A Multiobjective Evolutionary Algorithm Toolbox for Computer-Aided Multiobjective Optimization

K. C. Tan, *Member, IEEE*, Tong H. Lee, *Member, IEEE*, D. Khoo, and E. F. Khor

***Abstract*—This paper presents an interactive graphical user interface (GUI) based multiobjective evolutionary algorithm (MOEA) toolbox for effective computer-aided multiobjective (MO) optimization. Without the need of aggregating multiple criteria into a compromise function, it incorporates the concept of Pareto's optimality to evolve a family of nondominated solutions distributing along the tradeoffs uniformly. The toolbox is also designed with many useful features such as the goal and priority settings to provide better support for decision-making in MO optimization, dynamic population size that is computed adaptively according to the online discovered Pareto-front, soft/hard goal settings for constraint handlings, multiple goals specification for logical "AND"/"OR" operation, adaptive niching scheme for uniform population distribution, and a useful convergence representation for MO optimization. The MOEA toolbox is freely available for download at http://vlab.ee.nus.edu.sg/~kctan/moea.htm, which is ready for immediate use with minimal knowledge needed in evolutionary computing. To use the toolbox, the user merely needs to provide a simple "model" file that specifies the objective function corresponding to his/her particular optimization problem. Other aspects like decision variable settings, optimization process monitoring and graphical results analysis can be performed easily through the embedded GUIs in the toolbox. The effectiveness and applications of the toolbox are illustrated via the design optimization problem of a practical ill-conditioned distillation system. Performance of the algorithm in MOEA toolbox is also compared with other well-known evolutionary MO optimization methods upon a benchmark problem.**

***Index Terms*—Evolutionary algorithms, multiobjective optimization, software.**

I. INTRODUCTION

MANY real-world design tasks involve complex optimization problems with various competing design specifications and constraints which are often difficult, if not impossible, to be solved without the aid of powerful and efficient optimization algorithms [1]–[5]. Blessed with the rapid development of computer technology, the high power of computation and visualization available at the desk nowadays allows the embedding of computer-aided multiobjective optimization (CAMOO) technology into a virtual problem-solving environment for sophisticated optimization problems. Built upon a high-performance global optimization algorithm, such a computational framework promotes active man-machine interactions and supports the fundamental changes from conventional manual tuning process to automatic search of optimum solutions.

Manuscript received June 19, 2000; revised November 27, 2000 and April 24, 2001. This paper was receommended by Associate Editor E. Santos.

The authors are with Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119260, Republic of Singapore.

Publisher Item Identifier S 1083-4419(01)05971-4.

Evolutionary algorithm is a global search optimization technique based on the mechanics of natural selection and reproduction. It has been found to be very effective in solving complex multiobjective (MO) optimization problems where conventional optimization tools fail to work well [6]–[11]. Without the need of linearly combining multiple attributes into a composite scalar objective function, evolutionary algorithms incorporate the concept of Pareto's optimality or modified selection schemes to evolve a family of solutions along the tradeoff surface. Such a global optimization method has been applied to many real-world applications including treatment of cancer in medical fields [12], control engineering design in power systems [13], physiological processes of biological plants [14], and recognition of Chinese characters [15]. Several surveys are available for more information of evolutionary algorithms in MO optimization, e.g., [16]–[20].

Although evolutionary algorithms are powerful for MO optimization, the users require certain programming expertise with considerable time and effort in order to write a computer program for implementing the often sophisticated algorithm according to their need. This work could be tedious and needs to be done before users can start their design task for which they should really be engaged in. A simple solution is to get ready-to-use evolutionary optimization toolbox, which is often developed for general purposes but has the potential to be applied to any specific application. Generally there are two types of evolutionary algorithm toolboxes for MO optimization that are available in the market. 1) The key functions of the evolutionary algorithm are coded separately in the toolbox where the users can build their own programs by calling the relevant functions. 2) A ready-to-use toolbox where users merely write a "model" file that specifies the objective function corresponding to his/her particular optimization problem, and plugs the file into the toolbox for immediate solutions.

Existing evolutionary algorithm toolboxes include the genetic and evolutionary algorithm toolbox (GEATbx) for use with Matlab [21], which is developed by Pohlheim [22] and is commercially available. The toolbox is modular and contains some evolutionary functions that could be used for users' own programs. The interface with GEATbx can be performed via either command-line interpretation or simple GUIs. This toolbox is, however, not freely available and the current version of GEATbx (Version 1.92) provides no support for multiobjective optimization. The genetic algorithms for optimization toolbox (GAOT) developed by Houck et al., [23] in North Carolina State University requires the simulation settings to be specified in different variables, where reference materials may be needed to remind users of the functions of those variables storing the settings. Besides the need of inputting a long string of variables before each simulation, the toolbox also provides limited support for MO optimization and has no easy access to graphical displays for results analysis. Similar to GEATbx, GAOT requires users to be familiar with Matlab and evolutionary computing in order to understand the various functions as well as to specify the decision variables in the relevant m-files, since these toolboxes are mainly text-based driven with limited GUI supports. The FlexToolGA developed from Flexible Intelligence Group [24] is a ready-to-use toolbox which is also implemented in Matlab and is commercially available. Although this toolbox supports GUIs, some of the settings need to be entered via text-based commands. Moreover, the toolbox does not fully support MO optimization in identifying the entire tradeoffs for multiple conflicting criteria problems.

Addressing the need of a more user-friendly and comprehensive evolutionary algorithm toolbox for MO optimization, this paper presents a global optimization toolbox that is built upon the MOEA algorithm proposed in [25]. The MOEA toolbox, which is freely available for download at http://vlab.ee.nus.edu.sg/~kctan/moea.htm, is ready for immediate use with minimal knowledge needed in Matlab or evolutionary computing. It is fully equipped with interactive GUIs and powerful graphical displays for ease-of-use and efficient visualization of different simulation results, and hence provides excellent supports for decision-making and optimization in complex real-world optimization applications. Besides the ability of evolving a family of nondominated solutions along the observed Pareto optimal front, each of the objective components can have different goal settings or preferences to guide the optimization for individual specifications rather than pre-weighting the multiobjective functions manually. The toolbox also contains various analysis tools for users to compare, examine or analyze different results or tradeoffs at anytime during the simulation.

This paper is organized as follows: Section II presents a general architecture for computer-aided multiobjective optimization. The role of MOEA toolbox for global optimization and better decision-making in CAMOO is also illustrated. The design of MOEA toolbox which includes a brief description of various useful features as well as further developments for real-time optimization are also presented in Section II. The effectiveness of the toolbox is demonstrated via the design optimization problem of a practical ill-conditioned distillation control application in Section III. Section IV shows the performance comparison results among the MOEA toolbox and other well-known evolutionary methods upon a benchmark problem. Conclusions and future development of the toolbox are drawn in Section V.

II. Roles and Features of MOEA Toolbox

A. Background

In general, multiobjective optimization seeks to optimize a vector of noncommensurable and often competing objectives or cost functions. In other words, it tends to find a decision variable set $\boldsymbol{P}$ for

$$\mathrm{Min}_{P\in\Phi}\boldsymbol{F}(\boldsymbol{P}) \tag{1}$$

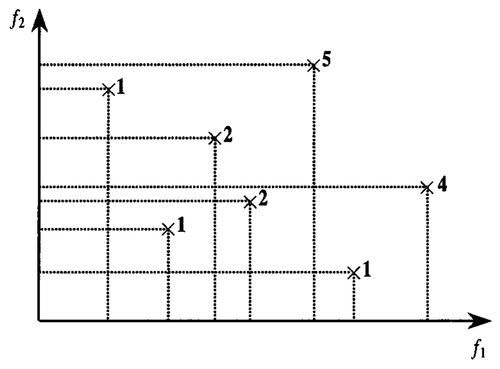


Fig. 1. Multiobjective Pareto ranking scheme.

where $\boldsymbol{P} = \{p_1, p_2, \ldots, p_n\}$, is the candidate vector with n decision variables and Φ defines the set of candidate vectors; $\boldsymbol{F} = \{f_1, f_2, \ldots, f_m\}$ are the m objectives to be minimized. In the total absence of information for preference of objectives, Pareto's dominance is regarded as a useful approach to compare strength or fitness between any two candidate solutions in MO optimization [7]. For a minimization problem, an objective vector $\boldsymbol{F}_a$ is said to dominate another objective vector $\boldsymbol{F}_b$, denoted by $\boldsymbol{F}_a \prec \boldsymbol{F}_b$, *iff*

$$\begin{aligned} &f_{a,i} \le f_{b,i} \quad \forall i \in \{1,2,\ldots,m\} \quad \text{and} \\ &f_{a,j} < f_{b,j}, \ \text{for some}\ j \in \{1,2,\ldots,m\} \end{aligned} \tag{2}$$

Solution to the above MO optimization problem is a family of points known as Pareto optimal solutions, where each objective component of any point along the Pareto-front can only be improved by degrading at least one of its other objective components [26], [27]. To illustrate the concept of Pareto's optimality, the Pareto ranking scheme proposed in [18] for a minimization problem of two objectives f_1 and f_2 is shown in Fig. 1. As can be seen, it assigns the same smallest cost for all nondominated strings, while the dominated strings are inversely ranked according to how many strings in the population dominate them.

B. General Architecture of CAMOO

To promote active man-machine interaction and to support automatic search of Pareto optimal solutions in MO optimization, a general architecture of CAMOO that accommodates three interactive modules is shown in Fig. 2.

- The human decision-making module that monitors and supervises the overall optimization process.
- The global optimization module that searches for Pareto optimal solutions automatically.
- The evaluation module that formulates and simulates all specifications and objective functions corresponding to the MO optimization problem on-hand.

According to the simulation results in evaluation module and any optimization guidance such as optional goal and priority information from decision-making module, the optimization module automates the search toward the global and Pareto optimal solutions, without the need of formulating a convex or linearly parameterized objective function. Online

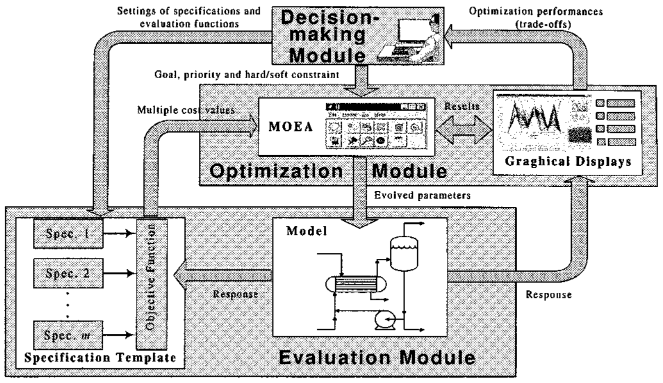
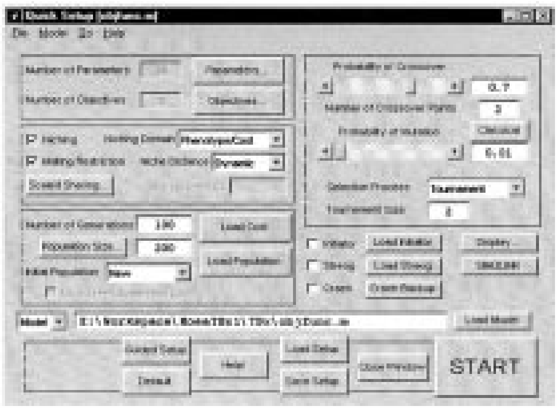


Fig. 2. A general architecture for computer-aided multiobjective optimization.



Probability of Crossover 0.7
Number of Crossover Points 2
Probability of Mutation 0.01
Number of Generations 100
Population Size 200

Fig. 3. GUI window for quick setting of simulation parameters.

optimization progress and simulation results like tradeoffs or convergence can be displayed graphically and fed back to the decision-making module. In this way, the overall optimization process is supervised and monitored closely, where the users can examine different competing tradeoffs conveniently, adjust goal settings that are too stringent or generous, or even alter the objective functions anytime during the simulation if necessary. This man-machine interactive optimization process may continue until user is satisfied with the required performances or after all design specifications have been met. The CAMOO architecture allows the interactive optimization process to be closely linked to the environment of the applications. Human decision-makers, for most of the part, are not required to deal with any details related to the optimization algorithm, which greatly simplifies the overall design task.

To achieve an efficient CAMOO, a powerful software package is essential for the optimization module to obtain the globally optimized solutions. The interactive GUI-based MOEA optimization toolbox developed under the Matlab programming environment is thus designed for this purpose. Matlab is a popular high-performance programming language used for technical computing. It integrates computation, visualization and programming in an easy-to-use environment, where problems and solutions can be expressed in familiar mathematical notation. It is chosen as the software environment for MOEA toolbox implementation due to the following reasons [23]:

1) it provides many built-in auxiliary functions useful for function optimization in engineering or nonengineering applications;
2) it is portable and is efficient for numerical computations;
3) it provides powerful and easy-to-use graphic utilities;
4) it provides Application Program Interface (API) to interact with data and programs that are external to Matlab environment;
5) it is capable of generating optimized code for embedded systems, rapid prototyping and hardware-in-the-loop designs.

Although execution speed in Matlab may be slow as compared to other low-level programming languages like C/C++, function files in Matlab that require extensive computational effort can be compiled into "mex" files using software like Matcom [28] for faster program execution, if so desired.

C. GUIs of MOEA Toolbox

The MOEA toolbox is developed based upon the technique of evolutionary computing and the concept of Pareto's optimality for effective MO optimization. Interfacing with the toolbox is through powerful GUI windows. Most simulation settings can be done by manipulating labeled graphical controls which have tool tips attached for easy function identification. The toolbox also provides many help documentations in HTML (HyperText Markup Language) format as well as simple m-file templates to assist users in writing "model" files. Besides, it is capable of representing simulation results in various formats, such as text files or graphical displays for the purpose of results viewing and analysis. The file-handling capability of the toolbox also allows users to store or retrieve simulation data. The main features of the toolbox are summarized as follows.

- Support both single- and multi-objective optimization.
- The ability to focus on finding Pareto optimal solutions with powerful graphical displays.
- Fixed and dynamic population size for optimal representation of Pareto-front.
- Handle both hard and soft constraints.
- Goal and priority information for better support of decision-making in MO optimization.
- Powerful GUIs and easy linking to other program workbench.
- Step-by-step guidance to interface with "model" files.
- Comprehensive HTML help files and tutorials.
- Include a simple installation program.

