AIAA JOURNAL
Vol. 41, No. 9, September 2003

Quality-Assisted Multi-Objective Multidisciplinary Genetic Algorithms

S. Gunawan,* S. Azarm,† and J. Wu*
University of Maryland, College Park, Maryland 20742
and
A. Boyars‡
U.S. Naval Surface Warfare Center, Indian Head, Maryland 20640

A new method is presented to solve multi-objective multidisciplinary optimization (M-MDO) problems. This M-MDO method is applicable to multi-objective optimization problems that can be decomposed hierarchically into multi-objective subproblems and whose objective functions are either separable or additively separable. In the decomposition, the subproblems may have both common and unique objectives. The method uses a multiobjective genetic algorithm (MOGA) to optimize the multi-objective subproblems; hence, it is referred to as a multi-objective multidisciplinary genetic algorithm (M-MGA). It is shown that for any Pareto point of the original (single-level) problem, M-MGA generates at least one point that is noninferior with respect to that Pareto point. Also a comparison is shown between the computational complexity of M-MGA and a single-level MOGA in terms of number of functions calls. The M-MGA is demonstrated by two engineering examples: the design of a speed reducer and the design of a payload for an undersea autonomous vehicle. In both examples, the generated solutions are similar to solutions generated by solving the examples as single-level problems. M-MGA produces relatively the same solutions from one M-MGA run to another.

Nomenclature

AC	=	accuracy metric
CL	=	cluster metric
DH	=	diameter of undersea autonomous vehicle (UAV) hull
$\boldsymbol{f}$	=	design objectives vector
$f_{j,i}$	=	ith objective of the jth subproblem
F_N	=	number of function calls
$\boldsymbol{g}$	=	design constraints vector
$\boldsymbol{g}_j$	=	local constraints vector of the jth subproblem
HD	=	hyperarea difference metric
HM	=	UAV hull material
$I1$	=	first inner material type
$I2$	=	second inner material type
itr	=	iteration counter
J	=	number of subsystem subproblems
K	=	total number of constraints
K_j	=	number of local constraints in the jth subproblem
M	=	total number of objectives
M_j	=	total number objectives in the jth subproblem
N	=	total number of variables
N_j	=	number of local variables in the jth subproblem
NP^*	=	set of points in P^* that are nondominated with respect to all other points in P^*
OS	=	overall spread metric
P^*	=	combination set having lowest HD value
$P_{i,j}$	=	number of Pareto solutions from the jth subproblem associated with x^i_{sh}
PL	=	payload length
pop	=	population size
PR^*	=	Pareto solution of the original problem
P_{Sj}	=	probability of success for the jth scenario
PT	=	payload type
P_0	=	number of feasible system individuals
$\boldsymbol{q}$	=	vector of quality metrics
R	=	number of objectives that are additively separable
rep	=	number of individuals replaced in a MOGA iteration
S	=	number of system objectives
$\boldsymbol{x}$	=	design variables vector
$\boldsymbol{x}_j$	=	local variables vector of the jth subproblem
$\boldsymbol{x}_{\text{sh}}$	=	shared (system) variables vector

Subscripts

L	=	lower bound
U	=	upper bound
0	=	system

I. Introduction

ONE way of solving a large or multidisciplinary design optimization problem is to decompose the problem into smaller subproblems and then solve the decomposed problem by an approach known as multidisciplinary optimization (MDO).[1] In MDO, the decomposed optimization problem consists of two or several levels of subproblems, with the topmost level called the system level and the lower levels called the subsystem levels.

Depending on the couplings between same level subproblems, MDO problems are categorized into two types[2]: hierarchical and nonhierarchical. A hierarchical MDO problem has only parent–children relationships between subproblems: a lower level (child) subproblem is coupled only to its upper level (parent) subproblem. A nonhierarchical MDO problem is more general, allowing couplings between same-level children subproblems as well. In this paper we consider only hierarchical problems. Overviews of MDO methods for aerospace system design optimization can be found by Balling and Sobieszczanski-Sobieski[2] and Sobieszczanski-Sobieski and Haftka.[3]

Many hierarchical MDO methods are reported in the literature, for example, Kirsch,[4] Lasdon,[5] and Azarm and Li.[6] Most of those methods, however, are applicable only to single-objective problems.[2,3]

Received 28 August 2002; revision received 21 January 2003; accepted for publication 19 February 2003.

*Graduate Research Assistant, Department of Mechanical Engineering.
†Professor, Department of Mechanical Engineering; azarm@eng.umd.edu. Member AIAA.
‡Engineer, Underwater Division, Weapons Department.

Even those that can handle multi-objective MDO (M–MDO) problems require that the problems have continuous variables with differentiable objective and constraint functions.[7] Often, M–MDO problems are solved by converting them into single-objective MDO problems using a weighting method or other methods, for example, Haimes et al.[7] and Tappeta and Renaud.[8] Such methods can obtain only a single multi-objective optimization solution, that is, a single Pareto solution, from each run of an optimizer. To obtain a reasonable representation of the complete set of solutions, such M–MDO methods have to be run numerous times.

Several methods have been developed to solve MDO problems that have mixed continuous–discrete variables. Korngold and Gabriele[9] used branch-and-bound and simulated annealing techniques in their method to solve such MDO problems that have mixed continuous–discrete design variables. However, their method is a single-objective MDO method. Kurapati and Azarm[10] introduced a genetic-algorithm-based method to account for the discrete variables in their M–MDO problems. However, their method requires that all subproblems have the exact same objectives, which is not the case with most real-world M–MDO problems.

To address the aforementioned shortcomings, we present in this paper a new method to solve hierarchical M–MDO problems using a multi-objective genetic algorithm (MOGA). This new method, multi-objective multidisciplinary genetic algorithm (M–MGA), allows different subproblems of the M–MDO problem to have unique as well as common multiple objectives. In addition because M–MGA uses a MOGA for the solution of the subproblems, it is applicable to M–MDO problems with a mix of continuous and discrete variables and with nondifferentiable objective and constraint functions.

II. Background

In this section, we present some definitions and terminology and brief descriptions of MOGA and set quality metrics. Note that all of the definitions and terminology given are for a "minimization" problem. A "maximization" problem can be converted to a minimization problem without loss of generality.

A. Definitions and Terminology

A general multi-objective optimization problem can be mathematically stated as follows:

Minimize

$$\boldsymbol{f}(x) = \{f_1, \ldots, f_M\}^t \qquad (M \geq 2) \tag{1a}$$

subject to

$$g_k(\boldsymbol{x}) \leq 0, \qquad k = 1, \ldots, K$$

$$\boldsymbol{x}_L \leq \boldsymbol{x} \leq \boldsymbol{x}_U \tag{1b}$$

where

$$\boldsymbol{x} = \{x_1, \ldots, x_N\}^t \tag{1c}$$

For simplicity of formulation, without loss of generality, we did not include equality constraints.

Typically, the design objectives of a multi-objective optimization problem as shown in Eq. (1) are at least partly conflicting. For this reason, in general, a multi-objective optimization problem does not have a single optimal solution. Instead, there exists a set of optimal solutions for such a problem. This set is called a Pareto set, and its components are called Pareto optimal solutions (definitions follow).

For a minimization problem, we provide definitions of terms used in this paper:

The variable space is the N-dimensional space wherein the coordinate axes are the design variables. The objective space is the M-dimensional space wherein the coordinate axes are the design objectives. The good point is the decision maker's estimate of the utopian[11] point. The bad point is the decision maker's estimate of the nadir[11] point. The explanation of inferior point, noninferior points, and noninferior region follow. A feasible design point $\boldsymbol{x}_a$ is said to be inferior with respect to another feasible design point $\boldsymbol{x}_b$ if

$$f_i(\boldsymbol{x}_b) \leq f_i(\boldsymbol{x}_a), \qquad \forall i = 1, \ldots, M \tag{2}$$

with a strict inequality for at least one i. Correspondingly, the design point $\boldsymbol{x}_b$ is said to dominate $\boldsymbol{x}_a$. If $\boldsymbol{x}_a$ neither dominates nor is inferior to $\boldsymbol{x}_b$, then $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are said to be noninferior with respect to each other. In the objective space, a region in which all of the points within the region are noninferior with respect to every member of a set of noninferior points is said to be a noninferior region, for example, the shaded rectangles in Fig. 1c are the noninferior regions of the set of noninferior points shown with a small triangle symbol.

