An Extensible Genetic Algorithm Framework for Problem Solving in a Common Environment

Angela S. Chuang, *Student Member, IEEE,* and Felix Wu, *Fellow, IEEE*

***Abstract*—We describe an object-oriented framework for solving mathematical programs using genetic algorithms (GA's). The advantages of this framework are its extensibility, modular design, and accessibility to existing programming code. The framework also incorporates a graphical user's interface that may be used to build new GA's as well as run GA simulations. Two power system problems are solved by implementing genetic algorithms using the said framework. The first is a continuous optimization problem and the second an integer programming problem. We illustrate the flexibility of the framework as well as its other features on our test problems.**

***Index Terms*—Genetic algorithms, graphical user interfaces, object-oriented programming, optimization methods.**

I. Introduction

IN RECENT years there has been growing interest in applying genetic algorithms to solve difficult problems in power systems. Hundreds of papers have been published in technical journals and proceedings reporting applications of GA's to power system problems, such as: economic dispatch [3], [19], [20], unit commitment [12], [17], generation planning [9], distribution network expansion planning [14], VAR planning and dispatch [11], [24], load flow [25], [26], and optimal capacitor selection [4], [21]. Other applications include distribution network re-configuration and service restoration [6], [8], fault diagnosis [22], distribution system harmonic worst case design [16], meter allocation [15], contingency screening [18], and transmission planning [27]. Because power system problems are complex, often combinatorial, and frequently NP-hard in nature, GA's have been employed more and more to find improved solutions for them.

GA's developed to solve particular problems are generally implemented as stand-alone programs. However, in this paper we describe a framework that can be used as a unifying platform for developing GA solvers. One immediate advantage of development under a common software environment is the ability to reuse existing code. Besides facilitating an increase in code reuse potential, the framework enables users to build algorithms easily using a GUI. Moreover, the framework is extensible, flexible, easy to maintain, and may be interfaced with existing software applications, including commercial software like Matlab. We call our Genetic Algorithm Framework: GAFrame. It is a flexible and extensible programming environment for implementing GA's and facilitating their application to power systems.

In the ensuing sections, we illustrate many advantages of the proposed framework from a user's perspective. Section II provides a brief overview of genetic algorithms. Section III introduces the software environment Ptolemy and describes the relationship of GAFrame with Ptolemy. Section IV details how to construct GA's using the framework. In Section V, we describe how two power systems applications were solved using GAFrame, while highlighting some of the framework's features.

Manuscript received January 5, 1998; revised August 29, 1998. This work was supported by a National Science Foundation Graduate Fellowship, a grant from Hong Kong Government Research Grant Council, and a Grant from HKU CRCG.

A. S. Chuang is with the Department of Electrical Engineering & Computer Science, University of California, Berkeley, Berkeley, CA USA

F. Wu is with the Center for Electrical Energy Systems, University of Hong Kong.

Publisher Item Identifier S 0885-8950(00)01883-6.

II. Genetic Algorithms

Genetic algorithms are global, randomized search techniques based on the mechanics of natural genetics [10]. In a GA, solutions represented by data structures called individuals are evolved and new populations of individuals created. Every individual is assigned a *fitness* measure which characterizes how it compares to other individuals in the same population. During the course of an algorithm run, populations with improved solutions are evolved until a stopping criterion is met.

GA's differ from traditional optimization methods in several ways. GA's use probabilistic transition rules and not deterministic ones. They work with a coding of the domain set, and not on the actual domain members themselves. Search is conducted over a population of points, not from a single point like in traditional gradient decent methods. Hence, a genetic algorithm outputs multiple solutions with alternatives to choose from. Furthermore, little assumption is required concerning the nature of the problem's objective function; the function must simply be defined over the problem domain. Consequently, GA's require no auxiliary knowledge about the objective function, such as continuity, differentiability, and unimodality, unlike traditional optimization methods which do.