An installation program is included in the toolbox for easy installation and setup of Matlab search paths. After the installation, the main toolbox GUI window can be called from Matlab workspace by the command "begin." This GUI can be minimized into a smaller window so that it occupies less space on the screen for easy access. Through the buttons on this GUI, many other toolbox GUI windows can be easily accessed including the help files and simulation setup files. There are two types of setup available in the toolbox, e.g., the "Quick" setup and the "Guided" setup. The GUI of "Quick" setup is shown in Fig. 3, which provides all simulation settings, such as the number of objectives and decision variables, generation and population size, selection strategy and so forth to be easily accessible within one GUI window. Fixed and dynamic niching schemes [25], [29] are

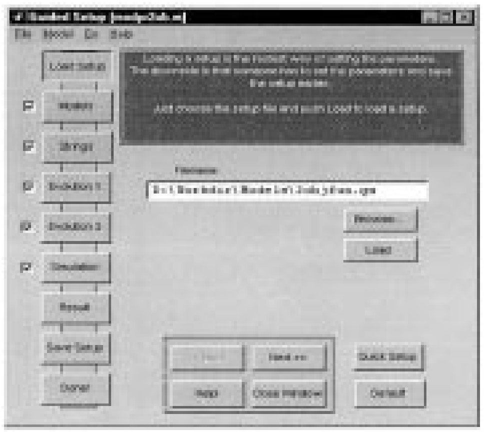


Fig. 4. "Guided" setup with loading setup details.

also included in this GUI, which allows sharing distance to be fixed or estimated adaptively based upon the online population distribution at each generation. The "Quick" setup also includes features to incorporate random strings or reuse strings from last evolution if necessary. Besides, the "model" file or Simulink [30] can be loaded directly through this GUI to achieve easy linking between MOEA toolbox and application setups.

For new users who have minor or no experience in setting up the parameters for simulation, an alternative "Guided" setup GUI window as shown in Fig. 4 is available to assist them by going through the whole setup process step-by-step with guidance information provided. The sparse multi-page arrangement in "Guided" setup also allows more information to be incorporated into the GUI window. Note that all parameter settings of "Quick" and "Guided" setups are interlinked, e.g., users may switch between these two GUI windows anytime as they wish, where all current settings in one setup window are automatically transferred to another. All settings in these two setups can also be saved into a file for reloading purpose. From the "Quick" setup or "Strings" setting in 'Guided' setup as shown in Fig. 5(a), the "Model Parameter Options" GUI window as shown in Fig. 5(b) can be opened to setup all decision variables involved in the optimization. Note that the toolbox does not have a limit on the number of decision variables that it can handle, although such limit may be imposed by the limited system resources. As shown in Fig. 5(b), settings over every ten decision variables can be easily accessed through the navigational controls in the toolbox.

Fig. 5(c) shows the "Summation Limits" GUI window, which is a primitive version of packet distribution method [31] for handling simple constraints with linear decision variables. This GUI window allows linear constraint specifications of the following format

$$\begin{aligned}
&T_L \le T = \sum_{i=1}^{n} p_i \le T_H \\
&p_{1L} \le p_1 \le p_{1H} \\
&p_{2L} \le p_2 \le p_{2H} \\
&\qquad\vdots \\
&p_{nL} \le p_n \le p_{nH}
\end{aligned} \tag{3}$$

where

T	summation of all decision variables;
T_L	lower summation limit;
T_H	upper summation limit;
p_i	decision variable i;
p_{iL}	lower limit for decision variable i;
p_{iH}	upper limit for decision variable i;
n	number of decision variables.

These constraints are automatically coded into the genetic structure of strings in the simulation, i.e., all strings in the population reproduced after the crossover and mutation operations remain as feasible strings, which avoids the need of repairing or rejecting any infeasible strings through specialized genetic operators. Fig. 6 shows the genetic structure as given by

$$P_i = P_{iL} + \sum_{j=1}^{g} \left(\begin{cases} s_j, & \text{if } a_j = i \\ 0, & \text{else} \end{cases} \right) \tag{4}$$

The genes in the chromosome represent packets of pre-specified sizes, $s_1 \ldots s_g$, and the value of each gene, $a_1 \ldots a_g$, determines the address of the decision variable that the packet is assigned to. For gene $j, a_j = i$ indicates that the packet of size s_j should contribute its value to the ith decision variable. Therefore the value of the ith decision variable is the sum of all the packets whose gene has the address that is corresponding to it. The packet distribution method has also been extended to handle nonlinear type of constraint optimization problems via the approach of angular transformation [32].

As shown in Fig. 7, the "Objective" setup window that specifies the setting of objective functions for the optimization problem can be called from either the "Quick" or '"Guided" setup. Similar to the setting of decision variables, there is no limit on the number of objectives although such limit may be imposed by the limited system resources. This GUI window is built to provide an easy and complete setup for each objective component involved in the optimization, which includes the support for setting of goal, priority and hard/soft constraints [25]. Note that the setting of a single specification (consists of a set of objective components with goal, priority and constraint) can also be extended to multiple specifications with logical "AND"/"OR" operations to accommodate more complex decision-making in the toolbox as shown in Fig. 7(c).

Fig. 8 shows the "Population Handling" GUI window of the MOEA toolbox. This GUI window allows strings or decision variables to be manually edited, removed or replaced by random ones as necessary. Any string in the population can be selected with its decision variables to be displayed on the left side of the window. At each generation, the fitness as well as the decision variables of the selected string is available for viewing or editing on the right half of the window. This GUI window is useful for providing online or off-line results analysis via strings editing, where users can have better interaction and understanding of the changing environment in the simulation. Note that strings for the entire population can also be created via the "Population Handling" GUI, which could then be saved and reloaded as an initial population for other setups.

The MOEA toolbox contains various GUIs for graphical displays, where simulation results can be represented in different

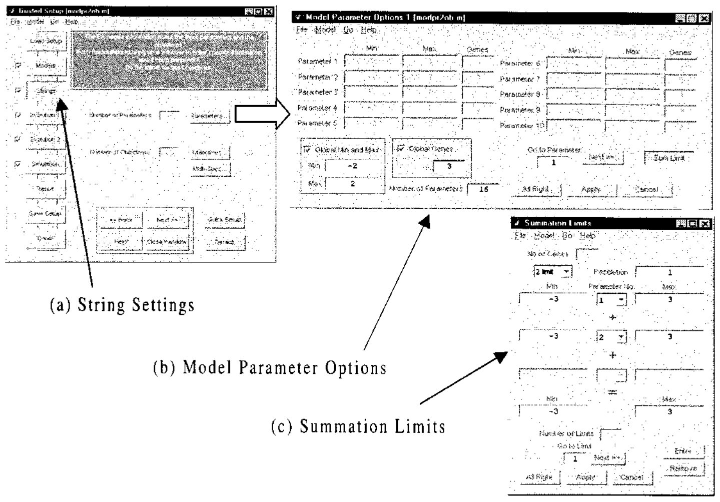


Fig. 5. Decision variable settings and linear constraint handling.

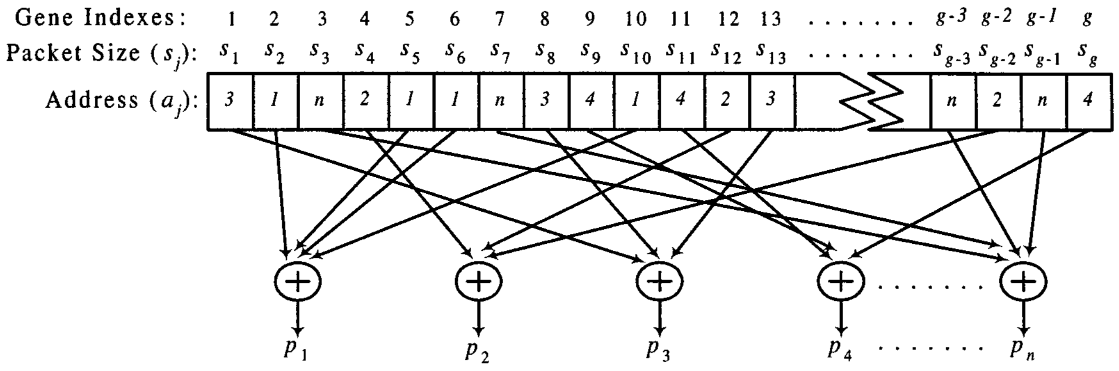
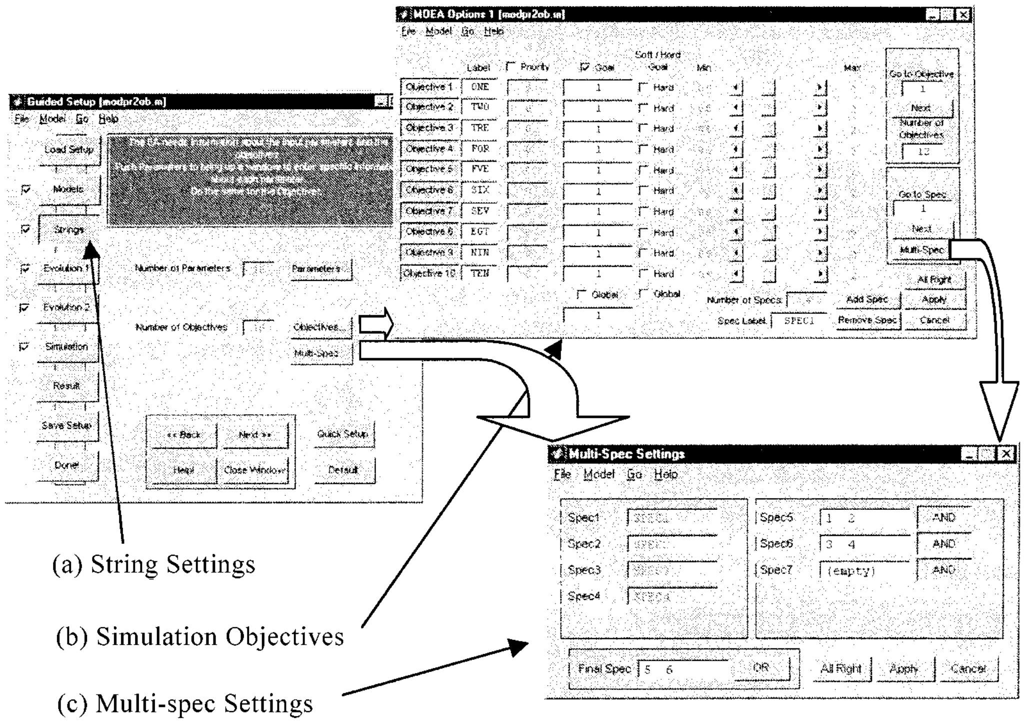

Gene Indexes: 1 2 3 4 5 6 7 8 9 10 11 12 13 .... g-3 g-2 g-1 g
Packet Size (s_j): s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} .... s_{g-3} s_{g-2} s_{g-1} s_g
Address (a_j): 3 1 n 2 1 1 n 3 4 1 4 2 3 n 2 n 4
p_1 p_2 p_3 p_4 .... p_n

Fig. 6. General structure of packet representation.



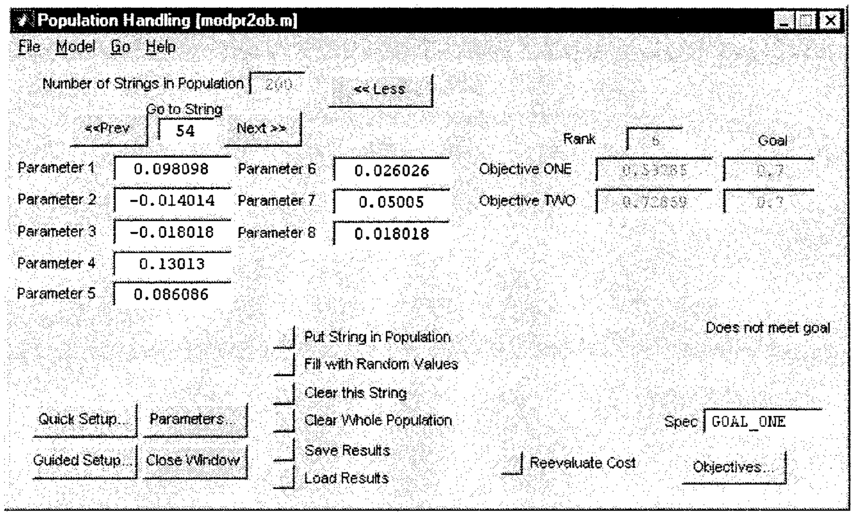
	Label	Priority	Goal	Soft / Hard Goal
Objective 1	ONE		1	Hard
Objective 2	TWO		1	Hard
Objective 3	TRE		1	Hard
Objective 4	FOR		1	Hard
Objective 5	FVE		1	Hard
Objective 6	SIX		1	Hard
Objective 7	SEV		1	Hard
Objective 8	EGT		1	Hard
Objective 9	NIN		1	Hard
Objective 10	TEN		1	Hard

Global: 1
Go to Objective: 1
Number of Objectives: 12
Go to Spec: 1
Spec Label: SPEC1

Spec1	SPEC1	Spec5	1 2	AND
Spec2	SPEC2	Spec6	3 4	AND
Spec3	SPEC3	Spec7	(empty)	AND
Spec4	SPEC4			

Final Spec: 5 6 OR

Fig. 7. ”Objective” setup window with setting of goal, priority and logical “AND”/”OR” operations.