For a population of design points, a feasible design point $\boldsymbol{x}_d$ is called a nondominated solution point if it is not inferior with respect to any other feasible design points in the population. A feasible design solution point $\boldsymbol{x}_p$ is called a Pareto solution if it is not inferior with respect to any other feasible design points.[11] The set of all Pareto solutions is called the Pareto set. The set of nondominated solution points at the completion of an optimization run is a discrete approximation of the Pareto set.

B. MOGA

MOGA is a genetic algorithm (GA)[12,13] extended for multi-objective optimization. GA solves a single-objective optimization problem by emulating the natural evolution process in which a population of design points (or individuals) is iteratively evolved to reach an optimum solution. In GA, at each iteration, the probability that an individual is selected to evolve is governed by its fitness value, which is a function of that individual's single objective value and its constraint values. MOGA extends GA's fitness assignment method so that it is applicable to multiple objectives.

The main advantage of GA over conventional gradient-based optimization methods is that it does not require gradient information. Hence, it is applicable to problems with non-differentiable objective and constraint functions with a mix of continuous and discrete variables. Because MOGA is an extension of GA, MOGA also inherits this feature. In addition, unlike many other multi-objective optimization methods, MOGA can obtain (a discrete approximation of) the Pareto solution set all at once.[14,15]

There are many variants of MOGA reported in the literature, for example, Schaffer,[14] Fonseca and Fleming,[15] Srinivas and Deb,[16] and Narayanan and Azarm.[17] (A comprehensive review of different MOGAs is given by Deb.[18]) In M–MGA, we use the MOGA developed by Fonseca and Fleming[15] along with the constraint handling method of Kurapati et al.[19] To avoid unnecessary repetitions, from this point on, we will use the term MOGA to refer to the MOGA of Fonseca and Fleming with the constraint handling method of Kuparati et al.

C. Set Quality Metrics

In M–MGA we use set quality metrics as objectives at the system level. These metrics are scalars that quantitatively measure the "goodness" of a set of points in the objective space. We use the following four metrics[20,21] in M–MGA: hyper area difference, overall spread, accuracy, and cluster. There are two main reasons for using these metrics. One reason, explained in Sec. IV, is that the mapping from the system subproblem to the subsystem subproblems is a point-to-set mapping. The second reason, explained in Sec. VI, is that these metrics, as a group, help to produce diversity[20,21] in the population and help in the convergence of M–MGA.

In the following, we give a brief mathematical description of each of these four metrics. Figure 1 shows the geometrical interpretation of the set quality metrics in a two-objective space. Let $P = \{p_1, p_2, \ldots, p_D\}$ be a set of design points in the objective space. Also let p_g and p_b be the good and bad points, respectively.

The hyperarea difference metric HD, the shaded area shown in Fig. 1a, is defined as the difference between the area (or hyperarea HA, for $M \geq 3$) bounded by p_b and p_g and the area bounded by p_b and the points in P:

$$HD(P) = HA(p_b, p_g) - HA(p_b, p_1, p_2, \ldots, p_D) \tag{3}$$

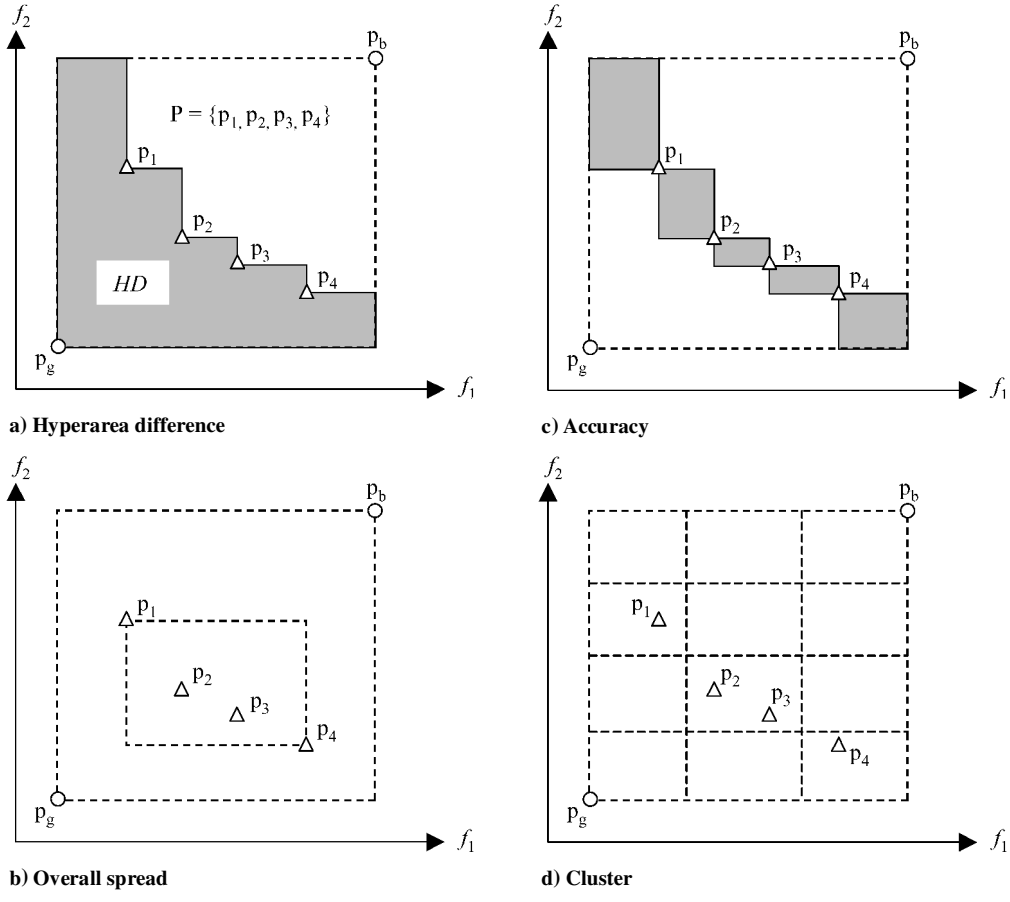


Fig. 1 Geometric interpretation of the set quality metrics.

The overall spread metric *OS* is defined as the ratio between the hyperarea bounded by the two extreme points of P and the hyperarea bounded by p_b and p_g:

$$OS(P) = \frac{HA[\text{extremes}(P)]}{HA(p_b, p_g)} \tag{4}$$

In Fig. 1b, the *OS* is the ratio of the inner rectangular area over the outer rectangular area.

The accuracy metric *AC* of a set P is defined as the inverse of the union of the hyperareas bounded by the pairs of adjacent points in P (shaded rectangles in Fig. 1c):

$$AC(P) = 1/\cup HA[\text{pairs}(P)] \tag{5}$$

The cluster metric *CL* of a set P is defined as the ratio between the total number of points D in the set P and the number of distinct points D_{distinct} of that set (according to a decision maker):

$$CL = D(P)/D_{\text{distinct}}(P) \tag{6}$$

The set P shown in Fig. 1d contains four points, but of the four points, only three are distinct.

III. M–MGA Problem Formulation

In this section, we describe the types of problems that M–MGA can solve and the restrictions that must be satisfied.

Equation (7) formulates the type of multi-objective problems in which M–MGA is applicable. In total, the problem has N design variables, M objectives, and K constraints. Equation (7) reflects two restrictions: the problem must be hierarchically decomposable, and its objectives must be separable or additively separable. We call the equation (7) formulation the original problem:

Minimize

$$\boldsymbol{f}(\boldsymbol{x}) = \left\{ \begin{array}{lll} \sum_{j=1}^{J} f_{j,1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_J); & \ldots; & \sum_{j=1}^{J} f_{j,R}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_J) \\ f_{1,R+1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1); & \ldots; & f_{1,M_1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1) \\ \vdots & & \\ f_{j,R+1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1); & \ldots; & f_{j,M_j}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j) \\ \vdots & & \\ f_{J,R+1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_J); & \ldots; & f_{J,M_J}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_J) \end{array} \right\}^t$$

$$J \geq 1$$

$$0 \leq R \leq M_j, \qquad \forall j = 1, \ldots, J \tag{7a}$$

subject to

$$\begin{array}{ll} \boldsymbol{g}_0(\boldsymbol{x}_{\text{sh}}) \leq \boldsymbol{0}, & \boldsymbol{g}_0 = \{g_{0,1}(\boldsymbol{x}_{\text{sh}}), \ldots, g_{0,K_0}(\boldsymbol{x}_{\text{sh}})\}^t \\ \boldsymbol{g}_1(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1) \leq \boldsymbol{0}, & \boldsymbol{g}_1 = \{g_{1,1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1), \ldots, g_{1,K_1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1)\}^t \\ \vdots & \\ \boldsymbol{g}_j(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j) \leq \boldsymbol{0}, & \boldsymbol{g}_j = \{g_{j,1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j), \ldots, g_{j,K_j}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j)\}^t \\ \vdots & \\ \boldsymbol{g}_J(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_J) \leq \boldsymbol{0}, & \boldsymbol{g}_J = \{g_{J,1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_J), \ldots, g_{J,K_J}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_J)\}^t \\ \boldsymbol{x}_L \leq \boldsymbol{x} \leq \boldsymbol{x}_U & \end{array} \tag{7b}$$

