrupts a processor, the script reports the intrusion and the data for that simulation are discarded. Simulations runs are repeated until there is a minimal level of other activity on each processor. Speedup and efficiency of both the master–slave and island implementations are reported for systems ranging from the nearly homogeneous Linux cluster defined in Table 2 to a combination SPARC, X86, and Pentium 3 machines.

Two accuracy metrics are used to determine how well the algorithms are meeting the goals of multiple-objective optimization [29]. The first goal, to find a wide range of values in the final nondominated set, is evaluated by calculating the error ratio ER. The error ratio [22] defined by Eq. (6) reports the finite number of vectors in $\mathrm{PF}_{\mathrm{known}}$ that are not members of $\mathrm{PF}_{\mathrm{true}}$. The Boolean variable takes the value of 0 if the solution in $\mathrm{PF}_{\mathrm{known}}$ is a member of $\mathrm{PF}_{\mathrm{true}}$ and takes the value of one, otherwise.

$$\mathrm{ER} = \frac{\sum_{r=1}^{v} e_r}{v} \tag{6}$$

The second goal of multi-objective optimization seeks to minimize the distance of the obtained nondominated front to the Pareto-optimal front. Generational distance [30] GD, defined by Eq. (7), reports how far $\mathrm{PF}_{\mathrm{known}}$ is from $\mathrm{PF}_{\mathrm{true}}$. The Euclidean distance between each vector and the nearest member of $\mathrm{PF}_{\mathrm{true}}$ are calculated in the phenotypic (objective) space using Eq. (8).

$$\mathrm{GD} = \frac{\left(\sum_{r=1}^{v} d_r^2\right)^{\frac{1}{2}}}{v} \tag{7}$$

$$d_r = \min(|f_1^r(x) - f_1^q(x)| + |f_2^r(x) - f_2^q(x)|), \qquad r, q = 1, \ldots, v \tag{8}$$

V. Results and Discussion

A. Parametric Analysis

The parametric analysis found 262 Pareto-optimal solutions ($\mathrm{PF}_{\mathrm{true}}$) after exhaustive search of the decision variables (3,171,952 function evaluations). Figure 4 depicts $\mathrm{PF}_{\mathrm{true}}$, along with all parametric solutions that fit within the bounds of the plot window (note that only 3727 of the possible solutions are displayed in the plot window). The figure reveals the discontinuous, rugged nature of the objective function space for this coverage tradeoff.

B. Efficiency

The efficiency analysis uses the NSGA-II parameters outlined in Table 7 for the master–slave and island implementations. Run time, used to calculate speedup, is dependent on the number of generations and population size (master–slave) or a specified terminal number of function evaluations (island). Both models complete a simulation at the end of 10,000 function evaluations. The remaining parameters were set arbitrarily and do not impact efficiency results. Figure 5 illustrates both the observed and ideal speed up for the master–slave implementation on the Pentium 3, SPARC, and X86 HS, respectively.

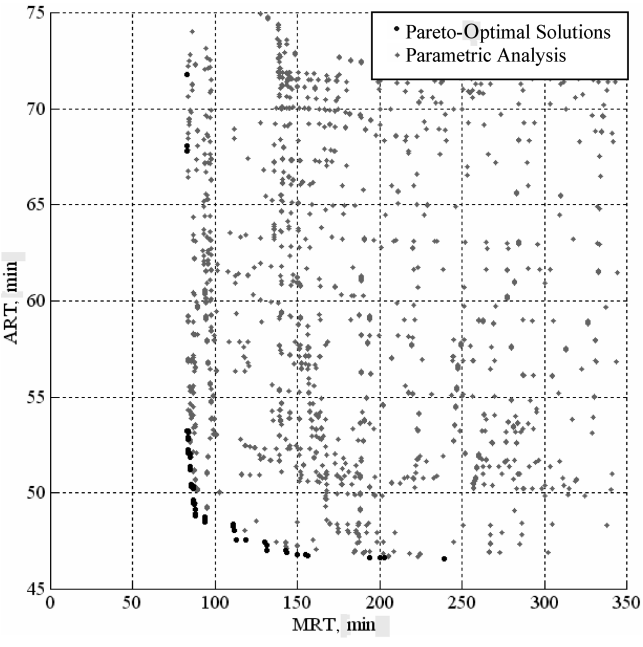

Fig. 4 Parametric analysis and Pareto-optimal solutions.