Number of Strings in Population 200

Go to String 54

Parameter 1	0.098098	Parameter 6	0.026026
Parameter 2	-0.014014	Parameter 7	0.05005
Parameter 3	-0.018018	Parameter 8	0.018018
Parameter 4	0.13013		
Parameter 5	0.086086		

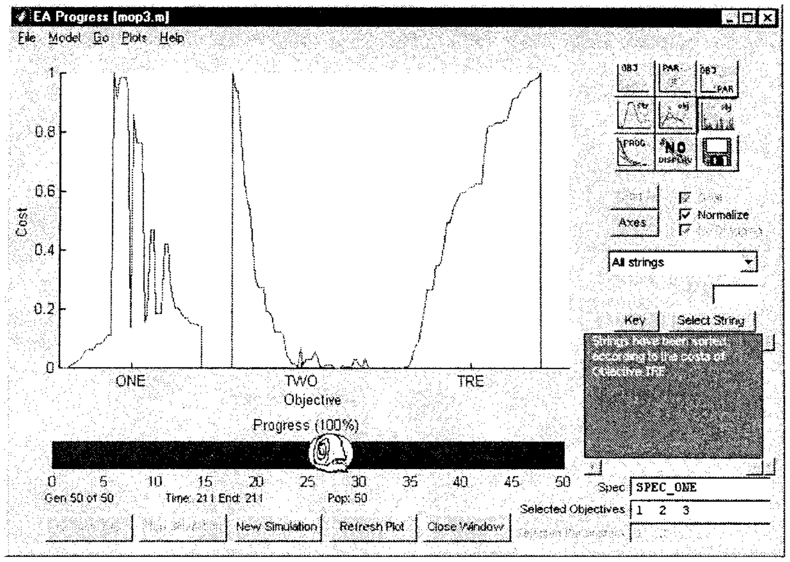
Rank 6

		Goal
Objective ONE	0.53785	0.7
Objective TWO	0.72859	0.7

Does not meet goal

Spec GOAL_ONE

Fig. 8. GUI window for strings manipulation.



Strings have been sorted according to the costs of Objective TRE

Gen 50 of 50 Time: 21 End: 21 Pop: 50

Spec SPEC_ONE

Selected Objectives 1 2 3

Fig. 9. Graphical displays of simulation results in MOEA toolbox.

plotting for graphical analysis or visualization. These plottings can be updated at each generation for which users can interactively manipulate controls to adjust the displays. One of such GUI windows is shown in Fig. 9, where strings can be arranged in an ascending order based on any selected optimization criteria. Fig. 10 shows the convergence trace for single- or multi-objective optimization as well as the number of strings meeting goal setting at each generation, which gives a quantitative idea of how the population is evolving over generations. For MO optimization, the convergence trace is measured by means of progress ratio [25]. In the sense of evolution progress toward the direction that is normal to the tradeoff surface formed by the current nondominated strings, the progress ratio $\mathrm{Pr}^{(n)}$ at generation n is defined as the ratio between the number of nondominated strings at generation n *dominating* the nondominated strings at generation $(n-1)$ over the total number of nondominated strings at generation n. This GUI also allows simulation data such as strings in a population, fitness of strings and progress ratio to be saved as "mat" file in Matlab or as "text" file to be loaded by external programs like Microsoft Excel.

The MOEA toolbox also comes with comprehensive online help files in HTML format, which can be easily accessed via button/link in each GUI window or menu bar as shown in Fig. 11. Whenever the help command is called, the relevant help document will be opened via the default Web browser in the system. The information contained in the help files include:

- General information on evolutionary algorithms and multiobjective optimization;
- Step-by-step demonstration of the toolbox;
- Guides to GUIs of the toolbox as well as to writing of Matlab and Simulink "model" files;
- List of possible error messages and ways of handling them.

D. Advanced Settings of MOEA Toolbox

The left side of Fig. 12 shows the GUI window for evolutionary parameter settings, where crossover and mutation rates can be set graphically. There are two methods of selection available in the toolbox, e.g., roulette wheel and tournament

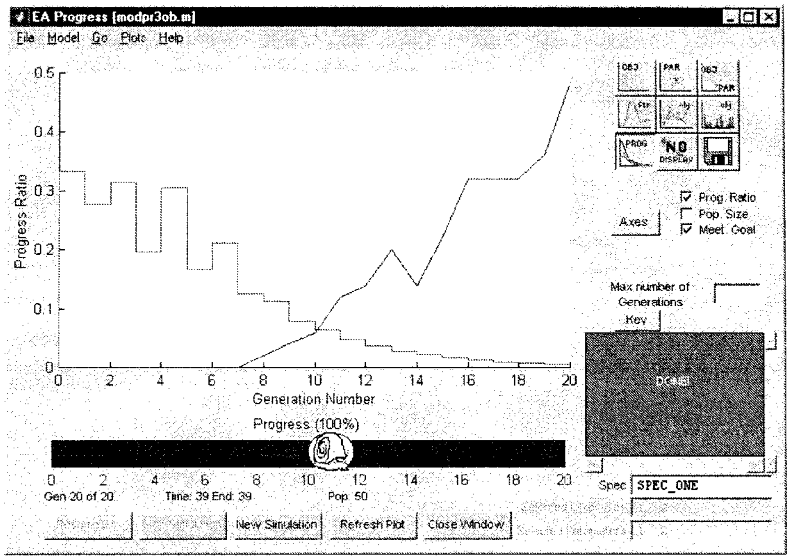


Fig. 10. Convergence trace for evolution progress.

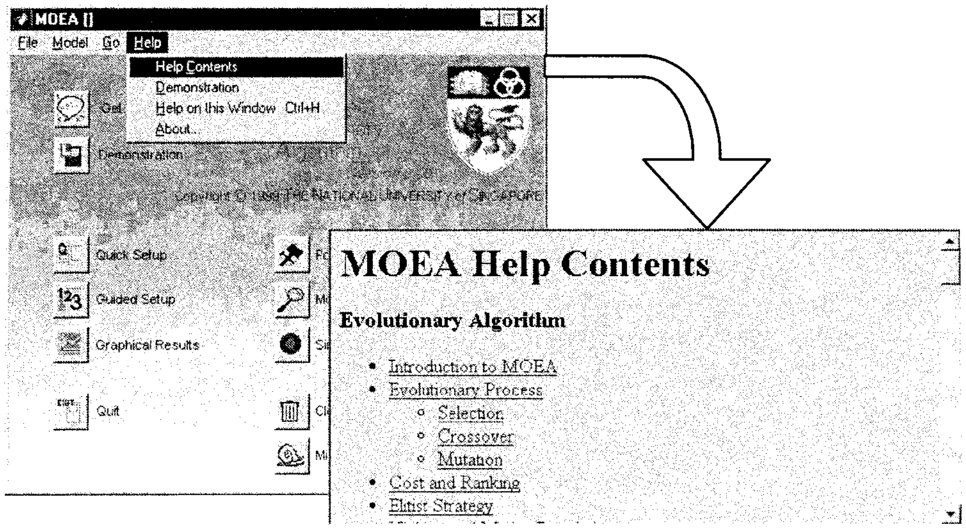
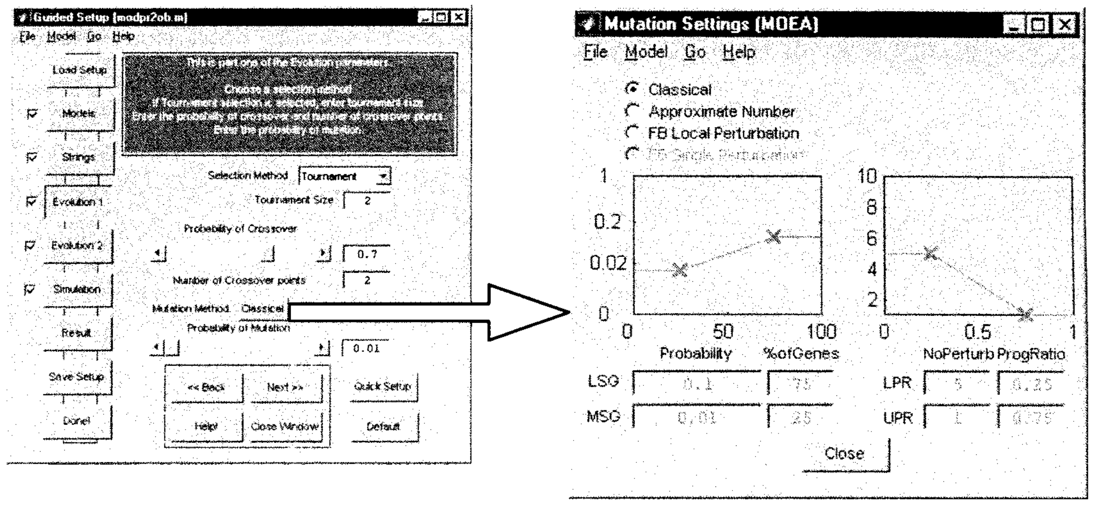


Fig. 11. Toolbox link from main window to help file contents.

selection schemes. Three types of mutation operator are provided in "Mutation Settings" GUI window as shown in the right side of Fig. 12. The first type is the classical mutation operator, where a gene is randomly mutated if a random generated number is smaller than the probability of mutation. The second type is the approximate-number mutation, where a number of randomly selected genes, equal to the rounded product of the total number of genes and the probability of mutation, are mutated randomly.

The third mutation method is called fuzzy boundary local perturbation (FBLP), which was proposed in [33] as a tool for reproducing strings to fill up discontinuities among nondominated strings in forming the Pareto-front for MO optimization. Each string in FBLP is perturbed in such a way that the resultant string from the perturbation is likely to be situated within a short distance from the original one. The number of perturbations for each string is related to the status of progress ratio [33], which is generally large for a small progress ratio and vice versa. A high progress ratio means that the performance of evolution is improving a lot and is likely to be far away from the tradeoff. A local search at this stage is less meaningful since the evolution is less likely to reproduce strings within or near to the tradeoff region. In contrast, the progress ratio is generally low when the evolution is approaching the tradeoff, and thus it is more meaningful to increase the number of perturbations at this stage in order to obtain more neighboring strings for better tradeoff representation. The setting of this relation is adjustable in the toolbox as shown in the right side of Fig. 12. When FBLP is performed on a string, mutation probability for each gene depends on its position in the respective decision variable. For each decision variable, the closer the gene to the LSG (Least Significant Gene), the higher the probability of mutation is assigned to this gene. By giving a higher chance of mutation for less significant genes, perturbed strings are likely to be located within a short distance from the original string, thereby fulfilling the purpose of local search in FBLP.


Guided Setup [modpr2ob.m]

This is part one of the Evolution parameters.

Choose a selection method.
If Tournament selection is selected, enter tournament size.
Enter the probability of crossover and number of crossover points.
Enter the probability of mutation.

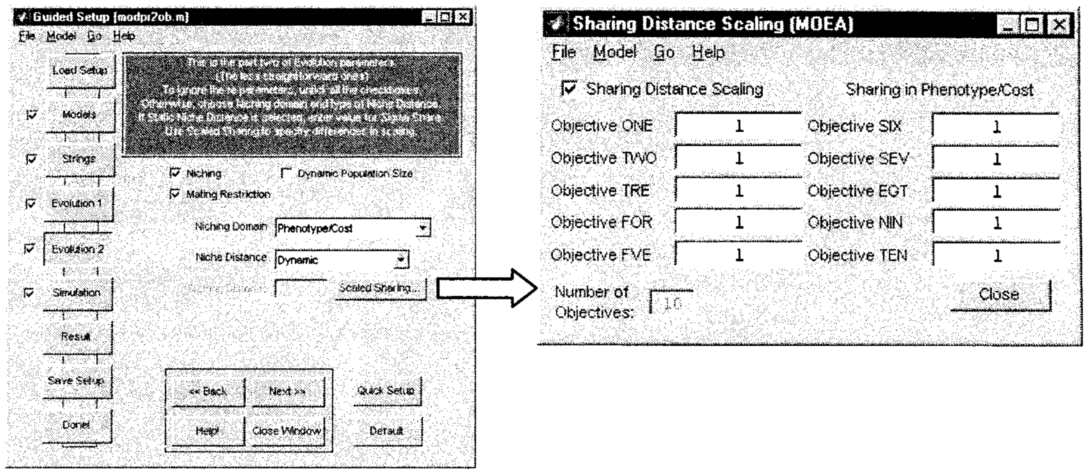
Selection Method: Tournament
Tournament Size: 2
Probability of Crossover: 0.7
Number of Crossover points: 2
Mutation Method: Classical
Probability of Mutation: 0.01

Mutation Settings (MOEA)

Classical (selected)
Approximate Number
FB Local Perturbation
FB Single Perturbation

LSG: 0.1, 75
MSG: 0.01, 25
LPR: 5, 0.25
UPR: 1, 0.75

Fig. 12. Settings of evolutionary operators and mutation types.


Guided Setup [modpr2ob.m]

This is the part two of Evolution parameters.
(The less straightforward ones)
To ignore these parameters, untick all the checkboxes.
Otherwise, choose Niching domain and type of Niche Distance.
If Static Niche Distance is selected, enter value for Sigma Share.
Use Scaled Sharing to specify differences in scaling.

[✓] Niching
[ ] Dynamic Population Size
[✓] Mating Restriction
Niching Domain: Phenotype/Cost
Niche Distance: Dynamic

Sharing Distance Scaling (MOEA)

[✓] Sharing Distance Scaling — Sharing in Phenotype/Cost

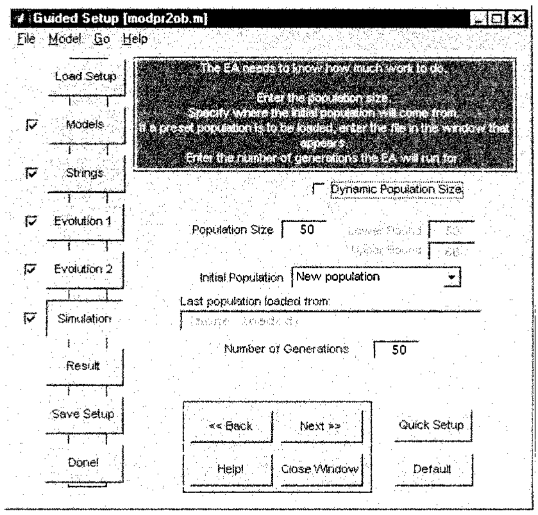
Objective ONE	1	Objective SIX	1
Objective TWO	1	Objective SEV	1
Objective TRE	1	Objective EGT	1
Objective FOR	1	Objective NIN	1
Objective FVE	1	Objective TEN	1

Number of Objectives: 10

Fig. 13. Settings of evolutionary operators and sharing distance scaling.