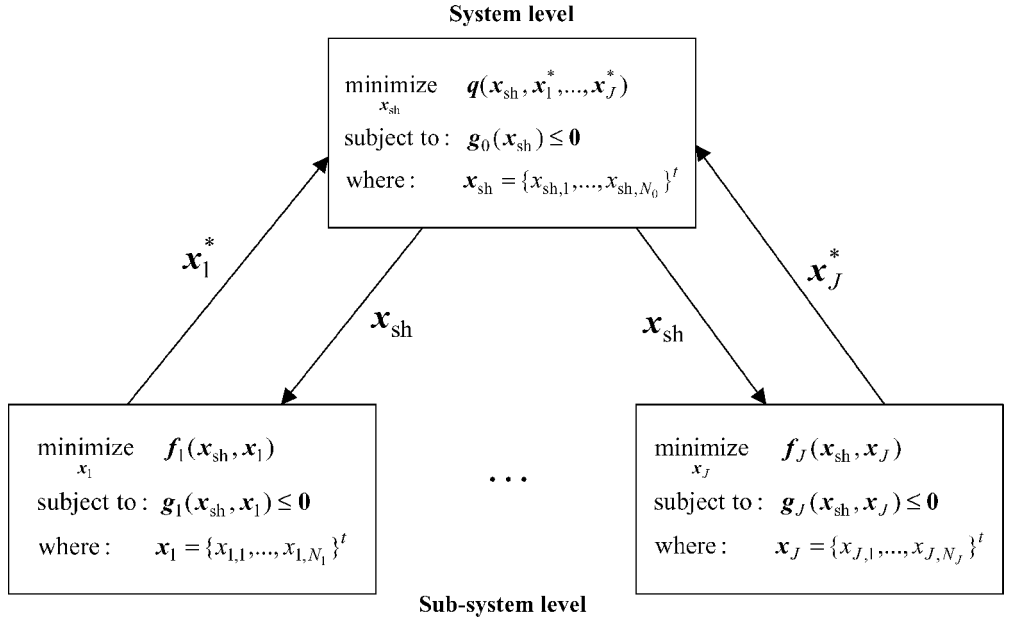


Fig. 2 Hierarchical M–MDO structure of decomposed problem.

where

$$\begin{aligned} \boldsymbol{x} &= \{\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_j, \ldots, \boldsymbol{x}_J\}^t \\ \boldsymbol{x}_{\text{sh}} &= \{x_{\text{sh},1}, \ldots, x_{\text{sh},N_0}\}^t \\ \boldsymbol{x}_1 &= \{x_{1,1}, \ldots, x_{1,N_1}\}^t \\ &\vdots \\ \boldsymbol{x}_j &= \{x_{j,1}, \ldots, x_{j,N_j}\}^t \\ &\vdots \\ \boldsymbol{x}_J &= \{x_{J,1}, \ldots, x_{J,N_J}\}^t \end{aligned}$$

(To simplify the equations, we have not shown the transpose sign t for all vectors.)

In applying M–MGA, we presume that the original problem has been hierarchically decomposed into $J+1$ subproblems, each with its own variables, objectives, and constraints. These subproblems are organized into two levels: one system level subproblem (subproblem 0) and J subsystem subproblems (subproblem 1 to subproblem J). Figure 2 shows the decomposition.

In the decomposed problem, subproblem 0 is optimized with respect to the shared variable vector $\boldsymbol{x}_{\text{sh}}$, subject to the system constraint vector $\boldsymbol{g}_0$. The system objectives are set quality metrics [not part of Eq. (7)] as described in Sec. II.C. In Sec. IV, the use of set quality metrics as system objectives is explained. Subproblem 0 is formulated as follows:

Minimize
$\boldsymbol{x}_{\text{sh}}$

$$\begin{aligned} &\boldsymbol{q}\left(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_J^*\right) \\ &= \left\{q_1\left(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_J^*\right), \ldots, q_S\left(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_J^*\right)\right\}^t \end{aligned} \tag{8a}$$

subject to

$$\boldsymbol{g}_0(\boldsymbol{x}_{\text{sh}}) \le \boldsymbol{0}, \qquad \boldsymbol{g}_0 = \{g_{0,1}(\boldsymbol{x}_{\text{sh}}), \ldots, g_{0,K_0}(\boldsymbol{x}_{\text{sh}})\}^t \tag{8b}$$

where

$$\boldsymbol{x}_{\text{sh}} = \{x_{\text{sh},1}, \ldots, x_{\text{sh},N_0}\}^t \tag{8c}$$

The notation $(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_J^*)$ refers to the combination sets associated with $\boldsymbol{x}_{\text{sh}}$ (explained in Sec. IV). Subproblem 0 has N_0 variables, S objectives, and K_0 constraints.

Each of the jth subsystem subproblems, $j = 1, \ldots, J$, optimizes its local objectives $f_{j,i}$, with respect to its local variable vector $\boldsymbol{x}_j$, subject to its local constraint vector $\boldsymbol{g}_j$. (The shared variables $\boldsymbol{x}_{\text{sh}}$ are fixed.) The jth subsystem subproblem has N_j variables, M_j objectives, and K_j constraints. When the notation from Eq. (7) is used, the jth subsystem subproblem is formulated as follows:

Minimize
$\boldsymbol{x}_j$

$$\boldsymbol{f}_j(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j) = \left\{\begin{matrix} f_{j,1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j), \ldots, f_{j,R}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j), \\ f_{j,R+1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j), \ldots, f_{j,M_j}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j) \end{matrix}\right\}^t \tag{9a}$$

subject to

$$\boldsymbol{g}_j(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j) \le \boldsymbol{0}, \qquad \boldsymbol{g}_j = \{g_{j,1}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j), \ldots, g_{j,K_J}(\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_j)\}^t \tag{9b}$$

where

$$\boldsymbol{x}_j = \{x_{j,1}, \ldots, x_{j,N_j}\}^t \tag{9c}$$

Finally, for a properly decomposed problem, the following conditions are satisfied: $(N_0 + N_1 + \cdots + N_J) = N$, $[R + (M_1 - R) + \cdots + (M_J - R)] = M$, and $(K_0 + K_1 + \cdots + K_J) = K$.

IV. Description of M–MGA

In this section, we describe M–MGA, including the details needed to understand the algorithm steps presented in the following section. The description focuses on three key features of how M–MGA solves the decomposed optimization problem: reconstituting complete design variable vectors from the solutions of the subproblems, creating a "grand Pareto pool," and using set quality metrics as system objectives.

Figure 3 shows an overall schematic of M–MGA, depicting the hierarchical structure of the decomposed problem with $J+1$ subproblems. Because the subsystem subproblems in M–MGA have multiple objectives, optimizing each of the subproblems will result in a Pareto set (for each fixed value of the shared variables). We use MOGA as the optimizer for all subproblems. Hence, we have $J+1$ MOGAs, each with its own population, individuals, and Pareto set. We call the subproblem 0 MOGA the system MOGA, and we will use the corresponding terms system population and system individuals. We call the jth subsystem subproblem MOGA the jth local MOGA, and we will use the terms jth local population and jth local individuals. We will refer to the Pareto set generated by the jth subsystem subproblem as the jth local Pareto set.