Table 7 Efficiency analysis PMOEA parameter settings

Parameter	Master–slave	Island
N	100	32
Generations	100	400
Real crossover probability	0.9	0.9
Real mutation probability	0.2	0.2
Crossover distribution index	20	20
Mutation distribution index	20	20
δ	n/a	4
α	n/a	8
β	n/a	1
Terminal function evaluation	n/a	10,000

Table 8 Master–slave efficiency summary

HS	Processors	Efficiency
1. X86	18	88%
2. Pentium 3	34	89%
3. SPARC	37	80%
4. X86 and Pentium 3	52	39%
5. X86 and SPARC	55	61%
6. SPARC and Pentium 3	71	54%
7. SPARC, X86, and Pentium 3	89	44%

1. Master–Slave Model

Each observed speedup curve for the master–slave paradigm exhibits stepwise behavior in which the addition of processors can result in plateaus in speedup. This behavior is most clearly illustrated in Fig. 5a for the nearly homogeneous 34-processor Pentium 3 cluster. Recall that the master–slave implementation requires that all slave processors have returned their evaluations at the end of a generation before EVOPs can be executed on the master processor. Near-ideal speedup is observed for 3, 5, 11, 21, and 26 processors. Because the master node occupies a single processor, these data points have 2, 4, 10, 20, and 25 slave processors, all of which are even factors of the 100 population members used for each simulation. When the total population size divided by the quantity of slave processors has a remainder, speedup inefficiencies result. The most extreme inefficiencies occur just before evenly divisible quantities of slave processors. For example, consider the 25-processor (24-slave) observed data point for the Pentium 3 HS in Fig. 5a. Here, 24 individuals may be evaluated at a given time. Because the system is nearly homogeneous, all function evaluations will be returned at nearly the same time in the following running sum: 24, 48, 72, and 96. After 96 evaluations, 4 remain, but 24 processors are available. At this point, 20 processors are sitting idle and the final 4 individuals are being evaluated to complete a generation. The remaining HS (SPARC and X86) also exhibit this stepwise behavior, but the plateaus are less pronounced and speedup diverges more dramatically as the processor count increases. These inefficiencies are most pronounced with the SPARC architecture, which has the highest degree of heterogeneity of the systems shown in Fig. 5.

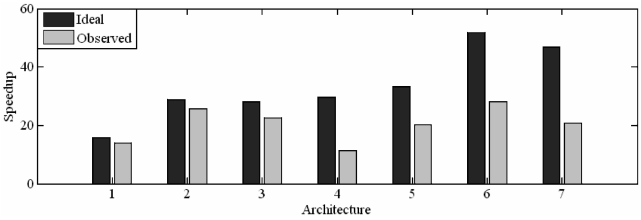

Fig. 6 Master–slave speedup summary.

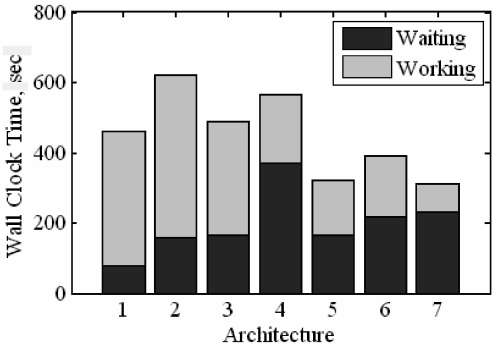

Fig. 7 Master–slave timing summary.

The processing capability shown in Tables 1–3 are combined to define seven HSs: X86 (18 processors); Pentium 3 (34 processors); SPARC (37 processors); X86 and Pentium 3 (52 processors); X86 and SPARC (55 processors); SPARC and Pentium 3 (71 processors); and SPARC, X86, and Pentium 3 (89 processors). A summary of the speedup obtained and resulting efficiency for these HSs are presented in Fig. 6 and Table 8, respectively. Generally, systems with increasing heterogeneity have decreasing efficiency.

Accurate timing scripts are used to generate Fig. 7, in which the amount of time each HS spends waiting for generational synchronization (waiting) to occur is plotted with the time spent executing the NSGA-II and performing function evaluations (working). Further inspection of the output from the scripts indicates that the communication costs of the master–slave implementation are very small when compared with the inefficiencies that result from generational synchronization.