Since there are no fundamental restrictions on the search space nor on the nature of the objective function, the class of problems for which GA's may be applied is vast. However, GA's are heavily problem dependent because representation schemes that encode individuals must be formed for any solution space considered [13]. Furthermore, genetic operators must be chosen so that solutions are evolved in such a way as to meet problem constrains yet not depreciate algorithm performance considerably. Ironically, while the disadvantage in developing GA's lies in the necessity of encoding solutions, the power of GA's also lies in their use of encodings. GA's exploit patterns in encoded structures to obtain new structures which represent improved solutions. Thus, once a suitable

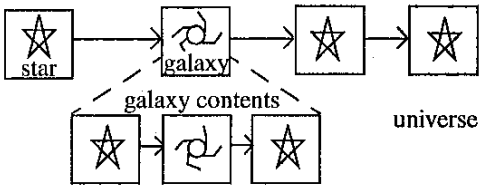

Fig. 1. A complete Ptolemy application (a universe) consists of a connection of blocks, which may be stars or galaxies.

representation can be found for a problem, the power of a GA can be unleashed.

III. Foundations and Advantages

The Genetic Algorithm Framework is built atop an object-oriented heterogeneous simulation environment called Ptolemy [5], a freely-distributed software package maintained by members of the Electrical Engineering Department at UC Berkeley.[1] The motivation for designing and implementing a framework for Genetic Algorithms in the Ptolemy environment evolves around several software issues. Since software applications written today are becoming increasingly modular, with single applications serving one part in a wide collection of applications, the ability to link different programs together is extremely valuable. Besides providing the necessary interfaces to link various applications within Ptolemy, Ptolemy also provides interfaces for linking with various external software packages such as Matlab (a commercial numerical computation software).

In Ptolemy, code is grouped in modular units called stars, and a collection of stars comprise a unit called a galaxy. Galaxies and stars can be interconnected to form a stand-alone executable program called a *universe*. Fig. 1 shows an example of a universe. The sequence of execution of stars in any universe is determined by a built-in *scheduler* that follows an inherent model of computation characteristic of the *domain* of choice [2]. Its scheduling capabilities combined with its modular structure make Ptolemy a flexible and extensible environment in which to develop applications.

In Ptolemy, stars, galaxies, and universes are represented by icons that are stored in and can be accessed graphically from libraries called *palettes*. Using the built-in GUI, one may copy, move, re-orient, and view the contents of icons. Each time an icon representing a star or galaxy is copied to a universe, a new instance of that object is created. In this way, pre-written stars can be reused in new simulations by simply copying their icons. Moreover, users need only be familiar with a single GUI to operate on a series of applications.

Since GAFrame is implemented within the Ptolemy environment, all the advantages mentioned above extend to algorithm-building using the framework, which will be described next.

[1]For more information on Ptolemy or for a copy of the latest distribution, visit the web site http://ptolemy.eecs.berkeley.edu.

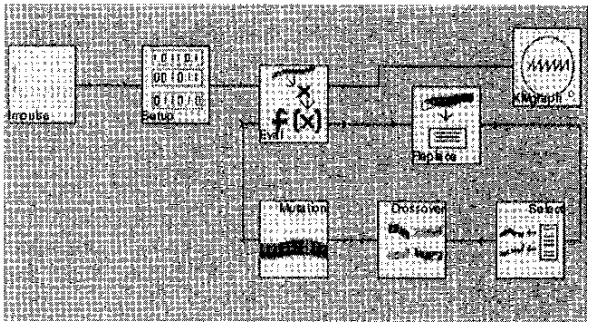

Fig. 2. The basic structure of a genetic algorithm universe in GAFrame.

TABLE I
GA Star Descriptions

Star	Description	Palette Location
Pop	sets population type	rep.pal
Setup	allocates memory and initializes population.	setup.pal
Eval	evaluates fitness of individuals in population.	eval.pal
Replace	elitist replacement	replace.pal
Select	proportional selection	select.pal
Cross	n-point crossover	cross.pal
Mutate	uniform mutation	mutate.pal

IV. GAFrame

A. Description

The Genetic Algorithm Framework (GAFrame) is a modularly designed universe implemented in Ptolemy for solving optimization problems by genetic algorithms. The framework includes a collection of stars stored in palettes that serve as basic building blocks for new GA applications.

Each GAFrame universe has the basic structure illustrated in Fig. 2. The functions of stars shown in the figure are summarized in Table I. The Pop star (not shown in the figure) defines the base population type from which other GA stars are derived. Genetic operations are implemented in the stars Setup, eval, Replace, Mutation, Crossover, and Select. These stars are extensions to Ptolemy and provided within GAFrame. Other stars such as Impulse and XMGraph, which are used to start and track the progress of an algorithm run, are part of the original Ptolemy release.