In executing M–MGA, each iteration of the system MOGA creates a system population, that is, a set of $\boldsymbol{x}_{\text{sh}}$. For the first iteration

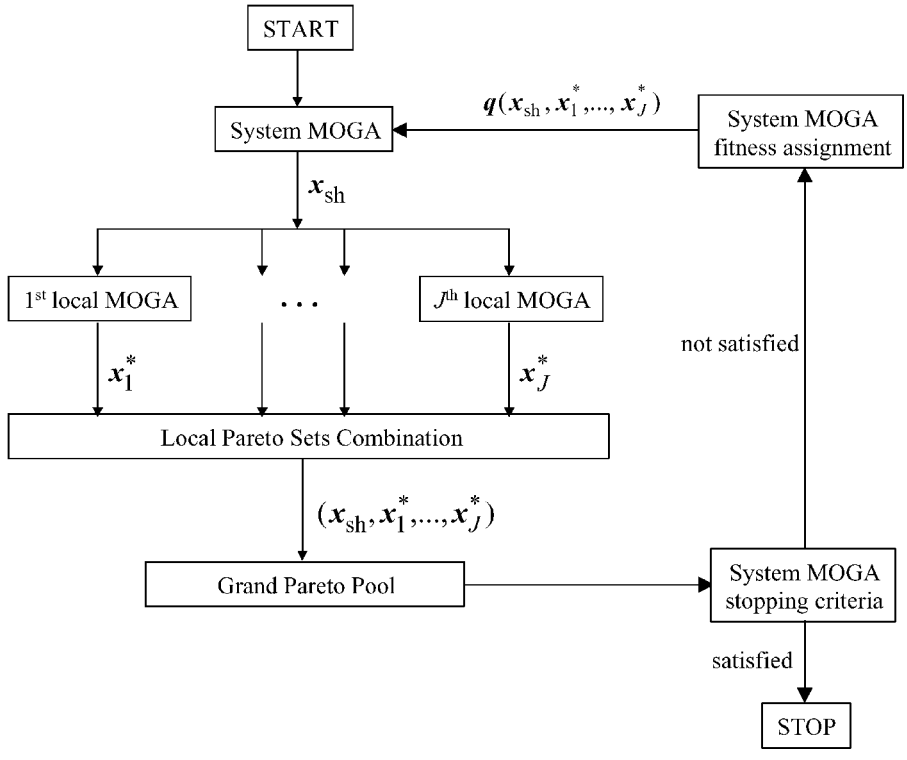


Fig. 3 Overall M–MGA schematic.

the population is generated randomly. For subsequent iterations the population is created by evolving the previous iteration's population. The system constraint $\boldsymbol{g}_0(\boldsymbol{x}_{sh})$ is evaluated for each individual in the population to identify the feasible individuals. The $\boldsymbol{x}_{sh}$ for each feasible individual is passed to the J local MOGAs. At the subsystem subproblems, each local MOGA treats the shared variables as fixed and optimizes its own local objectives with respect to its own local variables, creating its local Pareto set. For each $\boldsymbol{x}_{sh}$ there is now an associated group of J local Pareto sets.

We are seeking the Pareto set that is the solution of the original problem. The members of that set are complete variable vectors. To obtain complete variable vectors, we combine the $\boldsymbol{x}_{sh}$ and the points in the J local Pareto sets. For each $\boldsymbol{x}_{sh}$, one complete variable vector $\boldsymbol{x}$ can be formed by combining $\boldsymbol{x}_{sh}$ with one point from each of its associated J local Pareto sets. The point from the jth local Pareto set contributes to the local variables $\boldsymbol{x}_j$ that are unique to the jth subproblem.

Forming the combinations creates a set of complete variable vectors for each $\boldsymbol{x}_{sh}$. We refer to this set as the combination set. The number of complete variable vectors in the combination set, that is, the number of combinations, associated with each $\boldsymbol{x}_{sh}$ can be calculated as follows. Let $\boldsymbol{x}_{sh}^i$, $i = 1, \ldots, P_0$, denote a feasible system individual. The number of combinations associated with $\boldsymbol{x}_{sh}^i$ is then

$$P_{i,1} \times \cdots \times P_{i,J} = \prod_{j=1}^{J} P_{i,j} \tag{10}$$

Note that P_0 and the $P_{i,j}$ might change at every system MOGA iteration.

Each member of the combination set associated with $\boldsymbol{x}_{sh}^i$ is a point in the N-variable space that maps to a point in the M-objective space. Thus, each $\boldsymbol{x}_{sh}$ value maps to a set of points in the M-objective space. Because the $\boldsymbol{x}_{sh}$ do not map to unique points in the objective space of the original problem, the system MOGA cannot assign fitness based on the original objectives. Instead, the system MOGA uses as objectives the four set quality metrics described in Sec. B.III, *HD*, *OS*, *AC*, and *CL*.

We calculate the set quality metrics values of each $\boldsymbol{x}_{sh}$ set of points in M-objective space, thus mapping each $\boldsymbol{x}_{sh}$ value into a point in the four-objective space of the quality metrics. We then apply the methods of Fonseca and Fleming[15] (in the four-objective space of the quality metrics) to assign a fitness value to each $\boldsymbol{x}_{sh}$. The system MOGA uses the assigned fitness values in executing its evolutionary algorithms.

The preceding description of M–MGA applies only to the feasible individuals in the system population. Any infeasible system individuals are assigned arbitrarily low fitness values (typically 1/10 of the largest fitness value). When optimizing the subsystem subproblems, the shared variables $\boldsymbol{x}_{sh}$ are treated as parameters and kept fixed. It is possible that for some particular value of $\boldsymbol{x}_{sh}$, the feasible domain of a subsystem subproblem would be an empty set $\emptyset$, that is, the subproblem would not have any solutions. We treat any such $\boldsymbol{x}_{sh}$ as infeasible.

At each system MOGA iteration, a stopping criterion is checked to determine if further iterations are warranted. The basic concept of the stopping criterion is to have, at each iteration, a set that is an approximation of the Pareto set of the original problem and to stop system MOGA when that approximation has not changed significantly for some number of consecutive iterations. The solution that M–MGA will generate we call the grand Pareto set. We call the approximation at the current (or previous) iteration the current (or previous) grand Pareto set. We use the hyperarea difference *HD* set quality metric to measure changes in the current grand Pareto set. If *HD* does not change by more than some small amount, then the sets are considered not to have changed significantly.

At each system MOGA iteration, we form the current grand Pareto set by updating the previous grand Pareto set. The previous set is empty on the first iteration. We form the grand Pareto pool, which is the union of the previous grand Pareto set and all of the combination sets of all of the $\boldsymbol{x}_{sh}$. When the given notation is used, the number of points added to the grand Pareto pool at the current iteration is

$$\left(\prod_{j=1}^{J} P_{1,j} + \prod_{j=1}^{J} P_{2,j} + \cdots + \prod_{j=1}^{J} P_{P0,j} \right) = \sum_{i=1}^{P0} \prod_{j=1}^{J} P_{i,j} \tag{11}$$

The inferior points are removed from the pool, and the remaining nondominated solution points comprise the current grand Pareto set. When the stopping criterion is met, the current grand Pareto set

becomes the grand Pareto set, the M–MGA solution to the original problem.

V. M–MGA Implementation

We implement M–MGA with the following steps:

1) Generate an initial system population, a set of random $\boldsymbol{x}_{\text{sh}}$. Initialize the previous grand Pareto set to empty. Initialize the iteration counter: itr = 0.

2) Calculate the system constraint vector $\boldsymbol{g}_0$ for all system individuals, and identify the feasible system individuals.

3) For feasible system individuals, pass their $\boldsymbol{x}_{\text{sh}}$ values to all J subsystem subproblems, and perform the optimization of each subproblem to obtain local Pareto sets.

4) For each feasible system individual, combine its associated local Pareto sets to create combination sets of complete variable vectors.

5) Form the union of all of the combination sets and the previous grand Pareto set to form the grand Pareto pool.

6) Eliminate the inferior points from the grand Pareto pool, thus forming the current grand Pareto set.

7) Check whether the system MOGA stopping criterion has been satisfied. If it has, stop M–MGA. The current grand Pareto set is the M–MGA solution to the original problem. Otherwise, go to step 8.

8) For each feasible system individual, evaluate the set quality metrics of its associated combination set. Assign a fitness value to each feasible system individual based on the set quality metrics values. For infeasible system individuals, assign low fitness values.

9) Evolve system population based on the system individuals' fitness values. Set itr = itr + 1, and go to step 2 for the next iteration.

VI. Discussion

We have presented the details of the optimization procedure of M–MGA. In this section we show that the points in the M–MGA solution, that is, the grand Pareto set, are feasible with respect to the original problem. We then show that for any Pareto point of the original problem, M–MGA generates at least one point that is noninferior with respect to that Pareto point. Last, we show a comparison between the computational complexity of M–MGA and a single-level MOGA in terms of number of functions calls. In this discussion, we assume that each system and sub-system MOGA used in M–MGA is convergent and can obtain the Pareto set of its subproblem.