2. Island Model

Recall that the island model uses a buffering system that allows for asynchronous migration. The resulting speedup curves illustrated in Fig. 8 depict observed and ideal speedup for X86, SPARC, and Pentium 3 HSs, respectively. Figure 8 exhibits ideal (and in some cases, superlinear) speedup. The superlinear speedup is not due to the island implementation; rather, it can be accounted for by further consideration of the problem domain. Each constellation design has a function evaluation time that is dependent on the number of accesses generated during the propagation (increasing quantities of accesses

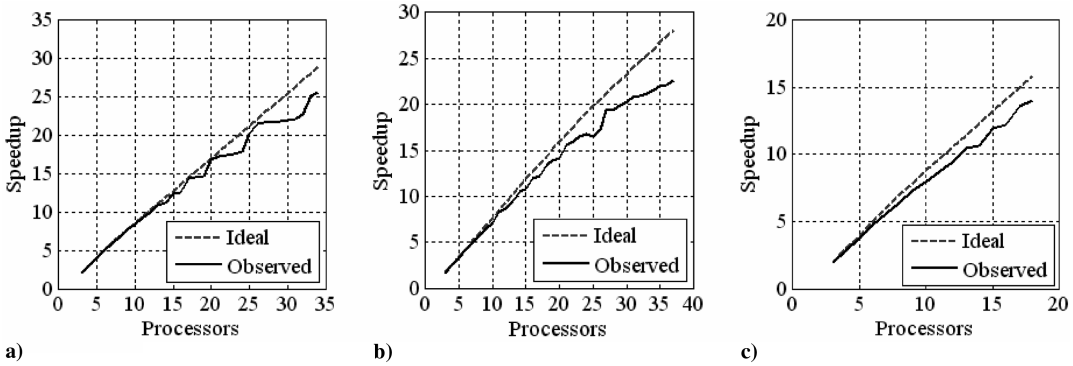

Fig. 5 Master–slave speedup HS a) Pentium 3 HS, b) SPARC HS, and c) X86.

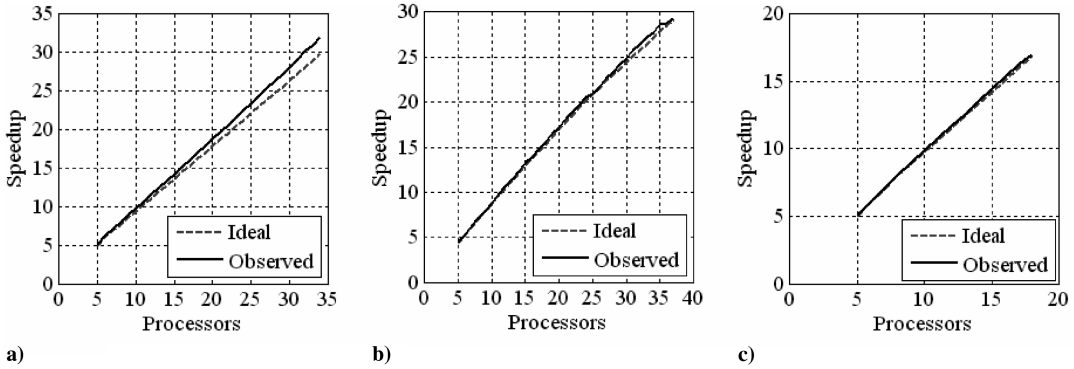

Fig. 8 Island speedup HS a) Pentium 3 HS, b) SPARC HS, and c) X86.

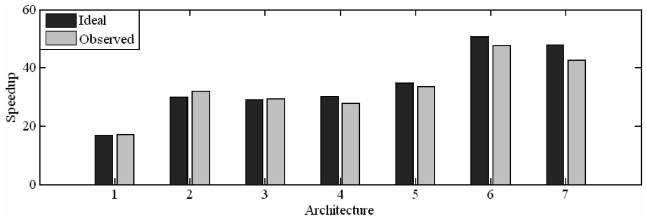

Fig. 9 Island speedup summary.

are a result of a higher-quality solution). For example, consider the 5- and 35-processor Pentium 3 simulations in Fig. 8a. Because all runs are terminated after 10,000 function evaluations, five processors can evolve for many more generations than will the 35-processor configuration. At termination, the designs discovered with 35 processors have fewer accesses than the 5-processor case, resulting in a faster execution time and the observed superlinear speedup. The more important point, however, is that asynchronous island execution eliminates the inefficiencies associated with generational synchronization observed with the master–slave model.

A summary of the island-model speedup and efficiency for each of the seven HSs outlined earlier is presented in Fig. 9 and Table 9. When differing processor types are combined to form HSs 4–7, the observed efficiency slightly deteriorates. The two contributors to these decreases in efficiency are 1) the Pentium 3 based router that connects the Pentium 3 cluster to the backbone, along with the increase in the number of packets shipped as a result of increased island count and 2) the MPI-2 one-sided communication post-start-wait-complete synchronization has a slight scalability problem as the number of islands grow [31].