B. Requirements

Other than the basic requirements for implementing any GA, building a GA with the proposed framework requires no additional information about the given application nor associated optimization problem. To implement a GA for a particular optimization problem, the developer chooses appropriate genetic operators (e.g., selection, crossover, mutation, and replacement

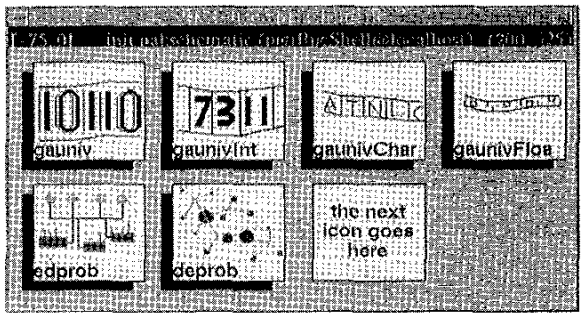

Fig. 3. A palette containing icon representations of GAFrame universes, plus two GA applications.

operators); an evaluation procedure; and a suitable encoding for representing population members.

Often some programming is required to tailor the universe to the specific problem at hand. As a general rule of thumb, pre-written stars that implement generic GA operations can usually be copied and incorporated into a new algorithm with little or no modification to their internal code. However, when manipulation of population members is necessary to ensure feasibility of solutions for a specific application, some programming is required of the algorithm builder. In that case, the algorithm builder either modifies existing stars or creates new stars and connects them inside a universe. Stars included with GAFrame can be regarded as templates that have encoded within them overhead code related to the framework design and Ptolemy itself. By using these templates, the amount of knowledge about the design and Ptolemy preprocessor language is reduced for the algorithm builder.

Because C is a strongly type-specific programming language, separate GA stars and universes were created for different population data types. Namely, populations can consist of individuals that are represented by strings (i.e., concatenations) of characters, or strings of binary, integer, or floating-point numbers. In GAFrame, star and universe names are associated with their corresponding population type, making the inherent data type explicit. Fig. 3 illustrates a palette of GA universes. The universe template associated with a population of binary strings is named "gauniv," integer strings is named "gaunivInt," floating point strings is named "gaunivFloa," and character strings is named "gaunivChar."

C. Creating a GA Universe

The procedure for creating GA's with GAFrame is straightforward. A GA developer copies the universe template appropriate for his/her choice of population type, and then renames the universe to reflect his/her application. For example, the "edprob" universe in Fig. 3 was created from a copy of "gauniv" and renamed to identify it as an economic dispatch application. Next, the developer adds, deletes, or copies stars from palettes or other universes, as needed. The developer also renames any star that will be modified, leaving the original templates unchanged. To alter the function of a star, the developer writes C code within the star. The programmer's guide in [7] gives details on how to

Step 1: Copy the appropriate universe template
Step 2: Rename the universe to reflect the particular application
Step 3: Look inside the universe (*I*)
Step 4: Delete & add stars appropriate to the application
Step 5: Rename modified stars, leaving originals unaltered
Step 6: Edit parameters of stars in the universe (*e*)
Step 7: Run the universe (*R*)
Step 8: Input the desired number of iterations (i.e., generations) into the run control window.

Fig. 4. A summary of the procedure for building a genetic algorithm using GAFrame (GUI *commands are italicized*).

TABLE II
DATA FOR THE PROBLEM INSTANCE

Unit i	P_i^{min} [MW]	P_i^{max} [MW]	cost curve $F_i(P_i)$ [$/h]
1	150	600	$0.001562P_1^2+7.92P_1+561$
2	100	400	$0.00194P_2^2+7.85P_2+310$
3	50	200	$0.00482P_3^2+7.97P_3+78$

code stars. In Fig. 4, we summarize the step-by-step procedure for creating GA's with GAFrame.

V. APPLICATIONS

A. Economic Dispatch

Economic dispatch is a well explored problem in power systems with established solution techniques and algorithms. In the last 3 years many papers have been published reporting on genetic algorithm solutions to this particular problem [3], [19], [20].

We have also solved an instance of the Economic Dispatch Problem (EDProb) by a GA constructed using our framework. From this example, we demonstrate the use of GAFrame in solving a continuous-time optimization problem.