A. Feasibility of M–MGA Solutions

Recall from Sec. III that, in M–MGA, the constraint vector of the original problem, $\boldsymbol{g}$, is decomposed into the system constraint vector $\boldsymbol{g}_0$ and the local constraint vectors, $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_J$. Recall also that the jth constraint vector $\boldsymbol{g}_j$ is a function of only the shared variable vector $\boldsymbol{x}_{\text{sh}}$ and the jth local variable vector $\boldsymbol{x}_j$.

The argument is then straightforward. All of the $\boldsymbol{x}_{\text{sh}}$ values that are passed to the subsystem level satisfy the system constraint vector, $\boldsymbol{g}_0$. The $\boldsymbol{x}_{\text{sh}}$ and the local variables $\boldsymbol{x}_j$ values of the points in the jth local Pareto set satisfy the jth constraint vector $\boldsymbol{g}_j$. Because $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_J$ are all independent, the points in the combination sets satisfy $\boldsymbol{g}$. That is, in a combination $\{\boldsymbol{x}_{\text{sh}}, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_J\}$, $\boldsymbol{x}_{\text{sh}}$ satisfies $\boldsymbol{g}_0$; $\boldsymbol{x}_{\text{sh}}$ and $\boldsymbol{x}_1$ satisfy $\boldsymbol{g}_1$; …; $\boldsymbol{x}_{\text{sh}}$ and $\boldsymbol{x}_J$ satisfy $\boldsymbol{g}_J$. Thus, all of the complete design variable vectors in all of the combination sets are feasible, and so all of the points in the M–MGA solution are feasible with respect to the original problem.

B. M–MGA Solutions vs Pareto Set of the Original Problem

The set quality metric *HD* is a measure of how close a set of nondominated solution points is to the good point.[20,21] A necessary and sufficient condition for the Pareto set is that it has a lower *HD* value than any other subset of the feasible points.[22] In M–MGA, one of the system MOGA objectives is to minimize the *HD* of the combination sets. Thus, the $\boldsymbol{x}_{\text{sh}}$ values of the system MOGA population will converge to a $\boldsymbol{x}_{\text{sh}}$ value whose corresponding combination set has the lowest *HD* value compared to any other combination set. Let P^* denote this lowest *HD* combination set, and also let NP^* denote the set of nondominated solution points in P^*.

We now show that for any point PR^* in the Pareto set of the original problem there is at least one point in NP^* that is noninferior with respect to PR^*. Assume that all points in NP^* are inferior with respect to PR^*. Then, by the definition of the *HD* metric, the combination set that contains PR^* will have a lower *HD* value than NP^*. However, this is a contradiction because P^* is a combination set that has the lowest *HD* value compared to any other combination sets. Thus, we conclude that at least one point in NP^* must be noninferior with respect to PR^*.

In creating the grand Pareto set, we remove all inferior points from the grand Pareto pool. Hence, all of the points in the grand Pareto set either dominate or are noninferior with respect to NP^*. Following our proof, for any Pareto point of the original problem, there is at least one point in the grand Pareto set that is noninferior with respect to that Pareto point.

To improve the probability of obtaining combination sets that contain Pareto points, all combination sets having *HD* values slightly higher than P^* also ought to be sought. M–MGA accomplishes this by optimizing additional set quality metrics, that is, the quality metrics *OS*, *AC*, and *CL*, as system objectives. When there is a multiobjective optimization at the system level, the $\boldsymbol{x}_{\text{sh}}$ in the system population will converge not only to the $\boldsymbol{x}_{\text{sh}}$ associated with P^*, but also to those $\boldsymbol{x}_{\text{sh}}$ for which the *HD* values of their combination sets are close to that of P^*.

C. Computational Complexity of M–MGA

Equation (12) gives the total number of function calls used in a single level MOGA, calculated by counting the number of function calls made per iteration:

$$F_N(\text{MOGA}) = (\text{pop} + \text{itr} \times \text{rep})(M + K) \tag{12}$$

When a similar counting procedure is used, the total number of functions calls made by M–MGA is shown in Eq. (13). Here, the subscripts 0 and l refer to the system and local MOGAs, respectively, and *PF* is the fraction of the system population that is feasible ($PF \leq 1$):

$$F_N(\text{MMGA}) = (\text{pop}_0 + \text{itr}_0 \times \text{rep}_0) \times [K_0 + PF \times (\text{pop}_l + \text{itr}_l \times \text{rep}_l)(M + K - K_0)] \tag{13}$$

Typically, the quantity $(\text{pop}_l + \text{itr}_l \times \text{rep}_l)$ is on the order of 10^3, whereas the number of system constraints K_0 is on the order of 10^1. In addition, because the original problem is decomposed into many subproblems, K_0 typically comprises only a small portion of the total number of constraints, K. Hence, in the second bracket of Eq. (13), the term K_0 is negligibly small and, thus, can be omitted. When Eqs. (12) and (13) are compared, the ratio between the numbers of functions calls made by M–MGA and by a single level MOGA is

$$\frac{F_N(\text{M–MGA})}{F_N(\text{MOGA})} = \frac{PF \times (\text{pop}_0 + \text{itr}_0 \times \text{rep}_0)(\text{pop}_l + \text{itr}_l \times \text{rep}_l)}{(\text{pop} + \text{itr} \times \text{rep})} \tag{14}$$

For a typical M–MDO problem, M–MGA can obtain solutions using pop = 50 and rep = 10 as the system and subsystem MOGAs parameters. Also, the grand Pareto set usually converges after approximately itr = 50 (as in Sec. VII). For the same problem, the equivalent single level MOGA can usually obtain solutions using pop = 300, rep = 30, and itr = 1000 (as in Sec. VII). Substituting these values into Eq. (14), we observe that for a typical problem, M–MGA requires about four or five times more functions calls than a single-level MOGA.

VII. Examples

As a demonstration, we applied M–MGA to two engineering examples: the design of a speed reducer and the design of a payload for an undersea autonomous vehicle. For comparison, we solved the

Table 1 System level MOGA parameters in M–MGA

Parameter	Value
Chromosome length (bits per variable)	
Continuous variables	10
Discrete variables	As needed
Population size	50
Replacement (per iteration)	10
Crossover probability	0.95
Mutation probability	0.05
Crossover type	Two-point crossover
Selection type	Stochastic universal selection

Table 2 Subsystem level MOGAs parameters in M–MGA

Parameter	Value
Chromosome length (bits per variable)	
Continuous variables	10
Discrete variables	As needed
Population size	50
Replacement (per iteration)	10
Crossover probability	0.95
Mutation probability	0.05
Crossover type	Two-point crossover
Selection type	Stochastic universal selection
Stopping criterion	Stop after 50 iterations

Table 3 MOGA parameters (single level)

Parameter	Value
Chromosome length (bits per variable)	
Continuous variables	10
Discrete variables	As needed
Population size	300
Replacement (per iteration)	30
Crossover probability	0.95
Mutation probability	0.05
Crossover type	Two-point crossover
Selection type	Stochastic universal selection
Stopping criterion	Stop after 1000 iterations

undecomposed formulations of the same examples using a single-level MOGA.[15,19] Tables 1 and 2 list the system and local MOGA parameters used in M-MGA. Table 3 lists the parameters used in the single-level MOGA.

When the parameters in Tables 1–3 are substituted in to Eq. (14), and $PF = 0.5$ is assumed, the ratio of the number of functions calls between M–MGA and MOGA for these examples is obtained:

$$\frac{F_N(\text{MMGA})}{F_N(\text{MOGA})} = \frac{(0.5)(50 + 50 \times 10)(50 + 50 \times 10)}{(300 + 30 \times 1000)} \approx 5 \quad (15)$$

A. Design of a Speed Reducer

This well-known speed reducer example was originally formulated by Golinski[23] as a single objective optimization problem. Figure 4 shows the speed reducer.