C. Effectiveness

To decompose the island-model search dynamics, δ, α, and β are given values outlined in Table 10. The first case, island A, represents a half-connected topology and half of the maximum migration rate. Island B and island C parameter settings represent bounding cases for the connectivity and migration rate. Island B is fully connected (every processor is connected to each other) with a maximum migration rate (all population members on each island are shipped at every epoch), whereas island C is minimally connected (each processor is connected to only one other) and has a minimum migration rate (ships only one individual at each epoch). Each of the three island cases was explored with β set to 1 and 100. A β value equal to 1 represents a maximum migration frequency, whereas a β equal to 100 gives individual demes sufficient opportunity to converge before each migration.

Figure 10 presents GD and ER for each island case and master–slave model as a function of evaluation count. For each parallelization scheme, 50 trial runs are used to gain insight into both the search dynamics and reliability. Generally, β set to 100 yields GD and ER performance that is superior to migrating individuals at every generation. These results are consistent with PGA theory [18], which states that migrations occurring before convergence and replacements selected based on fitness will result in increased selection pressure. In Fig. 10b, island A and island B show that a considerable number of trial runs fail to reach near-zero values of GD, indicating preconvergence. Island C has the lowest migration frequency, degree of connectivity, and ships only one chromosome per deme and, as a result, exhibits GD performance similar to the master–slave model. Figure 10a shows a rapid convergence toward PF_{true} with little variation between master–slave and island schemes. These results indicate that decreased migration frequencies increase the reliability of the island model. The ER for both migration frequencies improve with increasing function evaluations; however, the island cases of Fig. 10c demonstrate more reliable performance because they exhibit less variation across trial runs when compared with those of Fig. 10d.

The GD and ER are only calculated from the nondominated solutions at every time step, not from the entire population and therefore do not give a sense of absolute performance. Figures 11 and 12 depict the mean number of nonrepeating true Pareto-optimal solutions found in each population, averaged over 50 trials, for epochs of 1 and 100, respectively. These results indicate how well each approach captures the full extent of the tradeoff. The island model with $\beta = 1$, shown in Fig. 11, is outperformed by the master–slave that finds, on average, 19% of PF_{true}. As illustrated by Figs. 10b and 10d, Fig. 11 confirms that islands A and B are rapidly preconverging to the smallest set of Pareto-optimal solutions of all cases studies. The decrease in selection pressure due to island C migration parameters allows for the discovery of more true Pareto-

Table 9 Island efficiency summary

HS	Processors	Efficiency
1. X86	18	101%
2. Pentium 3	34	107%
3. SPARC	37	100%
4. X86 and Pentium 3	52	92%
5. X86 and SPARC	55	96%
6. SPARC and Pentium 3	71	94%
7. SPARC, X86, and Pentium 3	89	89%

Table 10 Effectiveness analysis PMOEA parameter settings

Parameter	Master–slave	Island A	Island B	Island C
N	300	40	40	40
Generations	400	n/a	n/a	n/a
Binary crossover probability	1.0	1.0	1.0	1.0
Binary mutation probability	0.2	0.2	0.2	0.2
δ	n/a	4	7	1
α	n/a	5	5	1
Terminal function evaluation	n/a	120,000	120,000	120,000