At the stage of "Evolution 2" in "Guided" setup as shown in the left side of Fig. 13, several advanced evolutionary settings are available. The "niche induction" technique by means of sharing function [34] is used to evolve an equally distributed population along the Pareto-front or multiple optima for evolutionary optimization. To avoid the need of *a-priori* knowledge to predefine a sharing distance as required by existing sharing methods, the toolbox includes a dynamic sharing scheme [25] which computes suitable sharing distance at each generation adaptively. Since each decision variable or objective may have different desired scaling values, the toolbox provides a "Niching Distance Scaling" GUI window where the scale of each decision variable or objective can be easily specified as shown in the right side of Fig. 13. Mating restriction [34], [35] is included in the toolbox to restrict mating of strings in order to prevent reproduction of highly unfit strings as well as to maintain the population diversity. If the mating restriction is enabled, two strings will be selected for mating if they are located within a certain distance, e.g., the sharing distance; otherwise a random string is selected for mating if no such strings are found.

The MOEA toolbox also allows users to load an initial population, generate a new population of random strings or use a combination of both before any simulations. The initial population can be loaded from a file generated from last simulation session or entered via the "Population Handling" GUI. As shown in Fig. 14, the feature of dynamic population size [33] is also included in the toolbox, which is particularly useful for automatically estimating an optimal population size at each generation so as to sufficiently explore the search space as well as to represent the Pareto-front effectively. Intuitively, it is hard to achieve a good evolution if the population size is too small due to insufficient exchange of genetic information. If the population size is too large, the evolution may take extra computational effort with greater demands on system resources and simulation time. The merit of dynamic population is that it avoids the need of presetting a constant population size that is usually obtained by repeating the simulation with different population size until a good solution is found. The toolbox also allows settings in the GUIs to be saved in a "setup" file for reference or use later in other simulations, besides having the feature of "crash backup"



The EA needs to know how much work to do.

Enter the population size.
Specify where the initial population will come from.
If a preset population is to be loaded, enter the file in the window that appears.
Enter the number of generations the EA will run for.

Dynamic Population Size

Population Size 50 Lower Bound 50 Upper Bound 50

Initial Population New population

Last population loaded from: (none loaded)

Number of Generations 50

Fig. 14. Population setting with optional feature of dynamic population size.

file that stores all simulation data at each generation for backup purpose.

E. Writing "Model" File

There are three types of user-written files in the toolbox, e.g., the "model" file, the "initiator" file and the "streog" file. According to users' optimization problem on-hand, the "model" file that specifies the objective function must be written in order to determine the fitness function of each string in the population before any simulation. The MOEA toolbox sends values of the decoded decision variables to this "model" file and expects a cost vector to be returned for each of the strings. The "model" file is also used in "Population Handling" GUI to allow any manual examination, modification or re-evaluation of strings of interest in a population. Besides providing help files in HTML format, several templates for writing the "model" files are included in the toolbox. There are also notes, guides and reminders in the form of comments in each template to assist users in writing the "model" file for his/her particular optimization problem. The "initiator" file is optional and is a function or a script that is run once at the beginning of the simulation. This "initiator" may be used to initiate a separate graphical display window or to initialize constants for the simulation. Similarly, the setting of "streog" file is also optional and is a function or a script that is run at the end of each generation, which can be useful for plotting simulation results in graphical display that is generated by the "initiator" or for producing backup data of the simulation.

Since the MOEA toolbox is written in Matlab, it is capable of running any "model" file created in Matlab as well as making use of the versatile functions and resources in Matlab. Users can also call their own C or Fortran "model" files from Matlab as if they are built-in functions. This Matlab callable C or Fortran program is defined as "mex" file [21], which is dynamically linked subroutine that Matlab interpreter can load and execute. Details of creating and executing "mex" file are available in Matlab application program interface guide [21]. The merit of "mex" file is that it executes faster than its m-file equivalent, and hence reduces the overall simulation time. Alternatively,

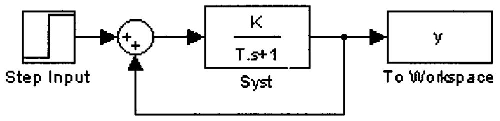


Fig. 15. Simulink model of a first-order system (filename: Fstorder).

"model" files can also be written as m-file and easily be compiled into "mex" file with the help of Matcom [28] for similar purpose.

The MOEA toolbox is capable of running a model built with Simulink, which is a GUI-based software in Matlab for modeling, simulation and analysis of dynamic systems. The toolbox provides a special template containing comments and sample codes that assist users in creating "model" files for running Simulink models. There are also various functions that allow m-files to modify the parameters of Simulink functions or to run the simulation in Simulink. The Simulink model can be loaded by typing its model name at the Matlab command window before running the simulation. An "initiator" file can also be used to load a Simulink model in a new window if none already exists. For example, consider a Simulink model with step response simulation of a first-order system as shown in Fig. 15. The model is saved as a file named "Fstorder.mdl," and the "initiator" file that opens this model consists of a command line "Fstorder".

In the model file, the following lines can be written to describe the system:

```
% A sample model file
  Time = 1 : 0.1 : 1000;
  num = [K];
  den = [T\ 1];
  set_param( "Fstorder/Syst," ...
  "Numerator," [ ' [ ' num2str(num) ' ] ' ], ...
    "Denominator," [ ' [ ' num2str(den) ' ] ' ])
sim("Fstorder," t, [ ], [ ]);
```

"Fstorder" is the name of Simulink "model" file; "Syst" is the name of blocks that represent the system; "Time" is the simulation time index; "K" and "T" are the parameters of "num" and "den" that define the first-order system, respectively. Note that after each "model" evaluation in Simulink, the step response data of the first-order system will be returned to the main function in Matlab, as desired.

F. MOEA Toolbox for Real-Time Optimization

The MOEA toolbox is well suited for global MO optimization where different noncommensurable or competing criteria can be optimized simultaneously without the need of a compromise function. From Simulink and real-time workspace toolbox [36], the evaluation module in Fig. 2 can be extended easily for online evaluation and hardware in-the-loop optimization, if desired. Without loss of generality, Fig. 16 illustrates an example of real-time control system design governed by the MOEA toolbox, as an extension from off-line CAMOO as shown in Fig. 2 to online real-time optimization.

In the extended evaluation module, Simulink is useful for control system modeling, simulation and controller design. The

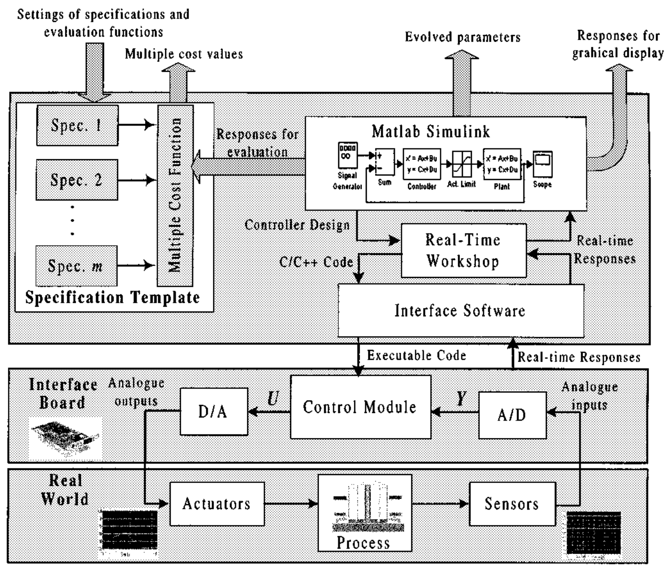


Fig. 16. Extended evaluation module for real-time optimization.

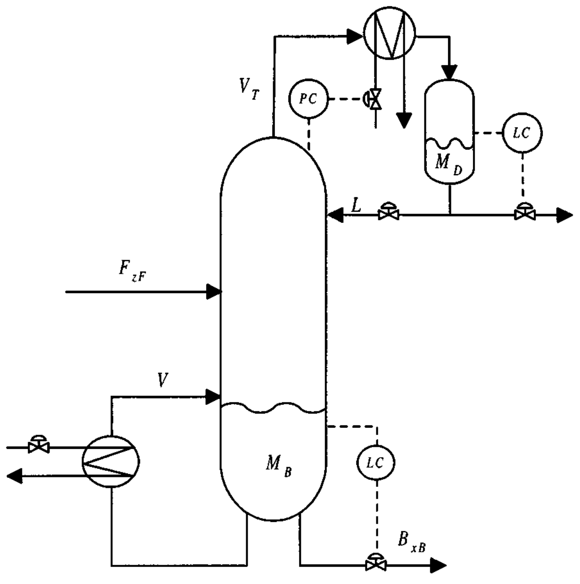


Fig. 17. Typical distillation column control system.

controller that is developed via the block diagram of Simulink can be compiled by real-time workshop toolbox into C/C++ codes, which are then compiled into executable codes for downloading onto the hardware. The interface software then initializes the hardware, loads the executable application codes onto the memory area and initiates the program execution to perform the real-time control application. At the same time, a set of plant data is fed back to the real-time workspace via the interface software for the evaluation of the objective functions. Intuitively, the overall design convergence of this architecture is guaranteed if the convergence of MOEA is faster than the dynamics of the process, which adds to the challenge of the MOEA toolbox.

Instead of using high-level programming like Matlab or Simulink block diagrams, fast and efficient low-level programs and interface boards can also be used to provide the linkage between MOEA toolbox and control applications. In this case, the interface software needs to allow controller parameters to be altered on the interface board in real-time within the toolbox environment, as well as to allow the real-time response data to be logged back onto the MOEA toolbox for fitness evaluations and graphical displays. As recommended in [37], this task can be conveniently realized with the aid of special hardware and software system such as dSPACE [38], MIRCOS [39] or TMS320C40 DSP industry standard card, using an appropriate analogue input/output module where the card can be sited in a PC to form a complete standalone dedicated control system.

III. Design Application of a Distillation Control System

To demonstrate the effectiveness and various features in MOEA toolbox, a practical control design optimization problem of multiple-input and multiple-output (MIMO) ill-conditioned distillation system is used as the case study in this section. It should be noted that the focus here is to illustrate how MOEA toolbox can easily assist the MO design optimization of a distillation control system. This control problem was originated from [40] and was studied later in [41]–[46]. The ordinary model of the distillation column is shown in Fig. 17, which consists of multiple inputs (flows: reflux L, boilup V, distillate D, bottom flow B and overhead vapor V_T) and multiple outputs (compositions and inventories: top composition y_D, bottom composition x_B, condenser holdup M_D, reboiler holdup M_B, pressure p).

This control problem usually has no inherent control limitations caused by RHP-zeros, but the plant has poles in or close to the origin and needs to be stabilized. The RGA-matrix may also have some large elements for high-purity separations [40]. Another complication posed in this design is that the composition measurements are often expensive and unreliable [41]. Consider a distillation process in "LV" configuration with two inputs (reflux L and boilup V), and two outputs (product compositions y_D and x_B). The five-state model of "LV" configuration with a state-space realization given as

$$G(s) = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{5}$$

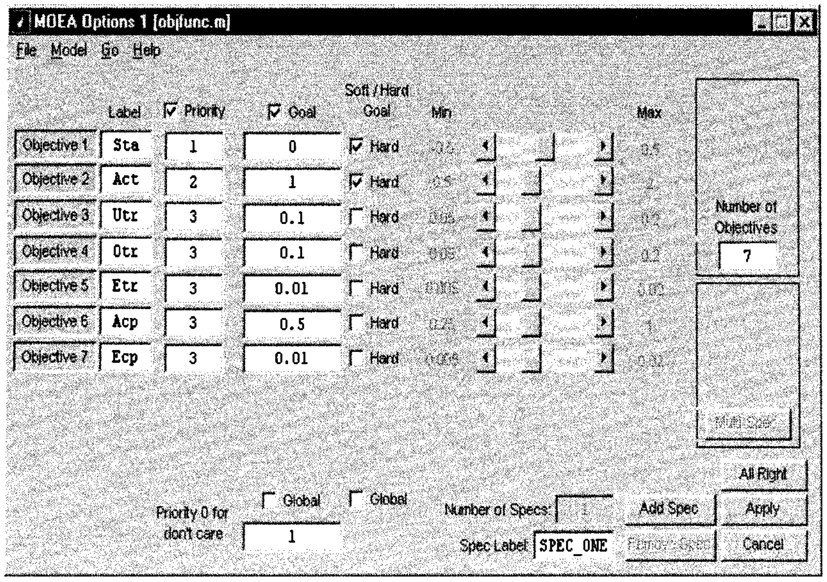
where $G(s)$ denotes the transfer matrix from the inputs (L and V) to the outputs (y_D and x_B) respectively. The matrix A, B, C and D are given in (6) shown at the bottom of the next page.

The above five-state model was obtained via model reduction of the original model with 82 states [41]. The process to be controlled is a distillation column with reflux flow and boilup as manipulated inputs and product compositions as outputs. The model has been scaled such that a magnitude of 1 corresponds to the following: 0.01 mole fraction units for each output (y_D and x_B), the nominal feed flow rate for the two inputs (L and V) and a 20% change for each disturbance (feed rate F and feed composition z_F).