We convert the problem into a three-objective optimization problem. The mathematical formulation of the problem is as follows:

Minimize

$$f_1 = 0.7854x_1x_2^2\left[\left(10x_3^2/3\right) + 14.933x_3 - 43.0934\right] - 1.508x_1\left(x_6^2 + x_7^2\right) + 7.477\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right)$$

minimize

$$f_2 = \sqrt{(745x_4/x_2x_3)^2 + 1.69 \times 10^7}\Big/0.1x_6^3$$

minimize

$$f_3 = \sqrt{(745x_5/x_2x_3)^2 + 1.575 \times 10^8}\Big/0.1x_7^3 \quad (16a)$$

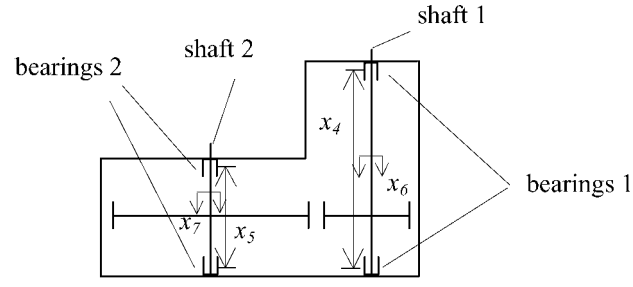


Fig. 4 Speed reducer.

subject to

$$g_1 : \left[1/\left(x_1x_2^2x_3\right)\right] - (1/27) \le 0$$

$$g_2 : \left[1/\left(x_1x_2^2x_3^2\right)\right] - (1/397.5) \le 0$$

$$g_3 : \left[x_4^3/\left(x_2x_3x_6^4\right)\right] - (1/1.93) \le 0$$

$$g_4 : \left[x_5^3/\left(x_2x_3x_7^4\right)\right] - (1/1.93) \le 0, \qquad g_5 : x_2x_3 - 40 \le 0$$

$$g_6 : (x_1/x_2) - 12 \le 0, \qquad g_7 : 5 - (x_1/x_2) \le 0$$

$$g_8 : 1.9 - x_4 + 1.5x_6 \le 0, \qquad g_9 : 1.9 - x_5 + 1.1x_7 \le 0$$

$$g_{10} : f_2 - 1300 \le 0, \qquad g_{11} : f_3 - 1100 \le 0 \quad (16b)$$

where

$$2.6 \le x_1 \le 3.6, \qquad 0.7 \le x_2 \le 0.8$$

$$17 \le x_3 \le 28, \qquad 7.3 \le x_4 \le 8.3, \qquad 7.3 \le x_5 \le 8.3$$

$$2.9 \le x_6 \le 3.9, \qquad 5.0 \le x_7 \le 5.5 \quad (16c)$$

The seven design variables are the gear face width x_1, the teeth module x_2, the number of teeth pinion x_3 (integer variable), the distance between bearings 1 x_4, the distance between bearings 2 x_5, the diameter of shaft 1 x_6, and the diameter of shaft 2 x_7. The first design objective f_1 is to minimize the sum of the volumes of the speed reducer parts. The second objective f_2 is to minimize the stress in shaft 1. The third objective f_3 is to minimize the stress in shaft 2.

The design is subject to a number of constraints imposed by gear and shaft designs. There are 11 inequality constraints: g_1 is an upper bound on the bending stress of the gear teeth, g_2 is an upper bound on the contact stress of the gear tooth, g_3 and g_4 are upper bounds on the transverse deflection of the shafts, g_5, g_6, and g_7 are dimensional restrictions based on space and/or experience, g_8 and g_9 are design requirements on the shaft based on experience, and g_{10} and g_{11} are constraints on stresses in the gear shafts. Finally, upper and lower limits are imposed on each of the seven design variables.

To apply M–MGA, the original problem [Eq. (16)] is decomposed into three subproblems as shown in Fig. 5.

The mathematical formulation of the decomposed problem is as follows: For subproblem 0, minimize

$$HD(x_1, x_2, x_3) \quad (17a)$$

maximize

$$OS(x_1, x_2, x_3) \quad (17b)$$

maximize

$$AC(x_1, x_2, x_3) \quad (17c)$$

minimize

$$CL(x_1, x_2, x_3) \quad (17d)$$

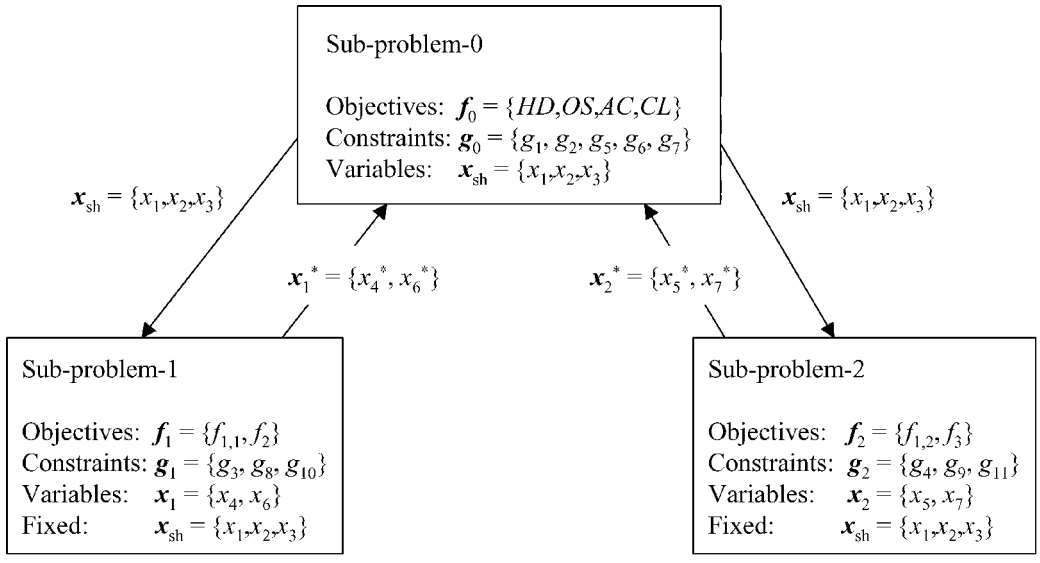


Fig. 5 Hierarchical M–MDO structure of the speed reducer design example.

subject to

$$g_1 : \left[1/\left(x_1 x_2^2 x_3\right)\right] - (1/27) \le 0$$

$$g_2 : \left[1/\left(x_1 x_2^2 x_3^2\right)\right] - (1/397.5) \le 0$$

$$g_5 : x_2 x_3 - 40 \le 0$$

$$g_6 : (x_1/x_2) - 12 \le 0, \qquad g_7 : 5 - (x_1/x_2) \le 0 \tag{17e}$$

where

$$2.6 \le x_1 \le 3.6, \qquad 0.7 \le x_2 \le 0.8, \qquad 17 \le x_3 \le 28 \tag{17f}$$

The shared variable vector is $\boldsymbol{x}_{sh} = \{x_1, x_2, x_3\}$. The system constraint vector is $\boldsymbol{g}_0 = \{g_1, g_2, g_5, g_6, g_7\}$.

For subproblem 1, minimize

$$f_{1,1} = 0.7854 x_1 x_2^2 \left(\frac{10 x_3^2}{3} + 14.933 x_3 - 43.0934\right) - 1.508 x_1 x_6^2 + 7.477 x_6^3 + 0.7854 x_4 x_6^2 \tag{18a}$$

minimize

$$f_2 = \sqrt{(745 x_4/x_2 x_3)^2 + 1.69 \times 10^7}\Big/ 0.1 x_6^3 \tag{18b}$$

subject to

$$g_3 : \left[x_4^3/\left(x_2 x_3 x_6^4\right)\right] - (1/1.93) \le 0, \qquad g_8 : 1.9 - x_4 + 1.5 x_6 \le 0$$

$$g_{10} : f_2 - 1300 \le 0 \tag{18c}$$

where

$$7.3 \le x_4 \le 8.3, \qquad 2.9 \le x_6 \le 3.9 \tag{18d}$$

In subproblem 1, the local variable vector is $\boldsymbol{x}_1 = \{x_4, x_6\}$, the local objective vector is $\boldsymbol{f}_1 = \{f_{1,1}, f_2\}$, and the local constraint vector is $\boldsymbol{g}_1 = \{g_3, g_8, g_{10}\}$.

For subproblem 2, minimize

$$f_{1,2} = -1.508 x_1 x_7^2 + 7.477 x_7^3 + 0.7854 x_5 x_7^2 \tag{19a}$$

minimize

$$f_3 = \sqrt{(745 x_5/x_2 x_3)^2 + 1.575 \times 10^8}\Big/ 0.1 x_7^3 \tag{19b}$$

subject to

$$g_4 : \left[x_5^3/\left(x_2 x_3 x_7^4\right)\right] - (1/1.93) \le 0, \qquad g_9 : 1.9 - x_5 + 1.1 x_7 \le 0$$

$$g_{11} : f_3 - 1100 \le 0 \tag{19c}$$

where

$$7.3 \le x_5 \le 8.3, \qquad 5.0 \le x_7 \le 5.5 \tag{19d}$$

In subproblem 2, the local variable vector is $\boldsymbol{x}_2 = \{x_5, x_7\}$, the local objective vector is $\boldsymbol{f}_2 = \{f_{1,2}, f_3\}$, and the local constraint vector is $\boldsymbol{g}_2 = \{g_4, g_9, g_{11}\}$. Overall, in the decomposed formulation $J = 2$, $R = 1$, $M_1 = 2$, and $M_2 = 2$.