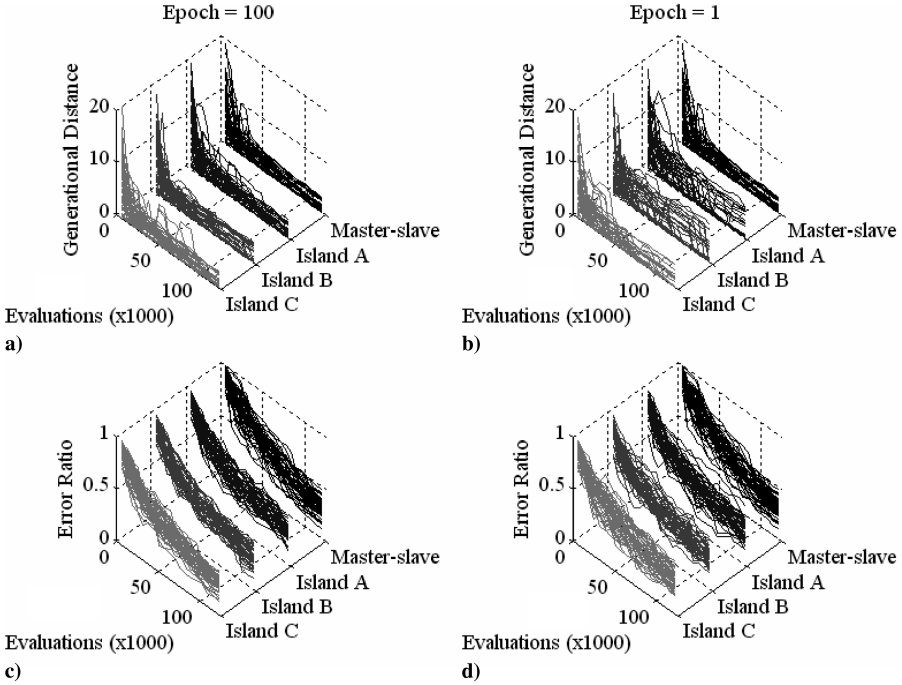

Fig. 10 Dynamic performance plots: a) ER at Epoch of 100, b) ER at Epoch of 1, c) GD at Epoch of 100, and d) GD at Epoch of 1.

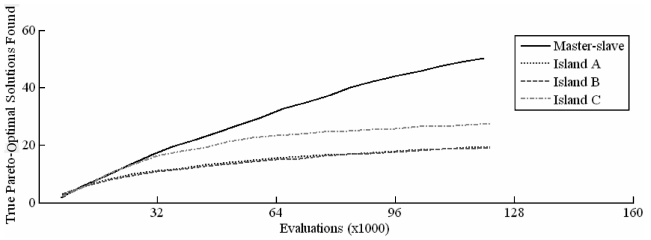

Fig. 11 True Pareto-optimal solutions found for $\beta = 1$.

optimal solutions, but preconvergence is again exhibited. The vertical gridlines of Fig. 12 are drawn at each migration (at increments of 32,000 function evaluations resulting from $\beta = 100$). During the first and second epoch, all island cases, on average, find slightly more true Pareto-optimal solutions than the master–slave. After the second epoch, the master–slave overtakes each island case.

Close inspections of the individuals that make up the combined population across all demes for island cases where $\beta = 1$ reveal that there are a large number of repeating nondominated solutions. These duplicates are a result of the following: 1) binary representation (coarse search space relative to real encoding); 2) migration of cloned individuals; 3) rapid dispersion of fit, cloned individuals in subsequent generations. In other words, each island with $\beta = 1$ mostly converges toward identical populations early in the search, resulting in inferior absolute performance when compared with the master–slave.

The practical utility of EMO for constellation design offers an essential perspective on the effectiveness of each paradigm.

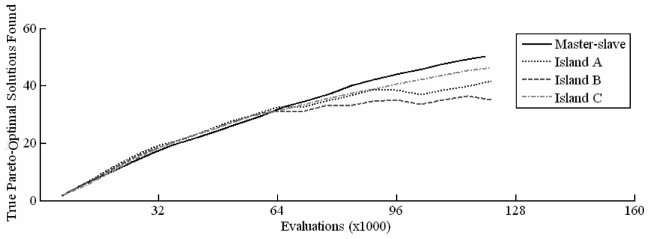

Fig. 12 True Pareto-optimal solutions found for $\beta = 100$.

Figure 13 and 14 illustrate PF_{true} plotted against PF_{known} for both the resulting front when solutions are combined and domination sorted across all 50 seeds, and a single trial run, respectively. The single-simulation PF_{known} for each paradigm is selected by choosing a seed that produced the average quantity of true Pareto-optimal solutions from Fig. 12 at run termination. The resulting approximated Pareto front represents a typical result from a single simulation. Island C is represented in Fig. 14 because it is the most competitive with the master–slave with respect to the quantity of true Pareto-optimal solutions found at run termination. The combined master–slave front (Fig. 14) contains 202 nondominated designs, of which 193 are Pareto-optimal (73% of PF_{true}), and the combined island front contains 214 Pareto-optimal designs (81% of PF_{true}) and 224 nondominated solutions. Although single-seed fronts (Fig. 13) for both paradigms exhibit only 19% of the true Pareto-optimal solutions, they provide excellent approximations of PF_{true} and meet the goals of multi-objective optimization: to find a set of nondominated solutions well spread across the entire Pareto frontier and to minimize the distance between the nondominated designs and PF_{true}. Figure 15 illustrates the effect of combining trial runs across seeds on absolute performance. A significant increase in the quantity of true Pareto-optimal solutions found is evident, and only a marginal improvement in absolute performance is obtained beyond 20 seeds.

VI. Conclusions

Parallel computing with MOEC gives decision-makers a powerful tool to search the complex tradeoffs of constellation design with an efficiency and effectiveness not achievable with traditional methods. The authors show that existing computational resources can be combined into the HS and used with a PMOEA to achieve highly efficient processor use. A significant loss in efficiency is observed with the master–slave paradigm due to generational synchronization. Choosing the number of slave processors to be a multiple of the population size results in near-ideal speedup for mostly homogeneous systems, but performance degrades significantly when used on systems with a high degree of heterogeneity. The island model's efficiency scales well as the number of processors used for a simulation increases.