The set of design specifications for this distillation system are listed in Table I, which aims to obtain a distillation control system that meets a set of transient and steady-state performance requirements, while satisfying certain system constraints such as actuator saturation. In Table I, y_1 and y_2 are system outputs,

TABLE I
DESIGN SPECIFICATIONS FOR THE MIMO ILL-CONDITIONED DISTILLATION SYSTEM

Design specification		Goal	Constraint	Priority
1. Closed-loop Stability (Sta)	Count{[Real(CL poles)] > 0}	0	Hard	1
2. Actuator saturation (Act)	Max($\lvert u_1\rvert$, $\lvert u_2\rvert$)	1	Hard	2
3. Tracking undershoot (Utr)	1-min(y_1, y_2) for $t > t_{overshoot}$	0.1	Soft	3
4. Tracking overshoot (Otr)	max(y_1, y_2) for all t	0.1	Soft	3
5. Tracking steady-state error (Etr)	max($\lvert y_1(\infty)-1\rvert$, $\lvert y_2(\infty)-1\rvert$)	0.01	Soft	3
6. Coupling amplitude (Acp)	max($\lvert y_1\rvert$, $\lvert y_2\rvert$) for all t	0.5	Soft	3
7. Coupling steady-state error (Ecp)	max($\lvert y_1(\infty)-1\rvert$, $\lvert y_2(\infty)-1\rvert$)	0.01	Soft	3



	Label	[✓] Priority	[✓] Goal	Soft / Hard Goal	Min	Max
Objective 1	Sta	1	0	[✓] Hard	-0.5	0.5
Objective 2	Act	2	1	[✓] Hard	0.5	2
Objective 3	Utr	3	0.1	Hard	0.05	0.2
Objective 4	Otr	3	0.1	Hard	0.05	0.2
Objective 5	Etr	3	0.01	Hard	0.005	0.02
Objective 6	Acp	3	0.5	Hard	0.25	1
Objective 7	Ecp	3	0.01	Hard	0.005	0.02

Number of Objectives: 7

Priority 0 for don't care: 1

Global   Global

Number of Specs: 1

Spec Label: SPEC_ONE

Fig. 18. "Objective Setup" GUI window for the distillation control problem.

while u_1 and u_2 are actuator outputs of the system. These design specifications, treated as the design objectives in MO optimization, can be easily set via the "Objective Setup" GUI in the toolbox as shown in Fig. 18. The underlying aim of setting the priorities in the last column of Table I is to obtain a controller that first stabilizes the system within the actuator saturation limit for hardware implementation. Note that the actuator saturation is set as a hard constraint reflecting the hard limit of this performance requirement, which requires no further minimization if the control actions u is within the saturation limit. Having fulfilled these requirements, the system should also satisfy some time domain specifications as defined by the transient and steady-state responses. Although determination of the priority settings may be a subjective matter and depends on the performance requirements, ranking the priorities is only optional and can be ignored for a "minimum-commitment" design [47]. If, however, an engineer commits himself to prioritizing the objectives, it is a much easier task than pre-weighting the different design specifications as required by other objective function aggregation approaches.

Fig. 19 shows the overall design block diagram of the distillation control system, where R is the command signal, E the error signal, D the disturbance signal and Y the plant output response. The design task here is to optimize the controller pa-

$$A = \begin{bmatrix} -0.005\,131 & 0 & 0 & 0 & 0 \\ 0 & -0.073\,66 & 0 & 0 & 0 \\ 0 & 0 & -0.1829 & 0 & 0 \\ 0 & 0 & 0 & -0.4620 & 0.9895 \\ 0 & 0 & 0 & -0.9895 & -0.4620 \end{bmatrix}, \quad B = \begin{bmatrix} -0.629 & 0.624 \\ 0.055 & -0.172 \\ 0.030 & -0.108 \\ -0.186 & -0.139 \\ -1.23 & -0.056 \end{bmatrix}$$

$$C = \begin{bmatrix} -0.7223 & -0.5170 & 0.3386 & -0.1633 & 0.1121 \\ -0.8913 & 0.4728 & 0.9876 & 0.8425 & 0.2186 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{6}$$

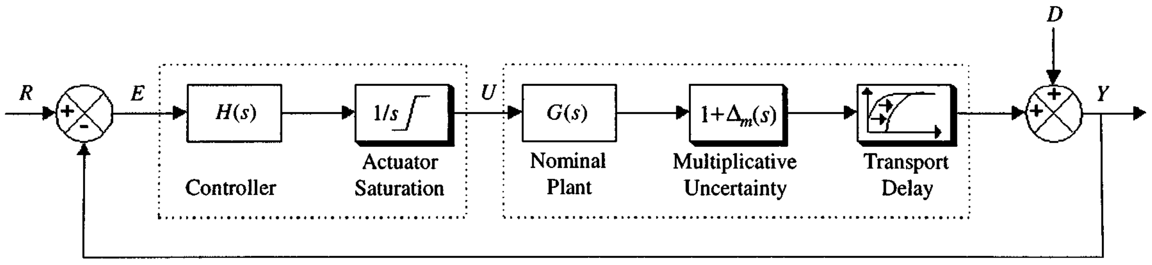


Fig. 19. Output feedback control system.

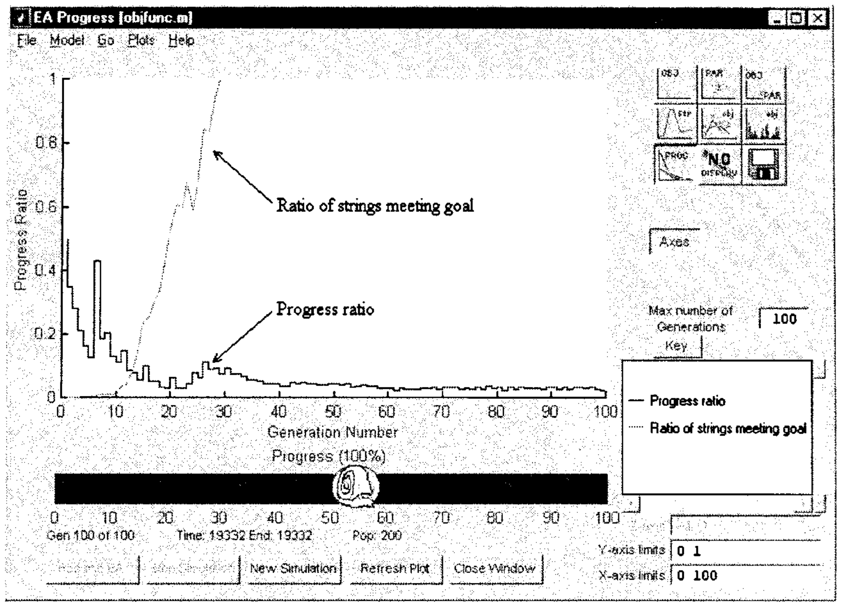


Fig. 20. Graphical displays of progress ratio and ratio of strings meeting goal.

rameters of $H(s)$ for the distillation column $G(s)$ so that it satisfies all design specifications as listed in Table I. Since MOEA toolbox is developed under the Matlab programming environment, users do not need to build the "model" files from scratch, i.e., any function libraries from any relevant Matlab toolboxes can be utilized directly for this purpose. For example, the control system toolbox in Matlab was utilized in this problem to define the complete MIMO distillation control system, without the need of writing the entire control block diagram or simulation program.

A full-matrix controller structure with simple 1st-order transfer function is adopted for this distillation control problem, which results in a total number of 16 controller parameters or decision variables given as

$$H(s) = \begin{bmatrix} H_{11}(s) & H_{12}(s) \\ H_{21}(s) & H_{22}(s) \end{bmatrix}; \quad \text{where}$$
$$H_{i,j}(s) = \frac{p_{3,i,j}S + p_{2,i,j}}{p_{1,i,j}S + p_{0,i,j}}, \quad i,j \in \{1,2\}. \tag{7}$$

Primary settings of the evolutionary design optimization are shown in Fig. 3, which include the number of design objectives and decision variables, generation and population size, selection strategy and so forth. Graphical displays in the toolbox can be used to observe the performance of the evolution, such as the convergence trace in the senses of progress ratio as well as the ratio of strings meeting goal (ratio of number of strings meeting goal to the population size). As illustrated in Fig. 20, the evolution begins with a zero ratio of strings meeting the goal. This value grows significantly from generations 10 to 30 and saturates at the value of one indicating all strings had met the goal setting. This graph shows that all controllers in the final generation satisfy all the design specifications as listed in Table I. It can also be observed in Fig. 20 that the progress ratio of the optimization is relatively high at the initial stage and decreases asymptotically toward zero as the evolution proceeds or as the population gets closer to the global tradeoff surface. This convergence feature is useful as an effective performance measure or stopping criterion for MO optimization.

The tradeoff graph for some of the evolved controllers is illustrated in Fig. 21, where each line represents a solution found by the optimization. The heavily crossing lines in Fig. 21 suggests that the solutions are nondominated and tradeoff against each other. To further illustrate the relationship among different specifications, the nondominated strings are plotted in Fig. 22 in terms of objective 4 (Otr) and objective 7 (Ecp). As can be seen, although all the plotted strings are nondominated to each other for objectives 1 to 7, they are not necessary tradeoff to each other with respect to objectives 4 and 7 since the tradeoff may

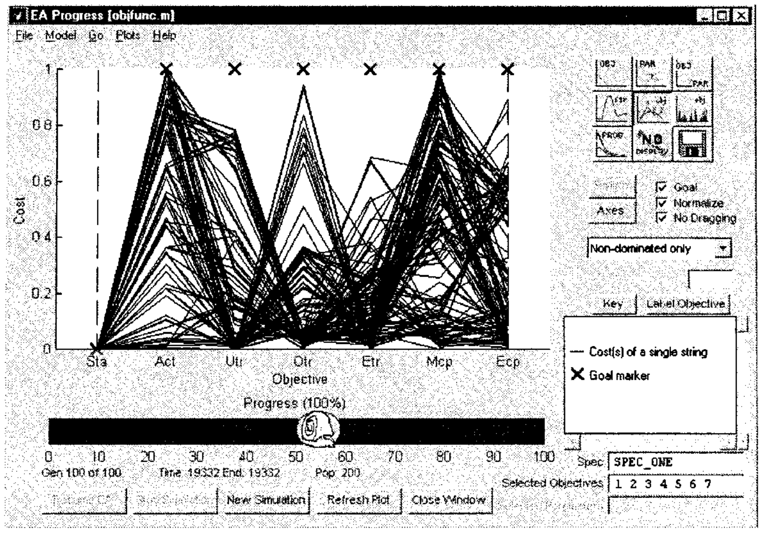


Fig. 21. Tradeoff graph for the design of MIMO ill-conditioned distillation control system.

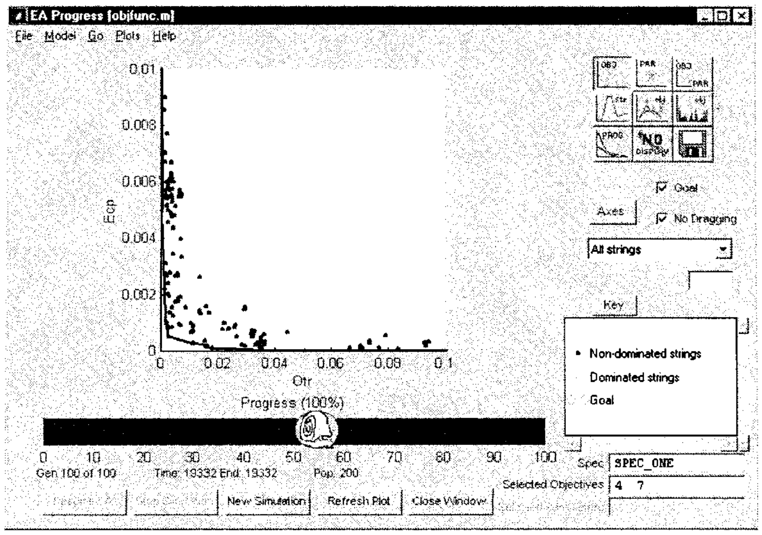


Fig. 22. Population distribution in the objective domain of Otr (4) and Ecp (7).

occur at other pair(s) of objective functions. If only strings that are tradeoff to each other with respect to objectives 4 and 7 are plotted, it can be observed that the tracking overshoot (Otr) performance deteriorates as more stringent bounds on the coupling steady-state error (Ecp) are demanded as shown by the solid line in Fig. 22. Further investigations between any other objectives can also be carried out in a similar manner.

The quantitative assessments of correlation coefficient and the slope of the least-square line can be used to provide statistical information of the evolving decision variables toward the objective components. For any decision variable x_j and objective component f_k, the correlation coefficient $r_{k,j}$ and slope of the least-squares line $m_{k,j}$ for a population size of N are defined as [48]

$$r_{k,j} = \frac{S_{xf}}{\sqrt{S_{xx}}\sqrt{S_{ff}}} \quad \text{and,} \quad m_{k,j} = \frac{S_{xf}}{\sqrt{S_{xx}}} \tag{8}$$

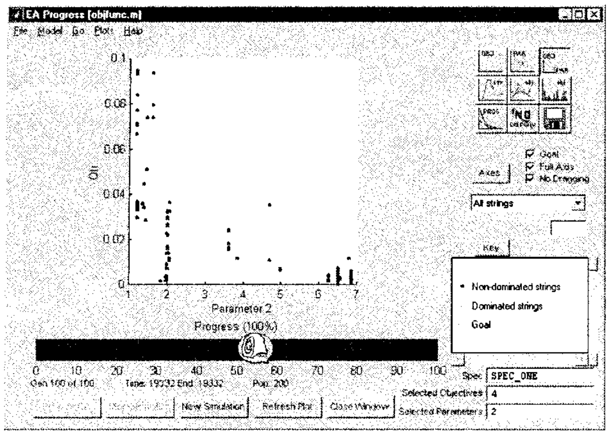
where

$$S_{xx} = \sum_{i=1}^{N} x_{i,j}^2 - \frac{1}{N}\left(\sum_{i=1}^{N} x_{i,j}\right)^2 \tag{9}$$

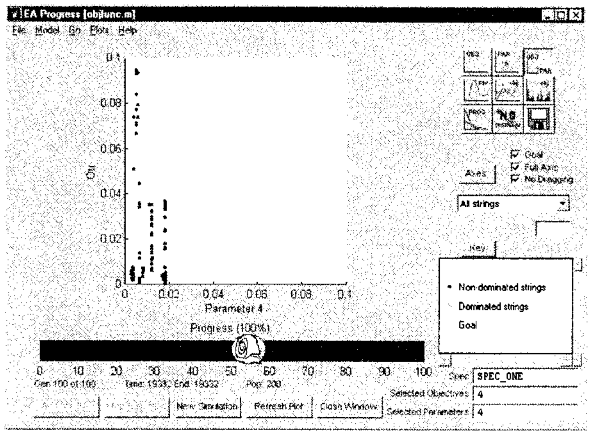
$$S_{ff} = \sum_{i=1}^{N} f_{i,k}^2 - \frac{1}{N}\left(\sum_{i=1}^{N} f_{i,k}\right)^2 \quad \text{and} \tag{10}$$

$$S_{xf} = \sum_{i=1}^{N} x_{i,j} f_{i,k} - \frac{1}{N}\left(\sum_{i=1}^{N} x_{i,j}\right)\left(\sum_{i=1}^{N} f_{i,k}\right). \tag{11}$$