A text book example from [23] was taken as our test problem. Namely, the test case is:

$$\min F_1 + F_2 + \cdots + F_N s.t. P_1 + P_2 + \cdots + P_N = P_r + P_l P_i^{\min} \le P_i \le P_i^{\max} \quad \text{for } i = 1, 2, \cdots, N$$

where P_i is the power output of the ith generator connected to a single bus-bar, P_r is the electrical load, P_l is the power loss over the system, and F_i is the cost curve of generator i as a function of P_i as shown in Table II. Note that the P_i's are the decision variables for this optimization problem.

Employing the Simple Genetic Algorithm described in [10], we solve the test problem using binary encoded strings and proportional selection, one-point crossover, uniform mutation, and elitist replacement operators. To ensure feasibility of solutions contained in the running population, we construct penalty functions similar to those used in [19].

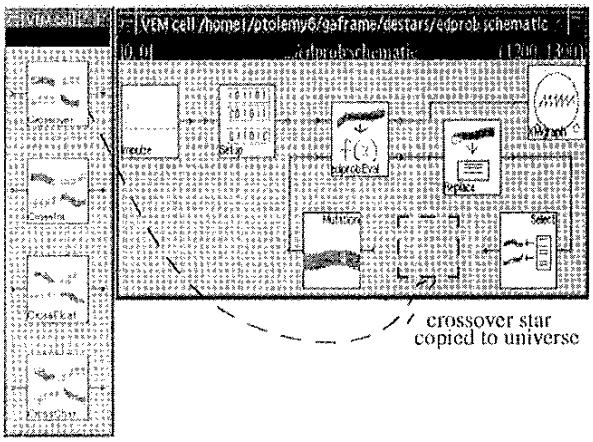

Fig. 5. An illustration of a step in the algorithm-building procedure: a crossover star is copied from a palette into a universe, using the Ptolemy GUI.

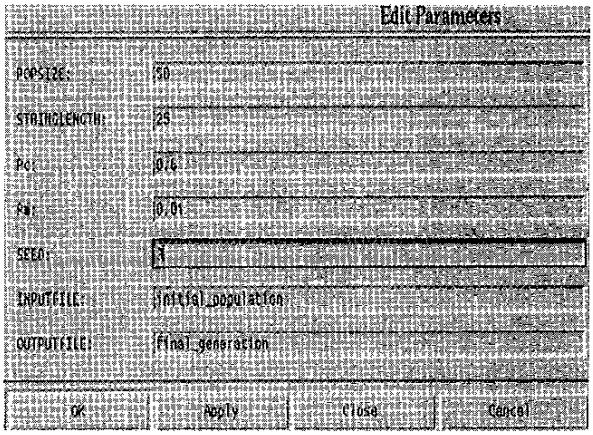

Fig. 6. An editable parameters window accessed from within a GAFrame universe.

Fig. 5 contains the universe that was built to solve the Economic Dispatch problem. The universe was created by copying the template "gauniv" shown in Fig. 2, the template that encodes binary string individuals. Since the genetic operators provided in this template are exactly those desired and because feasibility was enforced by penalties added to the objective function, only the Eval star needed modification to tailor the algorithm to this particular application.

Fig. 6 shows the GA parameters used for this application. Each trial run of the EDProb universe was initialized by a different random seed, and the GA was allowed to iterate for over 100 generations. This stopping criterion was specified in the run control panel window of the EDProb universe, illustrated in Fig. 7. Resulting outputs for 9 simulation runs are recorded in Table III. The GA obtained solutions with objective values within 0.2% of the known optimal objective value (computable using Lagrangian techniques).

B. Distribution Network Expansion Planning

In recent years, genetic algorithms have been applied toward solving Distribution Network Expansion Planning problems

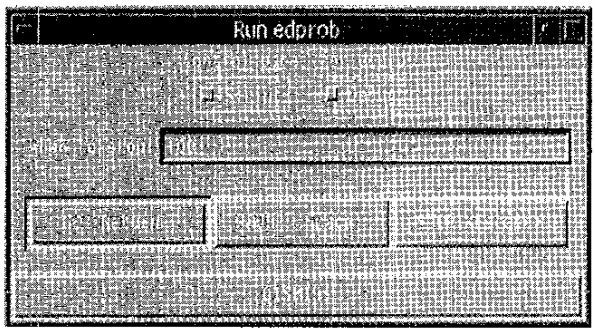

Fig. 7. Example of a run control panel window.