Both paradigms perform well with respect to meeting the goals of multi-objective optimization. Of all the island parameters, the migration rate had the greatest impact on search and selection pressure. When the demes were allowed to evolve independently

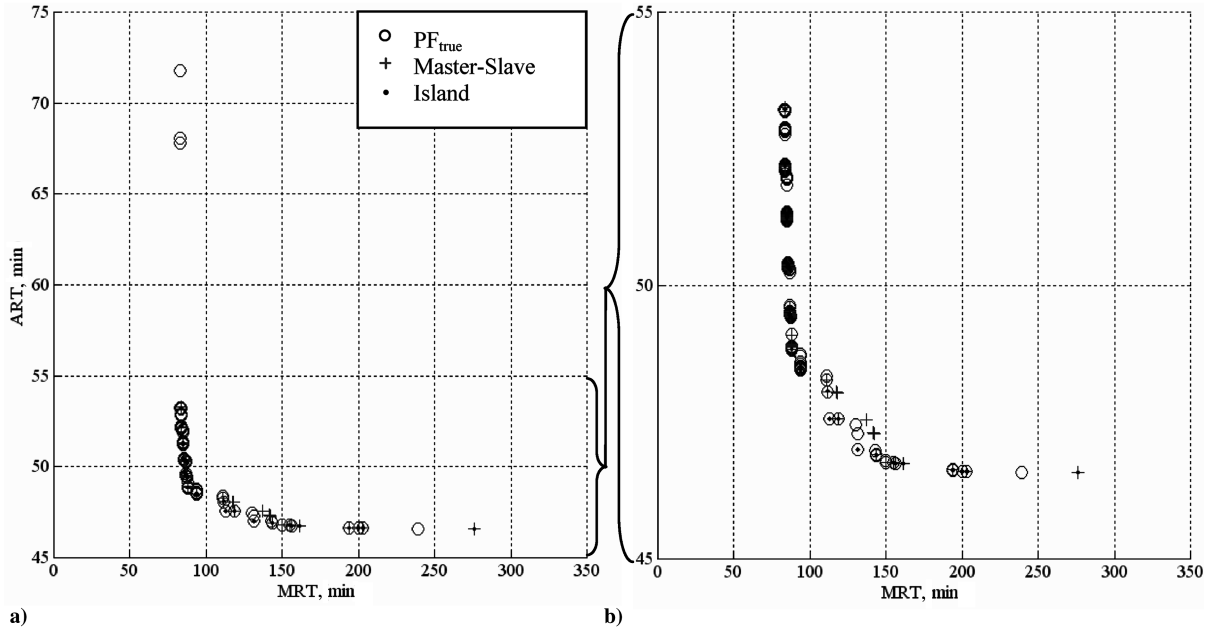

Fig. 13 PF_{true} and PF_{known} a) for a single simulation and b) zoomed view.