The correlation coefficient $r_{k,j}$ gives a quantitative measure of how strongly a decision variable x_j and an objective component f_k is related. The value of $r_{k,j}$ is between -1 and $+1$. A value near to the upper limit of $+1$ indicates a substantial positive relationship, whereas an $r_{k,j}$ close to the lower limit

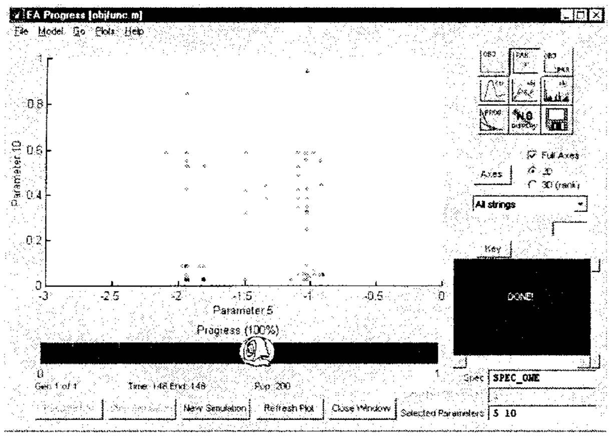


(a)

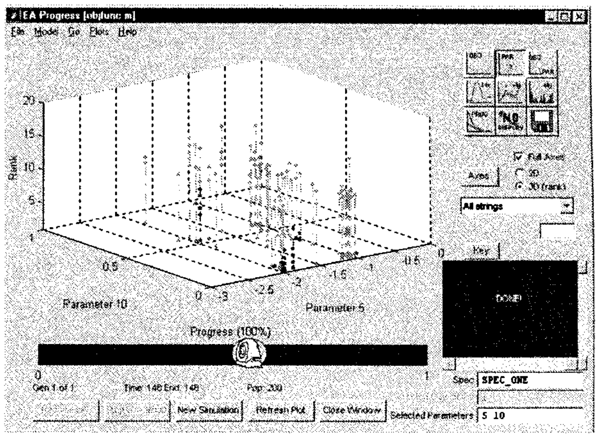


(b)

Fig. 23. Analysis of decision variables in correlation and sensitivity. (a) Population distribution of $\{x_2, f_4\}$ and (b) population distribution of $\{x_4, f_4\}$.



(a)



(b)

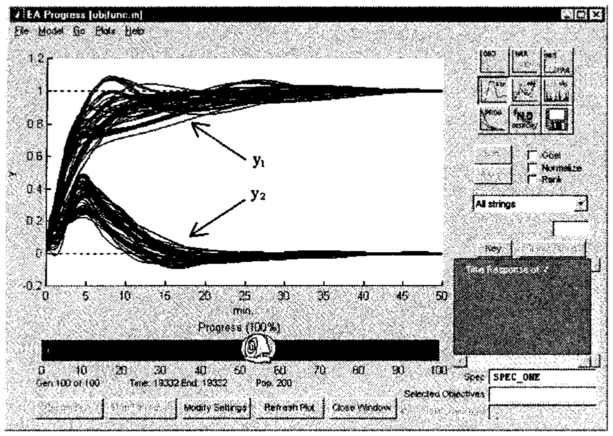
Fig. 24. Distribution of the controller parameters: (a) 2-D and (b) 3-D.

of -1 suggests a prominent negative relationship. In the case where decision variable x_j and objective component f_k are not correlated to each other, the value of $r_{k,j} = 0$ is found. For the slope of the least-squares line $m_{k,j}$, its magnitude represents the measure of sensitivity of the decision variable x_j to the objective component f_k. The larger the magnitude of $m_{k,j}$, the more sensitive the decision variable is.

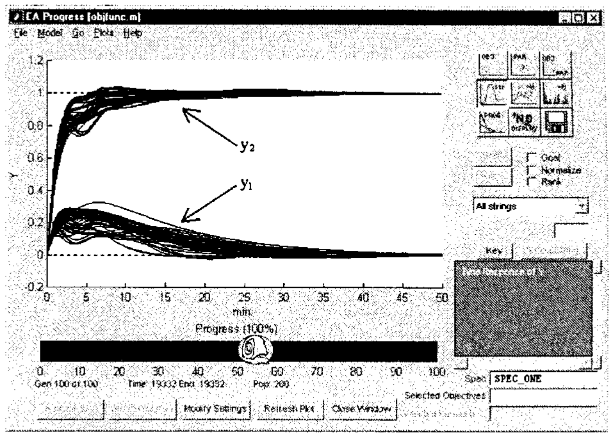
For example, consider the population distribution of the optimized controller parameters for the distillation control problem, the $r_{k,j}$ of $\{x_2, f_4\}$ and $\{x_4, f_4\}$ are $-0.6720(r_{2,4})$ and $-0.1956(r_{4,4})$, respectively; and their magnitude of $m_{k,j}$ are $0.0065(m_{2,4})$ and $-0.7596(m_{4,4})$, respectively. This shows that with respect to objective component f_4 (tracking overshoot or Otr), decision variable x_2 is more correlated than x_4; and both decision variables have a negative relationship with f_4. Therefore slight increment of decision variable x_2 should be given more concern than the increment of x_4 when performing manual reduction of f_4. On the other hand, as compared to x_2, the decision variable x_4 is more sensitive to the objective f_4. This indicates that a little variation of x_4 will lead to a large change in f_4. The visual impression of the correlation and sensitivity of $\{x_2, f_4\}$ and $\{x_4, f_4\}$ is shown in Fig. 23, which is useful for designers to manually change the controller parameters to achieve better closed-loop performance according to his/her particular needs.

The MOEA toolbox also supports other types of plotting in 2- or 3-dimension, such as the graph of decision variable versus decision variable as shown in Fig. 24. In the 3-dimensional plot, the z-axis represents the rank value where smaller rank implies better or fitter candidate string. These graphical displays are useful for better understanding and visualization of decision variable distributions as well as contribution of each decision variable to the overall optimization performance. These tools are also helpful in "Population Handling" GUI window which allows further manual modification or revaluation of any strings during or after the evolution process, as one desires. Fig. 25 shows the transient and steady-state responses of tracking and regulation performances for both channels in the system with two patterns of command signals R. It can be seen that all the time-domain performance requirements as specified by objectives of 3–7 in Table I have been met successfully.

To illustrate the robustness of the distillation system in the presence of disturbance, a sinusoidal input acting as the disturbance signal (D in Fig. 19) was applied to the system. The dis-

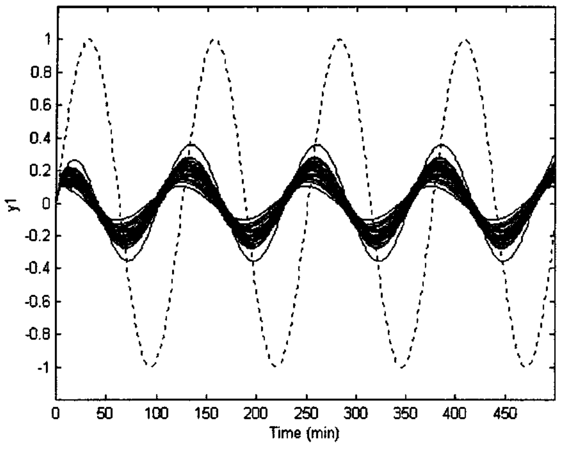


(a)

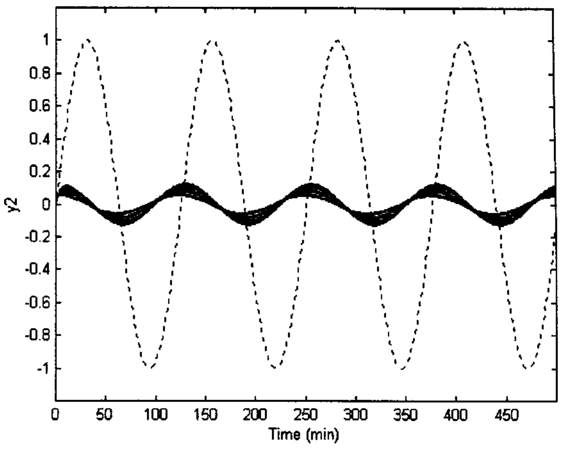


(b)

Fig. 25. Toolbox optimized output responses for the MIMO distillation system. (a) Output response for command signal $= [1\ 0]^T$ and (b) output response for command signal $= [0\ 1]^T$.



(a)



(b)

Fig. 26. Sinusoidal disturbance and its attenuated signals for the MIMO distillation system. (a) Disturbance attenuation for output y_1 and (b) disturbance attenuation for output y_2.

turbance input has an amplitude and angular frequency of 1 volt and 0.05 rad/s, respectively. The sinusoidal and its attenuated signal for all the Pareto optimal controllers are shown by the dashed and solid line in Fig. 26, respectively. Clearly, the disturbance has been attenuated substantially, with about five and ten times in gain reduction of the original sinusoidal for output y_1 and y_2, respectively.

Another powerful feature of the toolbox is that all the goal, priority and constraint settings can be conveniently examined and modified at any time during the evolution process. This can be easily performed with the embedded GUIs through three simple steps: pausing the evolution process, changing the settings and resuming the optimization process. For example, user may want to change the goal setting for actuator limit (Act) from 1 volt to 0.2 volt after a certain number of generations. Fig. 27 illustrates the effects of the evolution process upon further modification of this goal setting after the tradeoff shown in Fig. 21. Due to the sudden change of a tighter goal setting, none of the strings manage to meet all design specifications as shown in Fig. 27(a). After continuing the evolution for another two generations, the tradeoff moves toward satisfying the actuator limit (Act) at the expense of minor performance degradation for other objectives as shown in Fig. 27(b). In Fig. 27(c), the evolution continues and again leads to the satisfaction of all the goal settings by having less room for further improvements of other objectives (e.g., the fifth and seventh objectives) or having less Pareto optimal solutions as compared to the tradeoff found in Fig. 21. Clearly, this online interaction feature is useful where users can monitor or modify the design at any time during the evolution so as to suit their special needs, without the need of restarting the entire design or evolution cycles.

IV. Performance Comparisons of MOEA Toolbox

In this section, performance of the MOEA algorithm in the toolbox and other seven evolutionary MO optimization methods are compared upon a benchmark MO optimization problem. The biased-space minimization problem proposed in [49] is used

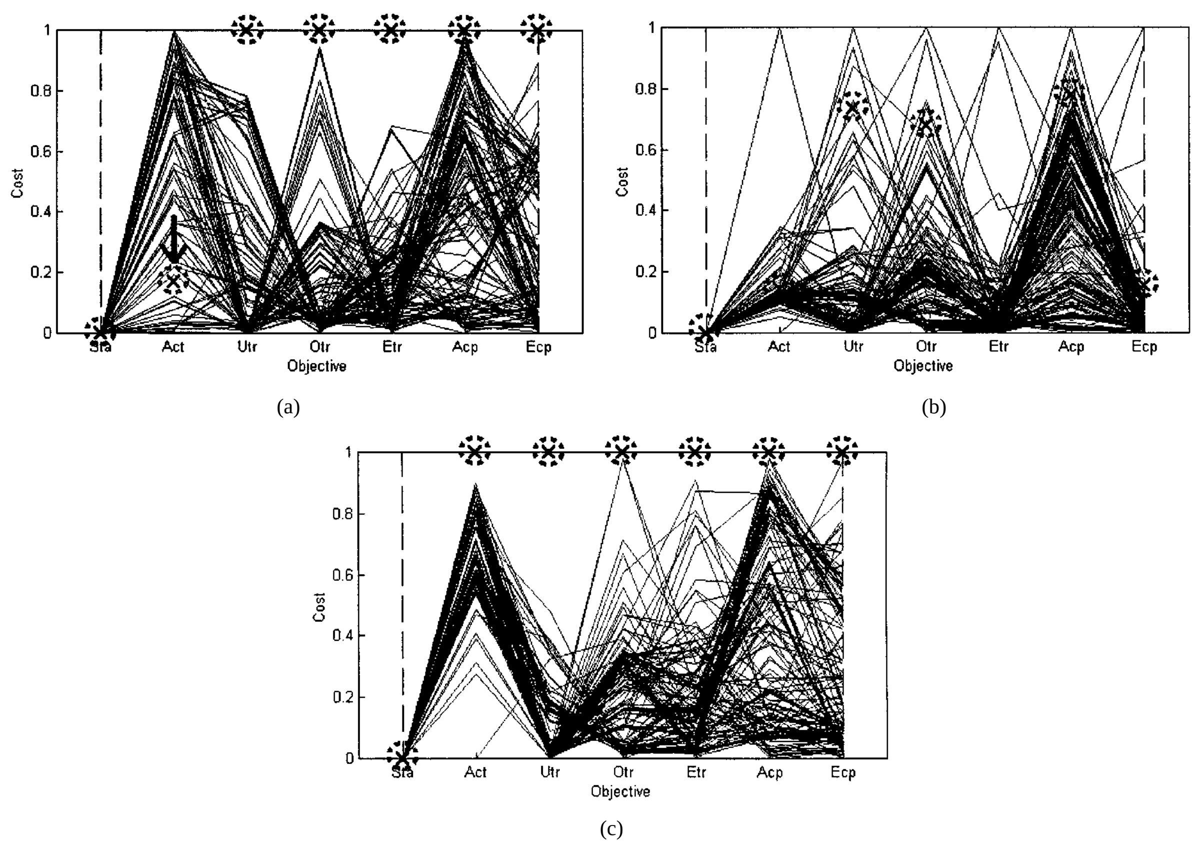


Fig. 27. Effects of the evolution upon online modification of goal setting, (a) Stringent the goal setting of Act from 1volt to 0.2 volt, (b) After two generations, (c) After another two generations.

here, which is to minimize a two-objective function and is mathematically defined as:

$$f_1(X) = x_1 \tag{12}$$

$$g(X) = g_{\min} + (g_{\max} - g_{\min}) \times \left(\frac{\sum_{i=2}^{N} x_i - \sum_{i=2}^{N} x_i^{\min}}{\sum_{i=2}^{N} x_i^{\max} - \sum_{i=2}^{N} x_i^{\min}} \right)^{\gamma} \tag{13}$$

$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2 \tag{14}$$

$$f_2(X) = g(X) \times h(f_1, g) \tag{15}$$

where the values $x_i^{\min}$ and $x_i^{\max}$ is minimum and maximum value of the variable x_i, while $g_{\min}$ and $g_{\max}$ is the minimum and maximum value that the function g can take. Pareto-optimal region occurs when g takes the value of $g_{\min}$, which happens when $x_i = x_i^{\min}\ \forall i = 2, \ldots, N$ [49]. As shown in Fig. 28, the shaded region represents the unfeasible space in the objective domain while the bold line is the Pareto-optimal curve for the two-objective biased-space minimization problem. This test function is chosen since it has a large and nonlinear tradeoff curve that challenges the MO evolutionary algorithm's ability to find and maintain the entire Pareto-front uniformly. Besides, qualitative optimization performance for this problem can be easily visualized and compared. As stated in [49], the difficulty of this MO optimization problem can be introduced by the parameter γ which controls the bias in the search space. The density of solution away from the Pareto-front is large when $\gamma < 1$.