The Pareto set generated by M–MGA is shown in the objective space in Fig. 6. In Fig. 6, we have shown the design points projected into a two-objective space. The results from solving the same problem [Eq. (16)] using a MOGA are also shown in Fig. 6.

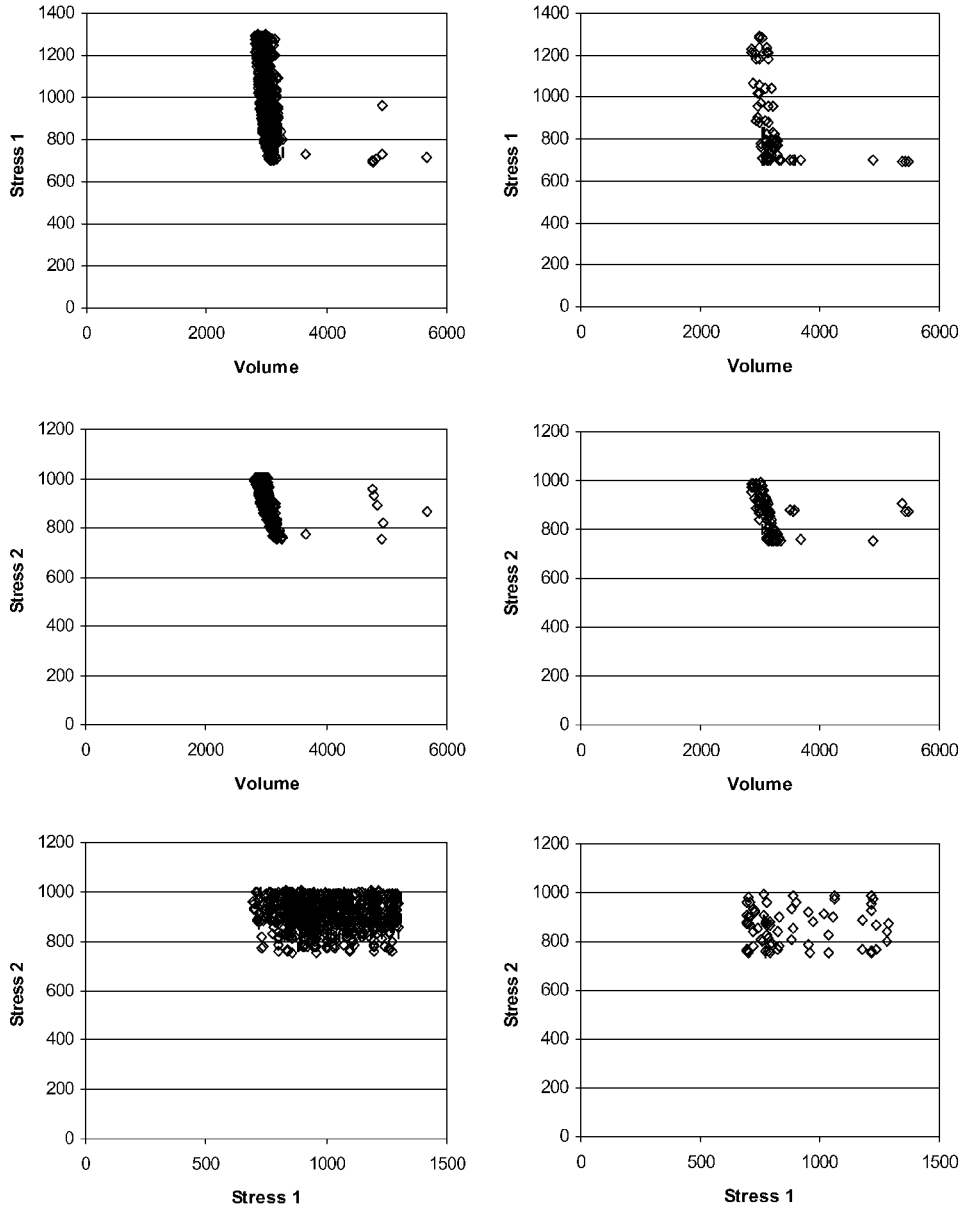
It can be seen in Fig. 6 that the M–MGA solutions closely approximate the MOGA solutions. In all three planes, M–MGA Pareto solution points closely follow the points found by MOGA. Figure 6 also shows that M–MGA produces more solution points than MOGA. For example, in the volume-stress2 plane, M-MGA found some Pareto points not found by MOGA.

Because M–MGA is a stochastic method, M–MGA solutions may differ from one M–MGA run to another. In this example, to assess how the solutions change, we ran M–MGA 50 times and measured the *HD* of the grand Pareto set generated from each run. From these 50 M–MGA runs, we obtain an average *HD* value of 0.588, and the *HD* values are within ±0.36% from this mean value. This result shows that, for this example, M–MGA produces relatively the same solutions from one M–MGA run to another.

B. Design of Payload for Undersea Autonomous Vehicle

As a second example, we applied M–MGA to the design of an undersea payload. This problem is part of a larger undersea autonomous vehicle (UAV) design problem that has many other subproblems, such as propulsion, sensor, and control systems, that interact with the system coordinator. Our investigations have been limited to the design of the payload.

Typically, the payload must be effective in several different uses, called scenarios. Effectiveness in a scenario is measured by the probability of success P_S in that scenario. The design goal is to maximize simultaneously the individual P_S for all of the scenarios. The payload design is constrained by upper limits on the weight of the payload and on the radiated noise generated by the payload.



a) Generated by M-MGA b) Generated by MOGA

Fig. 6 Pareto sets of the speed reducer design.

There are six design variables: the payload length *PL*, the hull diameter *DH*, the material of the hull *HM*, the payload type *PT*, the first inner material type $I1$, and the second inner material type $I2$. Four of the variables are discrete: *HM, PT,* $I1$, and $I2$. For discrete variable $I2$, the range, that is, the options available to the designer, depends on the values (selected options) of *PT* and of $I1$. The other two variables, *PL* and *DH*, are continuous. For any particular payload design problem, maximum-value constraints on *PL* and *DH* are provided. In addition to the six design variables, there is a fixed continuous design variable, the maximum depth at which the payload operates. Unlike the speed reducer design example, there are no closed-form relationships to map the design variables to the constraints and to the P_S. Rather, we were provided with a design analyzer (computer program) that maps the design variables to the payload weight, the radiated noise, and the P_S for the scenarios.

In this example, we address a two objective payload design optimization with two constraints. The two objectives are to maximize P_{S1} and P_{S2} for two different scenarios. The two constraints are a 65-lb (30-kg) upper bound on the payload weight and a 0.16-W/m^2 upper bound on the radiated noise generated. The value of the maximum depth is 3000 ft (915 m) (fixed). The problem is mathematically formulated as follows:

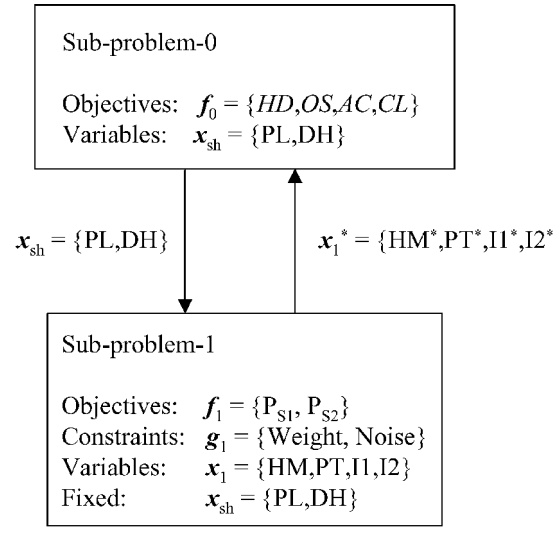
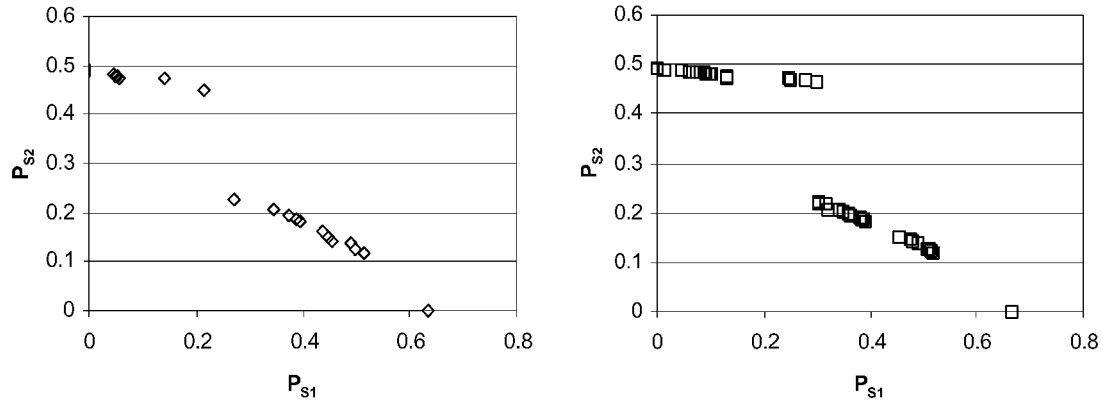