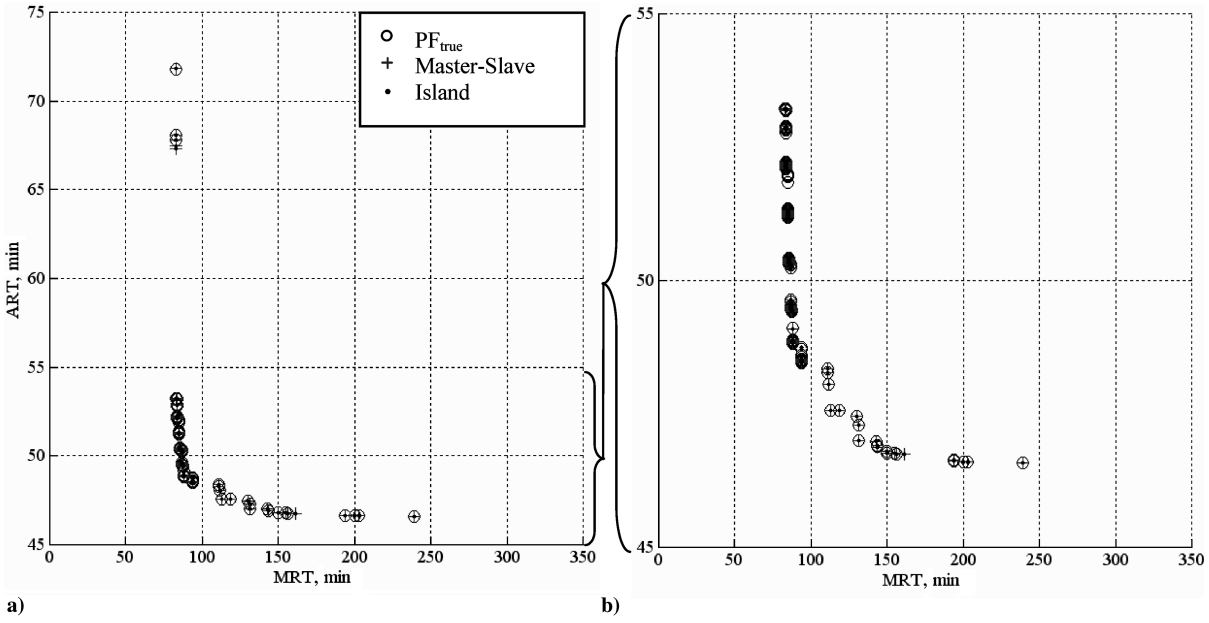

Fig. 14 PF_{true} and PF_{known} a) combined across all seeds for each paradigm and b) zoomed view.

before migrating individuals, the island paradigm exhibited similar search dynamics and competitive absolute performance with respect to finding true Pareto-optimal solutions when compared with the master–slave model. From a practical use perspective, both the master–slave and island models approximate the true Pareto front remarkably well with a single seed. When nondominated solutions are combined and sorted across multiple seed trials, both models find nearly complete representations of the true Pareto frontier.

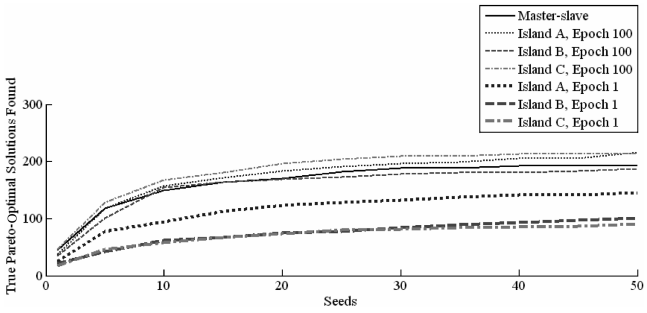

Fig. 15 True Pareto-optimal solutions found as a function of seed quantity.

Acknowledgments

The authors acknowledge The Aerospace Corporation Research and Development Program Office for the financial means to complete this research. Thanks to Tim Meisenhelder, Matthew Jones, Jim Cowan, Tom Gurlitz, Ted Muelhaupt, Tom Gallini, Marilee Wheaton, and John Fujita for their efforts. The master–slave and island models developed for this research are the parallel multi-objective evolutionary algorithm components of The Aerospace Corporation's Genetic Resources for Innovation and Problem Solving (GRIPS) software. GRIPS is an integration of the state of the art in evolutionary computation paradigms (genetic programming, genetic algorithms, and multi-objective evolutionary algorithms).

The authors are grateful to Peter Palmadesso and David Jansing for their contributions to GRIPS.

References

[1] Ely, T. A., Crossley, W. A., and Williams, E. A., "Satellite Constellation Design for Zonal Coverage Using Genetic Algorithms," *Advances in the Astronautical Sciences*, Vol. 99, Feb. 1998, p. 443.

[2] Confessore, G., Di Gennaro, M., and Ricciardelli, S., "A Genetic Algorithm to Design Satellite Constellations for Regional Coverage," *Operations Research Proceedings 2000*, Springer, Berlin, 2001, pp. 35–41.

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

[4] Lang, T. J., "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, Aug. 2001, p. 625.

[5] Asvial, M., Tafazolli, R., and Evans, B. G., "Non-GEO Satellite Constellation Design with Satellite Diversity Using Genetic Algorithm," AIAA Paper 02-2018, May 2002.

[6] Asvial, M., Tafazolli, R., and Evans, B. G., "Genetic Hybrid Satellite Constellation Design," AIAA Paper 03-2283, Apr. 2003.

[7] Martin, E. T., Hassan, R. A., and Crossley, W. A., "Comparing the N-branch Genetic Algorithm and the Multi-Objective Genetic Algorithm," *AIAA Journal*, Vol. 42, No. 7, July 2004, pp. 1495–1500.

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

[9] Ferringer, M. P., and Spencer, D. B., "Satellite Constellation Design Trade-Offs Using Multiple-Objective Evolutionary Computation," *Journal of Spacecraft and Rockets*, Vol. 43, No. 6, Nov.–Dec. 2006, pp. 1401–1411.