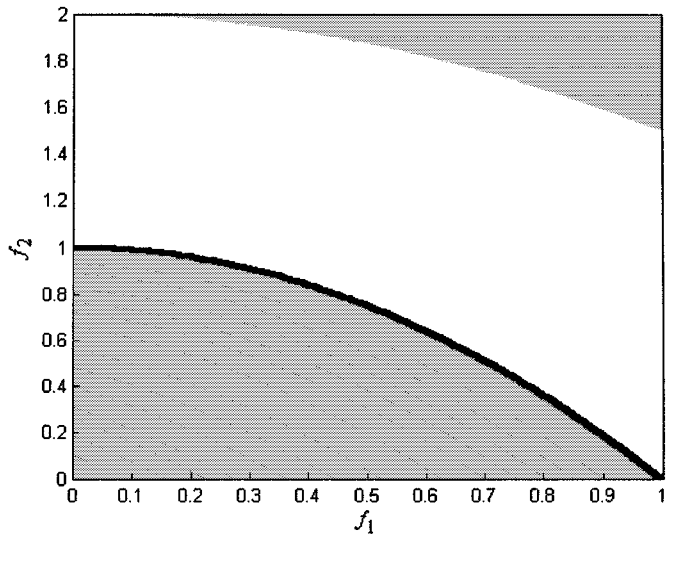


Fig. 28. Pareto-optimal curve in the objective domain.

Therefore random-like search methods are likely to face difficulties in finding the tradeoff in this problem. Similar to [49], the parameter values of $g_{\min} = 1, g_{\max} = 2, \gamma = 0.25, N = 6, x_i^{\min} = 0$ and $x_i^{\min} = 1\ \forall i = 1, 2, \ldots, N$ are used for the study here.

There are many performance measures for MO optimization proposed in the literatures, e.g., [50]–[52]. In general, these measures compare the performance of evolutionary optimization in: (1) to attain the Pareto-front, e.g., C from [52], error

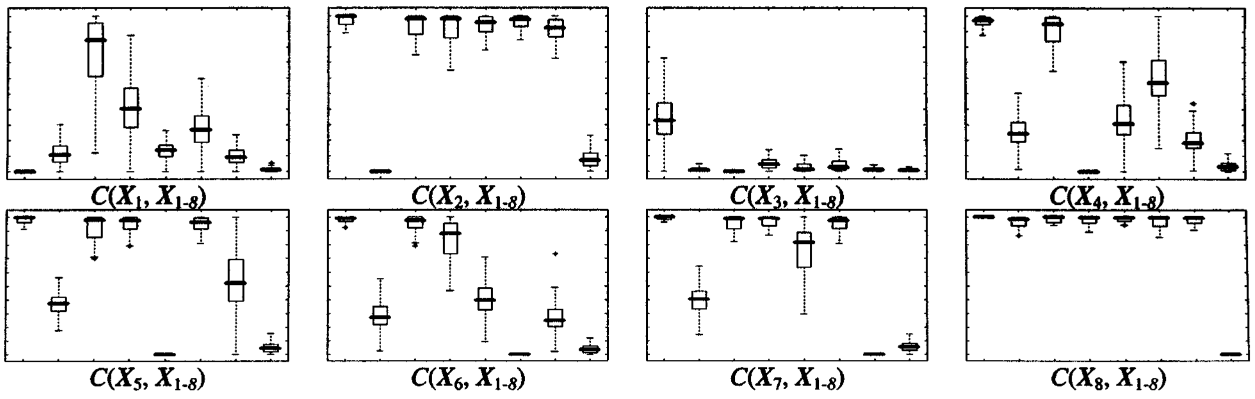


Fig. 29. Box plot based on C measure. Each rectangle, refers to the measure of $C(X_i, X_{1-8})$, represented by box plots arranged left to right, between i algorithm and algorithms ranging from 1 to 8. The scale is -0.05 at the bottom and 1.05 at the top of each rectangle.

ratio and generational distance from [51]; (2) to spread the nondominated strings along the available Pareto-front, e.g., size of space covered *SSC* from [52], Spread t from [27]. The measures of C and *SSC* proposed by Zitzler and Thiele [52] are employed here to access the performance of different evolutionary algorithms, since these two measures consider both cases (1) and (2), and are generally applicable without the need of pre-finding the actual or best found Pareto optimal solutions.

The MOEA algorithm in the toolbox has been compared with various evolutionary MO optimization methods, which include (1) VEGA from [53]; (2) MIMOGA from [54]; (3) HLGA from [55]; (4) NPGA from [3]; (5) MOGA from [7]; (6) NSGA from [27]; and (7) SPEA from [52]. These methods are chosen for comparison since they have been frequently referenced by other researchers. In addition, these algorithms consist of different type of evolutionary methods for MO optimization and some of them have been applied to real-world applications, especially MOGA. These algorithms have been indexed according to the above sequence, where the MOEA algorithm is assigned the last index number (8).

Since the control parameter settings may be different from one algorithm to another, the setting of these parameter values are based upon two principles in this study: (1) the value of the parameters that are commonly used by several algorithms are identical to those algorithms and, (2) the value of the parameters that are used in specific algorithms are decided based upon the recommended values from their original literature. Fitness sharing [7] is applied to all methods that use sharing scheme in their algorithms. The sharing distance for MOGA, NSGA, NPGA and HLGA are set as 0.01 in the normalized space since the population size was set at 100. No sharing parameter settings are required by SPEA [52] and MOEA in the toolbox. The MOEA applies dynamic sharing scheme where sharing distance can be computed adaptively at each generation. Tournament selection scheme with tournament size of 2 is used in MOGA, SPEA and MOEA as suggested in their original literatures. The Pareto tournament selection scheme with $t_{\text{dom}} = 10\%$ of the population size was used in NPGA for tight and complete population distribution as recommended in [26].

All methods under comparison were implemented with the same common sub-functions in Matlab on an Intel Pentium II 450 MHz processor. Each of the simulation was terminated automatically at the time of 120 sec, in the same platform that is free from other computation or being interrupted by other programs. Here, 30 independent simulation runs have been performed for each method so as to study the consistency and robustness of the algorithms. Note that randomly generated population with an initial population size of 100 is used for all the 30 simulations except MIMOGA and SPEA. For MIMOGA and SPEA, combinations of $\{P, P'\}$, namely $\{80, 20\}$, where $P + P' = 100$, is used as similar to the setting in [52]. For the measure of *SSC* that works on the normalized fitness space, the standard ranges in fitness space of $0 \leq f_1 \leq 1$ and $0 \leq f_2 \leq 2$ are chosen, which are determined based upon the space covered by the Pareto-front.

The performance measure of $C(\boldsymbol{X}_i, \boldsymbol{X}_j)$ for the comparison sets between algorithms i and j where, $i, j = 1, 2, \ldots 8$, are shown in Fig. 29. Box plots [56] are used to summarize the sample distributions of 30 independent runs per each case, which has been applied in [52] to visualize the distribution of simulation data efficiently. Each box plot represents the distribution of a sample population where a thick horizontal line within the box encodes the median, while the upper and lower ends of the box are the upper and lower quartiles. Dashed appendages illustrate the spread and shape of distribution, and dots represent the outside values. In each rectangle containing box plots, the sequence of box plots from the left to right is based on the indexes of each algorithm under compared. As can be seen, $C(\boldsymbol{X}_i, \boldsymbol{X}_j)$ for $i = j$ always takes the value of zero since two identical populations cannot dominate each other. There is also no clear evidence that any of the population had completely dominated any other population in all the 30 runs since there appears no cases where $C(\boldsymbol{X}_i, \boldsymbol{X}_j) = 0$ and $C(\boldsymbol{X}_j, \boldsymbol{X}_i) = 1$, for $i \neq j$. However, MOEA appears to dominate other algorithms in most of the cases, besides being dominated the least by other algorithms as shown in the rectangle of $C(\boldsymbol{X}_8, \boldsymbol{X}_{1-8})$.

Fig. 30 summarizes the performance of each algorithm with respect to the measure of *SSC*. The higher the value of *SSC*, the larger the dominated volume covered by the Pareto-front and hence the better is the MO optimization performance. As can be seen, the MOEA (indexed 8) has the highest value with best performance in spreading strings along the Pareto-front as compared to other methods. Besides MOEA, it can be observed that

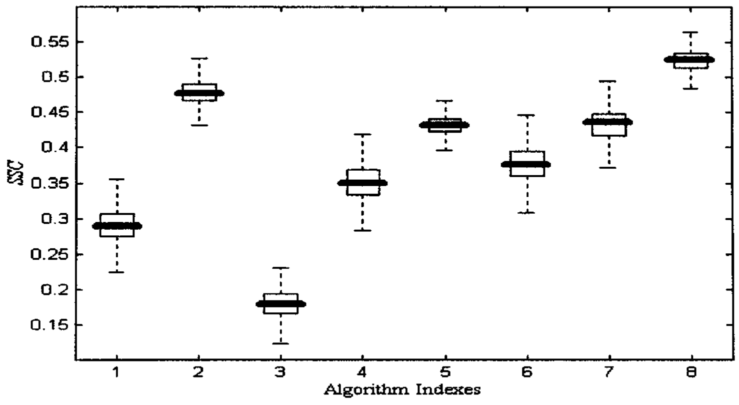


Fig. 30. Box plots on the size of space covered (*SSC*).

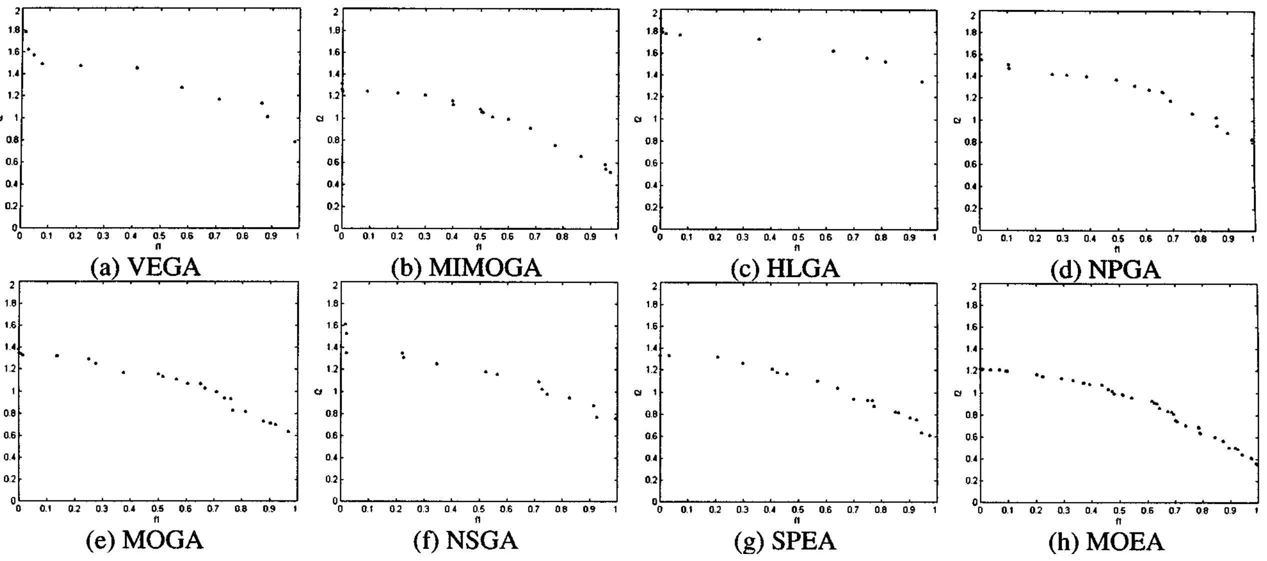


Fig. 31. Best selected distribution of nondominated strings with respect to *SSC*.

MIMOGA (indexed 2), SPEA (indexed 7) and MOGA (indexed 5) also perform satisfactorily for this performance measure.

Fig. 31 unveils the distribution of final evolved nondominated strings in the objective domain. These distributions are best selected among the 30 independent runs with respect to the measure of *SSC*. In general, the purpose of producing these figures is to visually inspect the performances of various algorithms in terms of their final population distribution, i.e., to evaluate the performances qualitatively. By inspection, it is noticeable that MOEA produces more nondominated strings along the Pareto-front, and the final tradeoffs found by the MOEA is better and more uniformly distributed as compared to other methods in literature. Besides having many interactive GUIs and useful features, the MOEA toolbox presented in this paper also shows excellent performance for MO optimization as illustrated in Fig. 31.

V. Conclusions and Future Work

A general computer-aided MO optimization architecture that promotes active man-machine interaction and supports the automatic search of optimum solutions for MO optimization has been discussed in the paper. A powerful GUI-based MOEA toolbox has been presented which applies the concept of Pareto's optimality for uniform distribution of nondominated solutions along the tradeoff. The toolbox is also fully equipped with many useful features for better decision-making in MO optimization, and is capable of representing simulation results in various formats, such as text files or interactive graphical displays for results viewing and analysis. It is freely available for download at http://vlab.ee.nus.edu.sg/~kctan/moea.htm, which is ready for immediate use with minimal knowledge needed in evolutionary computing. Practical usefulness of the toolbox has been demonstrated through the MO design optimization of an MIMO distillation control application. Extensive simulations for the MOEA toolbox and other well-known evolutionary methods upon a benchmark problem have also been performed. The performances are compared both statistically and qualitatively, which shows that the MOEA toolbox is effective for MO optimization with more nondominated solutions evenly distributed along the final Pareto-front.