Fig. 7 Hierarchical (bilevel) M-MDO structure of the undersea payload design example.



a) Generated by M-MGA **b) Generated by MOGA**

Fig. 8 Pareto sets of the payload design of UAV (maximization).

Maximize

$$P_{S1}(PL, DH, HM, PT, I1, I2) \tag{20a}$$

maximize

$$P_{S2}(PL, DH, HM, PT, I1, I2) \tag{20b}$$

subject to

$$\text{weight}(PL, DH, HM, PT, I1, I2) - 65 \leq 0$$

$$\text{noise}(PL, DH, HM, PT, I1, I2) - 0.16 \leq 0 \tag{20c}$$

To apply M–MGA, we pose the problem as a bilevel UAV design problem, but we consider only one subsystem subproblem, the payload (Fig. 7). The shared variables from the system level UAV subproblem are *PL* and *DH*.

Figure 8 shows the Pareto set of the problem generated by M–MGA. Figure 8 also shows the Pareto set generated by applying MOGA to the same problem [Eq. (20)].

As with the speed reducer example, the M–MGA solution closely approximates the MOGA solution. The Pareto points obtained by M–MGA closely resemble the Pareto points obtained by MOGA. M–MGA results also have a good diversity similar to that of the diversity of MOGA's results. Observe in Fig. 8 that some MOGA solutions dominate M–MGA solutions. This discrepancy might be due to the stochastic nature of MOGA and M–MGA.

To assess the reproducibility of the undersea payload example results, we ran M–MGA 50 times and measured the *HD* value of the grand Pareto set from each run. The 50 *HD* values that we obtained have a mean value of 0.864, and they are distributed within 2.3% of this mean value. This result shows that, for this example, M–MGA produces relatively the same solutions from one M–MGA run to another.

VIII. Summary

In this paper, we presented M–MGA, a new method to solve M–MDO problems. M–MGA is applicable to those M–MDO problems that can be decomposed hierarchically into multi-objective subproblems and whose objectives are separable or additively separable. In the decomposition, the subproblems may have both common and unique objectives. Each subproblem is solved by a MOGA, so that M–MGA is applicable to problems with nondifferentiable objective and constraint functions and to problems with a mix of continuous and discrete variables. M–MGA has three novel features used in solving the decomposed optimization problem: reconstitution of complete design variable vectors from the solutions of the subproblems, creation of a grand Pareto pool, and use of set quality metrics as system objectives.

We showed that for any Pareto point of the original (single-level) problem, M–MGA generates at least one point that is noninferior with respect to that Pareto point. We presented two engineering examples: the design of a speed reducer and the design of a payload for a UAV. We observed in both examples that the solutions generated by M–MGA are similar to the solutions generated by MOGA. We also observed that M–MGA produces relatively the same solutions from one M–MGA run to another.

Acknowledgments

The work presented in this paper was supported in part by a contract from the Indian Head Division, Naval Surface Warfare Center. The Indian Head Division was funded by Office of Naval Research via funding documents N00014-01-WX-2-0307 and N00014-02-WX-20470. The work was also supported in part by National Science Foundation Grant 0112767. Such support does not constitute an endorsement by the funding agency of the opinions expressed in the paper.

References

[1]Sobieszczanski-Sobieski, J., "A Linear Decomposition Method for Large Optimization Problem—Blueprint for Development," NASA TM 83248, Feb. 1982.

[2]Balling, R. J., and Sobieszczanski-Sobieski, J., "Optimization of Coupled Systems: A Critical Overview of Approaches," *AIAA Journal*, Vol. 34, No. 1, 1996, pp. 6–17.

[3]Sobieszczanski-Sobieski, J., and Haftka, R. T., "Multidisciplinary Aerospace Design Optimization—Survey of Recent Developments," *Structural Optimization*, Vol. 14, No. 1, 1997, pp. 1–23.

[4]Kirsch, U., "Multilevel Approach to Optimum Structural Design," *Journal of the Structural Division*, Vol. 101, No. ST4, 1975, pp. 957–974.

[5]Lasdon, L. S., *Optimization Theory for Large Systems*, Macmillan, London, 1970.

[6]Azarm, S., and Li, W. C., "A Two-Level Decomposition Method for Design Optimization," *Engineering Optimization*, Vol. 13, 1988, pp. 211–224.

[7]Haimes, Y. Y., Tarvainen, K., Shima, T., and Thadathil, J., *Hierarchical Multiobjective Analysis of Large-Scale Systems*, Hemisphere, New York, 1990.

[8]Tappeta, R. V., and Renaud, J., "Multiobjective Collaborative Optimization," *Journal of Mechanical Design*, Vol. 119, 1997, pp. 403–411.

[9]Korngold, J., and Gabriele, G. A., "Multidisciplinary Optimization of Mixed-Discrete Problems," *SIAM Multidisciplinary Design Optimization: State of the Art*, edited by N. M. Alexandrov and M. Y. Hussaini, Society for Industrial and Applied Mathematics, Philadelphia, 1997, pp. 117–132.

[10]Kurapati, A., and Azarm, S., "Immune Network Simulation with Multiobjective Genetic Algorithms for Multidisciplinary Design Optimization," *Engineering Optimization*, Vol. 33, 2000, pp. 245–260.

[11]Miettinen, K. M., *Nonlinear Multiobjective Optimization*, Kluwer Academic, Boston, 1999.

[12]Holland, J. H., *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, MI, 1975.

[13]Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley, Reading, MA, 1989.

[14]Schaffer, J. D., "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," *Proceedings of the 1st International Conference on Genetic Algorithms*, edited by J. Grefenstette, Lawrence Erlbaum Associates, Mahwah, NJ, 1991, pp. 93–100.

[15]Fonseca, C. M., and Fleming, P. J., "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion, and Generalization," *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann, San Francisco, 1993, pp. 416–423.

[16]Srinivas, N., and Deb, K., "Multiobjective Function Optimization Using Nondominated Sorting Genetic Algorithms," *Evolutionary Computation Journal*, Vol. 2, No. 3, 1995, pp. 221–248.

[17]Narayanan, S., and Azarm, S., "On Improving Multiobjective Genetic Algorithms for Design Optimization," *Structural Optimization*, Vol. 18, 1999, pp. 146–155.

[18]Deb, K., *Multi-objective Optimization Using Evolutionary Algorithms*, Wiley, Chichester, England, U.K., 2001.

[19]Kurapati, A., Azarm, S., and Wu, J., "Constraint Handling Improvements for Multi-Objective Genetic Algorithms," *Structural and Multidisciplinary Optimization*, Vol. 23, No. 3, 2002, pp. 204–213.

[20]Wu, J., and Azarm, S., "Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set," *Journal of Mechanical Design*, Vol. 123, 2001, pp. 18–25.

[21]Wu, J., "Quality Assisted Multiobjective and Multi-Disciplinary Genetic Algorithms," Ph.D. Dissertation, Dept. of Mechanical Engineering, Univ. of Maryland, College Park, MD, Dec. 2001.

[22]Fleischer, M., "The Measure of Pareto Optima: Applications to Multi-Objective Metaheuristics," Inst. for Systems Research, TR 2002-32, Univ. of Maryland, College Park, MD, 2002.

[23]Golinski, J., "Optimal Synthesis Problems Solved by Means of Nonlinear Programming and Random Methods," *Journal of Mechanisms*, Vol. 5, 1970, pp. 287–309.

A. Messac
Associate Editor