[10] Pareto, V., *Manuale di Economia Politica*, Societa Editrice Libraria, Milano, Italy, 1906; also *Manual of Political Economy*, MacMillan, New York, 1971 (in English).

[11] Deb, K., *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, Chichester, England, U.K., 2001, Chaps. 1–6.

[12] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, Apr. 2002, pp. 182–197.

[13] Knowles, J., and Corne, D., "Approximating the Non-Dominated Front Using the Pareto Archived Evolution Strategy," *Evolutionary Computation*, Vol. 8(2), 2000, pp. 149–172.

[14] Zitzler, E., Laumanns, M., Thiele, L., "SPEA-2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," *Evolutionary Methods for Design, Optimization, and Control*, International Center for Numerical Methods in Engineering, Barcelona, Spain, 2002, pp. 95–100, .

[15] Seungwon, L., von Almen, P., Fink, W., Petropoulos, A., and Terrile, R., "Comparison of Multi-Objective Genetic Algorithms in Optimizing Q-Law Low-Thrust Orbit Transfers," *Genetic and Evolutionary Computation Conference (GECCO 2005)*, Late-Breaking Paper, Association for Computing Machinery, New York, 2005.

[16] Xu, K., Louis, S., and Mancini, R., "A Scalable Parallel Genetic Algorithm for X-ray Spectroscopic Analysis," *Genetic and Evolutionary Computation Conference (GECCO 2005)*, Vol. 1, Association for Computing Machinery, New York, 2005, pp. 811–816.

[17] Kollat, J. B., and Reed, P. M., "Comparing State-of-the-Art Evolutionary Multi-Objective Algorithms for Long-Term Groundwater Monitoring Design," *Advances in Water Resources* (to be published).

[18] Cantu-Paz, E., *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic, Boston, 2000, Chaps. 4–7.

[19] Darwin, C., *Journal of Researches into the Natural History and Geology of the Countries Visited During the Voyage Round the World of H.M.S. Beagle*, 11th ed., John Murray, London, 1913, p. 404.

[20] Van Veldhuizen, D. A., Zydallis, J. B., and Lamont, G. B., "Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 2, Apr. 2003, pp. 144–173.

[21] Anon., "LAM/MPI Parallel Computing," *LAM/MPI Source Code* [online database], http://www.lam-mpi.org, [retrieved 1 June 2006].

[22] Coello Coello, C. A., Van Veldhuizen, D. A., Lamont, G. B., *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic/Plenum, New York, 2002, pp. 155–159.

[23] George, E., "Optimization of Satellite Constellations for Discontinuous Global Coverage via Genetic Algorithms," *Advances in the Astronautical Sciences*, American Astronautical Society Paper 97-621, Aug. 1997.

[24] Anon., "Department of Defense World Geodetic System 1984, Its Definition and Relationships with Local Geodetic Systems," 3rd ed., National Geospatial-Intelligence Agency, Rept. TR8350.2, June 2004.

[25] Elitzur, R., Gurlitz, T. R., Sedlacek, S. B., and Senechal, K. H., "ASTROLIB Users Guide," Navigation and Geopositioning Systems Department, Systems Engineering Div., Engineering and Technology Group, The Aerospace Corp., Technical Operating Rept. 93-(3917-1), Los Angeles, 1993.

[26] Kollat, J. B., and Reed, P. M., "A Computational Scaling Analysis of Multiobjective Evolutionary Algorithms in Long-Term Groundwater Monitoring Applications," *Advances in Water Resources* (to be published).

[27] Wolpert, D. H., and Macready, W. G., "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 1, Apr. 1997, pp. 67–82.

[28] Al-Jaroodi, J., Mohamed, N., Jiang, H., and Swanson, D., "Modeling Parallel Applications Performance on Heterogeneous Systems," *International Parallel and Distributed Processing Symposium*, Inst. of Electrical and Electronics Engineers Computer Society, Los Alamitos, CA, 2003.

[29] Knowles, J., and Corne, D., "On Metrics for Comparing Nondominated Sets," *Proceedings of the 2002 Congress on Evolutionary Computation*, Vol. 1, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2002, pp. 711–716.

[30] Van Veldhuizen, D. A., and Lamont, G. B., "Evolutionary Computation and Convergence to a Pareto Front," *Genetic Programming 1998*, Morgan Kaufman, San Francisco, CA, pp. 221–228.

[31] Rajeev, T., William, G., and Brian, T., "Optimizing the Synchronization Operations in MPI One-Sided Communication," *International Journal of High Performance Computing Applications*, Vol. 19, No. 2, 2005, pp. 119–128.

C. Kluever
Associate Editor