TABLE III
RESULTS OF GENETIC ALGORITHM RUNS

Run	P_1	P_2	P_3	objective value	total power	% error in obj. val.
1	352.4	373.5	124.5	8203.5	850.4	0.1%
2	381.6	363.8	105.0	8200.6	850.4	0.08%
3	438.0	292.5	119.9	8204.5	850.4	0.1
4	360.5	370.0	117.4	8179.7	847.9	0.2%
5	405.5	303.2	141.3	8197.9	850.0	0.04%
6	405.8	303.1	141.2	8198.3	850.1	0.05%
7	405.8	303.1	141.2	8198.3	850.1	0.05%
8	367.0	368.6	114.6	8200.0	850.2	0.07%
9	422.6	297.5	129.9	8198.6	850.0	0.05%

[14]. These integer optimization problems are difficult to solve using traditional optimization methods.

We apply our framework toward solving a variant of the planning problem which minimizes cable cost and resistive loss in distribution networks constrained to radial network configurations. The problem studied can be stated as: given a fixed set of substations and demand locations, what is the cheapest way to radially interconnect them so that cable cost (a planning factor) and resistive loss (an operations factor) are minimized? This problem may be formulated as:

$$\min C_f(d) + C_1(P) s.t.\ f(P, Q, V, \theta) = 0 \qquad (1)$$

$$V_i^{\min} \le V_i \le V_i^{\max} \qquad \text{for } i = 1, 2, \cdots, N \qquad (2)$$

$$I_i \le I_i^{\max} \qquad \text{for } i = 1, 2, \cdots, N \qquad (3)$$

where $C_f(d)$ is the total feeder cost for a given distribution network, and $C_1(P)$ is the total resistive loss for the network multiplied by a cost factor. Equation (1) is the requirement that load flow equations are satisfied by candidate solutions. Inequality (2) restricts voltage magnitude at each network node. And inequality (3) is a set of thermal limit constraints for feeder lines.

The GA universe created to solve this problem utilizes integer string type populations, for which each individual is a particular

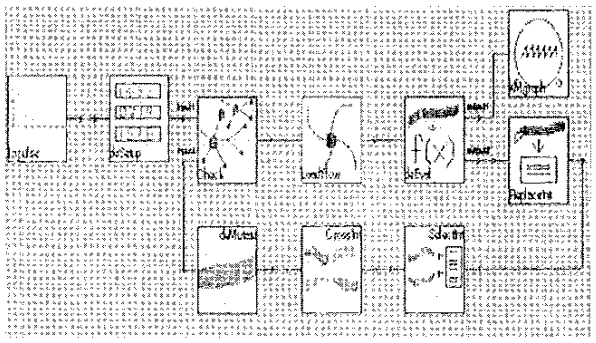

Fig. 8. GA universe that solves the distribution expansion planning problem.

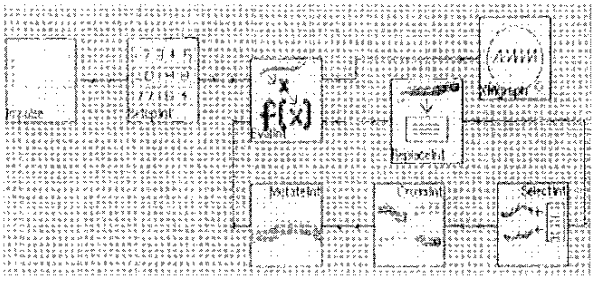

Fig. 9. GA universe template ("gaunivInt") that encodes integer string population members.

distribution network configuration. Fig. 8 illustrates the newly created universe DEProb. To build this universe we started from the template "gaunivInt" shown in Fig. 9. After copying and modifying some existing stars, we added 2 new ones (Check and Loadflow) that check feasibility of solutions.

In all, 3 stars representing genetic operators were copied from GAFrame and reused without further modification; 2 stars from the original Ptolemy distribution were reused; 3 GAFrame stars were copied and slightly modified; and 2 totally new stars were added. Although the latter 2 stars were newly created for the particular application, they utilized an existing Matlab script file that performs loadflow computations. We have incorporated use of a Matlab load-flow program in our GA to illustrate the potential of accessing external programs from within GAFrame universes.

In general, implementing feasibility checking functions and constraint enforcement techniques are must's for any GA application. A major advantage of the said framework is the ability to try various genetic operators in an algorithm by simply inserting their graphical icons into the universe being constructed, and rerunning the simulation. If the feasibility enforcement technique utilized is separable from the genetic operator functions, as in our GA in Fig. 8 (or by use of penalty functions as in the Economic Dispatch Problem of Section V-A), then various genetic operator stars can be freely substituted into the universe.