Although the MOEA toolbox presented in this paper is powerful and ready for immediate use, other features are currently being incorporated into the toolbox. Besides MOEA, other nonevolutionary-based algorithms such as simulated annealing [57] and Tabu search [58] are being included in the toolbox to provide users with a wider choice of optimization tools. To achieve faster program execution and to make the toolbox independent to Matlab, conversion of the toolbox into standalone executable software is also underway. For this, a link to Matlab will be provided in the toolbox so that users can easily access any built-in auxiliary functions in Matlab if necessary. Other toolbox development includes the extension for real-time learning and optimization, such that online communication with real-world applications is possible.

REFERENCES

[1] E. M. Beale, *Introduction to Optimization*. New York: Wiley , 1988. Wiley-Interscience Series in Discrete Mathematics and Optimization.

[2] A. Grace, *Optimization Toolbox User's Guide*. Natick, MA: The MathWorks, Inc., 1992.

[3] J. Horn and N. Nafpliotis, "Multiobjective Optimization Using the Niche Pareto Genetic Algorithm," Univ. Illinois, Urbana, IlliGAL Rep. 93 005, 1993.

[4] G. W. Greenwood, X. S. Hu, and J. G. D'Ambrosio, "Fitness functions for multiple objective optimization problems: Combining preferences with Pareto rankings," in *Foundations of Genetic Algorithms*, R. K. Belew and M. D. Vose, Eds. San Mateo, California: Morgan Kaufmann, 1997, pp. 437–455.

[5] J. Lis and A. E. Eiben, "A multi-sexual genetic algorithm for multiobjective optimization," in *IEEE Int. Conf. Evolutionary Computation*, 1997, pp. 59–64.

[6] M. P. Fourman, "Compaction of symbolic layout using genetic algorithms," in *Proc. of the First Int. Conf. Genetic Algorithms*, 1985, pp. 141–153.

[7] C. M. Fonseca and P. J. Fleming, "Genetic algorithm for multiobjective optimization, formulation, discussion and generalization," in *Proc. of the Fifth Int. Conf. Genetic Algorithms*, S. Forrest, Ed.. San Mateo, CA, 1993, pp. 416–423.

[8] P. B. Wilson and M. D. Macleod, "Low implementation cost IIR digital filter design using genetic algorithms," in *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, Chelmsford, U.K., 1993, pp. 4/1–4/8.

[9] W. Jakob, M. Gorges-Schleuter, and C. Blume, "Application of genetic algorithms to task planning and learning," in *Parallel Problem Solving from Nature, 2nd Workshop*, R. Männer and B. Nanderick, Eds. Amsterdam, The Netherlands, 1992, pp. 291–300. Lecture Notes in Computer Science.

[10] H. Adeli and N. T. Cheng, "Augmented Lagrangian genetic algorithm for structural optimization," *J. Aerosp. Eng.*, vol. 7, pp. 104–118, 1994.

[11] B. J. Ritzel, J. W. Eheart, and S. Ranjithan, "Using genetic algorithms to solve a multi objective groundwater pollution containment problem," *Water Resources Res.*, vol. 30, pp. 1589–1603, 1994.

[12] O. C. Haas, K. J. Burnham, and J. A. Mills, "On improving physical selectivity in the treatment of cancer: A systems modeling and optimization approach," *Contr. Eng. Practice*, vol. 5, no. 12, pp. 1739–1745, 1997.

[13] M. Reformat, E. Kuffel, D. Woodford, and W. Pedrycz, "Application of genetic algorithms for control design in power systems," in *Proc. Inst. Elect. Eng., Generation, Transm. Distrib.*, vol. 145, 1998, pp. 345–354.

[14] T. Morimoto, T. Torii, and Y. Hashimoto, "Optimal control of physiological processes of plants in a green plant factory," *Contr. Eng. Practice*, vol. 3, no. 4, pp. 505–511, 1995.

[15] D. S. Lin and J. J. Leou, "A genetic algorithm approach to chinese handwriting normalization," *IEEE Trans. Syst. Man, Cybern. B*, vol. 27, pp. 999–1007, Dec. 1997.

[16] C. A. Coello-Coello, "An empirical study of evolutionary techniques for multiobjective optimization in engineering design," Ph.D. dissertation, Dept. Comput. Sci., Tulane Univ., New Orleans, LA, 1996.

[17] ——, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Int. J. Knowl. Inform. Syst.*, vol. 1, no. 3, pp. 269–308, 1999.

[18] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.*, vol. 3, no. 1, pp. 1–16, 1995.

[19] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the art," *Evol. Comput.*, vol. 8, no. 2, pp. 125–147, 2000.

[20] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms—A comparative case study," in *Parallel Problem Solving from Nature V*, A. E. Eiben, Ed. Amsterdam, , The Netherlands: Springer-Verlag, 1998, pp. 292–301.

[21] *Using MATLAB*. Natick, MA: The MathWorks, Inc., 1998, ver. 5.

[22] Genetic and Evolutionary Algorithm Toolbox (GEATbx) for Use with Matlab, H. Pohlheim. (1998). [Online]. Available: http://www/geatbx.com

[23] C. Houck, J. Joines, and M. Kay. (1995) A genetic algorithm for function optimization: A Matlab implementation. North Carolina State Univ., Raleigh. [Online]. Available: http://www.ie.ncsu.edu/mirage/GAToolBox/gaot/

[24] Flex Tool (GA) (1999). [Online]. Available: http://www.flextool.com/

[25] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms with goal and priority information for multi-objective optimization," *Congr. Evol. Comput.*, vol. 1, pp. 106–113, 1999.

[26] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," *Proc. First IEEE Conf. Evolutionary Computation*, vol. 1, pp. 82–87, 1994.

[27] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.

[28] *MATCOM*. Natick, MA: The MathWorks, Inc., 1999.

[29] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multi-modal function optimization," in *Proc. Second Int. Conf. on Genetic Algorithms*. Hillsdale, NJ, 1987, pp. 41–49.

[30] *Using Simulink Version 3*. Natick, MA: The MathWorks, Inc., 1999, ver. 5. The Math Works, Inc.

[31] K. C. Tan, T. H. Lee, D. Khoo, and E. F. Khor, "Gene domain constraint handling technique via genetic structure design," in *Congr. Evol. Comput.*, Seoul, Korea, 2001, pp. 693–703.

[32] K. C. Tan, T. H. Lee, E. F. Khor, C. M. Heng, and D. Khoo, "Nonlinear constraint handling technique via angular transformation," in *Genetic and Evolutionary Computation Conf. (GECCO)*, San Francisco, CA, 2001, pp. 665–662.

[33] K. C. Tan, T. H. Lee, and E. F. Khor, "Incrementing multi-objective evolutionary algorithms: Performance studies and comparisons," in *First Int. Conf. Evolutionary Multi-Criteria Optimization (EMO'01)*, Zürich, Switzerland, 2001, pp. 111–125. Springer-Verlag Lecture Notes on Computer Science 1993.

[34] K. Deb and D. E. Goldberg, "An investigation on niche and species formation in genetic function optimization," in *Proc. Third Int. Conf. Genetic Algorithms*. San Mateo, CA, 1989, pp. 42–50.

[35] C. M. Fonseca, "Multiobjective genetic algorithms with application to control engineering problems," Ph.D. thesis, Dept. Automat. Contr. Syst. Eng., Univ. Sheffield, U.K., 1995.

[36] *Real-Time Workshop: For Use with Simulink*. Natick, Ma: The Math Works, Inc., 1995.

[37] P. Schroder, B. Green, N. Grum, and P. J. Fleming, "On-line genetic auto-tuning of mixed H_2/H_∞ optimal magnetic bearing controllers," *Int. Conf. Control*, vol. 2, pp. 1123–1128, 1998.

[38] H. Hanselmann, "Automotive control: From concept to experiment to product," presented at the IEEE Int. Conf. Control Application and System Design, Dearborn, MI, 1996.

[39] S. Rebeschieß, "MIRCOS—Microcontroller-based real time control system toolbox for use with Matlab/Simulink," in *IEEE Int. Conf. Control Application and System Design*, Hawaii, 1999, pp. 267–272.

[40] G. Zames, "On the input-output stability of time-varying nonlinear feedback systems, parts I and II," *IEEE Trans. Automat. Contr.*, vol. AC-11, no. 2 & 3, pp. 228–238 and 465–476, 1966.

[41] S. Skogestad, M. Morari, and J. Doyle, "Robust control of ill-conditioned plants: High-purity distillation," *IEEE Trans. Automat. Contr.*, vol. 33, pp. 672–681, Dec. 1989.

[42] I. Postlethwaite, J. L. Lin, and D. W. Gu, "Robust control of a high purity distillation column using mu-k iteration," in *Proc. 30th Conf. Decision and Control*, 1991, pp. 1586–1590.

[43] T. Zhou and H. Kimura, "Controller design of ill-conditioned plant using robust stability degree assignment," in *Proc. 30th Conf. Decision and Control*, 1991, pp. 1591–1595.

[44] F. Diggelen and K. A. Glover, "Hadamard weighted loop shaping design procedure," in *Proc. 31st Conf. Decision and Control*, 1992, pp. 2193–2198.

[45] D. J. Limebeer, E. M. Kasenally, and J. D. Perkins, "On the design of robust two degree of freedom controllers," *Automatica*, vol. 29, no. 1, pp. 157–168, 1993.

[46] P. Lundström, S. Skogestad, and J. C. Doyle, "Two-degree-of-freedom controller design for an ill-conditioned distillation process using μ-synthesis," *IEEE Trans. Control Syst. Technol.*, vol. 7, no. 1, pp. 12–21, 1999.

[47] K. X. Guan and K. J. MacCallum, "Adopting a minimum commitment principle for computer aided geometric design systems," in *Artificial Intelligence in Design'96*, J. S. Gero and F. Sudweeks, Eds. Norwell, MA, 1996, pp. 623–639.

[48] J. Devore and R. Peck, *Statistics: The Exploration and Analysis of Data*. London, U.K.: Duxbury, 1997.

[49] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problem," *Evol. Comput.*, vol. 7, no. 3, pp. 205–230, 1999.

[50] C. M. Fonseca and P. J. Fleming, "On the performance assessment and comparison of stochastic multiobjective optimizers," in *Parallel Problem Solving from Nature*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 1996, pp. 584–593.

[51] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suites," in *Proc. Symp. Applied Computing*, San Antonio, TX, 1999, pp. 351–357.

[52] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, 1999.

[53] J. D. Schaffer, "Multiple-objective optimization using genetic algorithm," in *Proc. First Int. Conf. Genetic Algorithms*, 1985, pp. 93–100.

[54] T. Murata and H. Ishibuchi, "MOGA: Multi-objective genetic algorithms," in *IEEE Proc. Congr. Evolutionary Computation*, vol. 1, 1995, pp. 289–294.

[55] P. Hajela and C. Y. Lin, "Genetic search strategies in multicriterion optimal design," *J. Struct. Optim.*, vol. 4, pp. 99–107, 1992.

[56] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Turkey, *Graphical Methods for Data Analysis*. Pacifica, CA: Wadsworth & Brooks/Cole, 1983.

[57] P. Czyzak and A. Jaszkiewicz, "Pareto simulated annealing," in *Proc. XIIth Int. Conf. Multiple Criteria Decision Making*, G. Fandel and T. Gal, Eds, 1997, pp. 297–307.

[58] M. P. Hansen, "Tabu search in multiobjective optimization: MOTS," presented at the Proc. MCDM'97, Cape Town, South Africa, 1997.

K. C. Tan (S'95–A'97–M'99) received the B.Eng. degree (with First Class Honors) in electronics and electrical engineering 1994 and the Ph.D. degree in 1997, both from the University of Glasgow, U.K.

He was with the Centre for Systems & Control and the Evolutionary Computing Group, Glasgow, before joining the Department of Electrical and Computer Engineering at the National University of Singapore as an Assistant Professor in 1997. His research interests include computational intelligence, evolutionary multiobjective optimization, intelligent control and engineering designs optimization. He has more than 60 technical publications in these areas, and has served as program committee or organizing member for many international conferences. He is currently an Associate Editor for the *Institution of Engineers* journal, Singapore.

Tong H. Lee (M'88) received the B.A. degree (with First Class Honors) in the Engineering Tripos from Cambridge University, U.K., in 1980 and the Ph.D. degree from Yale University, New Haven, CT, in 1987.

He is a Professor in the Department of Electrical and Computer Engineering, National University of Singapore. He is also currently Head of the Control Engineering Section in this Department, and the Vice-Dean (Research) in the Faculty of Engineering. His research interests are in the areas of adaptive systems, knowledge-based control, intelligent mechatronics and computational intelligence. He has published extensively in these areas, and is currently an Associate Editor for *Automatica*; *Control Engineering Practice* (an IFAC journal); the *International Journal of Systems Science* (Taylor and Francis, London, U.K.); and *Mechatronics* (Oxford, U.K., Pergamon Press).

Dr. Lee was a recipient of the Cambridge University Charles Baker Prize in Engineering. He is an Associate Editor for the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS.

D. Khoo received the B.Eng. degree (with First Class Honors) in electronic and electrical engineering from University of Strathclyde, Glasgow, U.K., in 1998.

He is currently a Research Engineer in the Department of Electrical and Computer Engineering at the National University of Singapore. His research interests include evolutionary algorithm implementation and genetic coding of optimization problems.

E. F. Khor was born in Malaysia in 1974. He received the B.Eng. degree (with First Class Honors) in electrical engineering from the University of Technology (UTM), Malaysia, in 1998. He has been pursuing the Ph.D. degree in the Centre for Intelligent Control, National University of Singapore, since 1998. His research interests include evolutionary multiobjective optimization, artificial intelligence, stochastic optimization and design automation in control engineering.