We employed our GA toward finding a cost-minimizing distribution network that connects each demand center to a substation, for the nodes and buses shown in Fig. 12. Fig. 11 gives additional data corresponding to the test problem, where costs are dependent on the power flow across each line as well as the

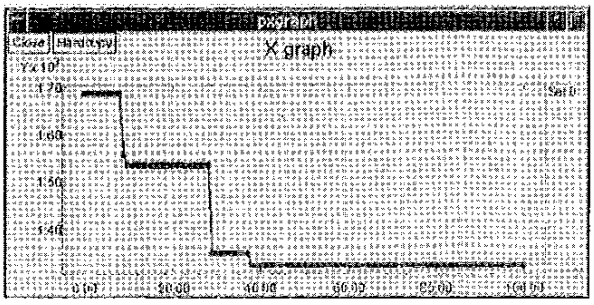

Fig. 10. Results of a GA run. The objective function value of the best member of the current population is plotted.

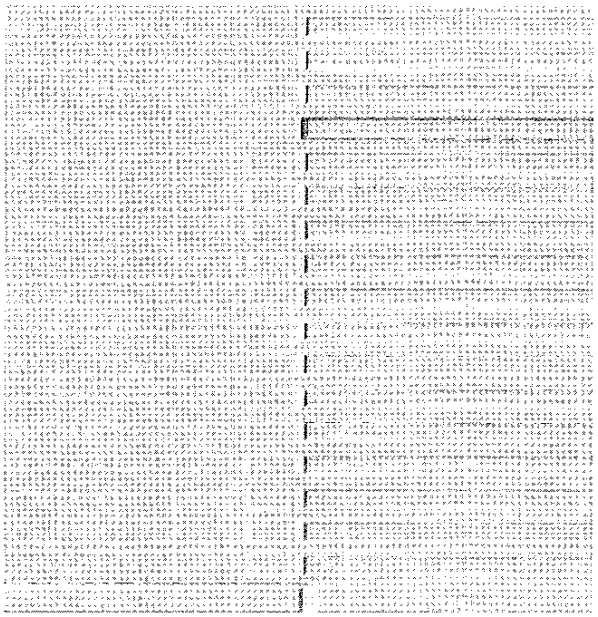

Fig. 11. Editable parameters window for the distribution expansion planning setup star.

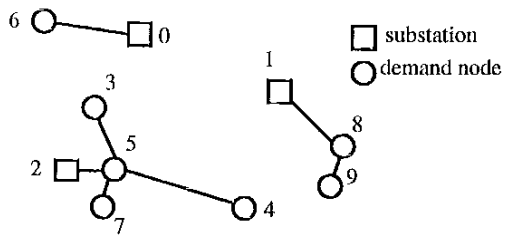

Fig. 12. The best solution found.

distance of separation between two nodes. Output for a single run of the GA is plotted in Fig. 10.

Due to the complex nature of the problem's feasible domain, feasible solutions were difficult to randomly generate. This is reflected in the sharp downward jumps displayed in Fig. 10, where improved feasible solutions are only represented at where the jumps occur. Despite the difficulty in random generation, we formed a good initial mix of feasible individuals by fine

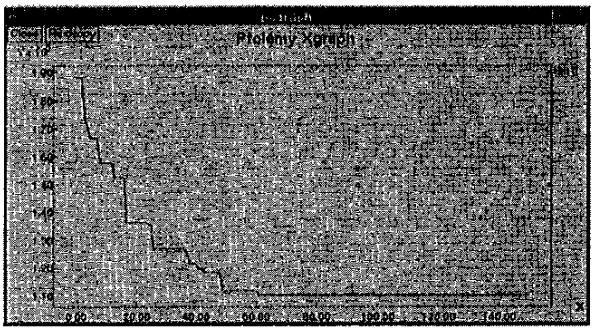

Fig. 13. Results of a fine-tuned GA.

tuning GA parameters (like crossover and mutation probabilities) and by using the final population of a previous run as the initial population of the next. Such fine-tuning operations were easily performed through altering the editable parameters of stars. Moreover, making changes to parameter settings could be done interactively during the course of a simulation run; not just between separate executions of an algorithm. Such an ability to alter parameters as one monitors the progress of an algorithm is a powerful and useful feature for developing GA's. In addition, different genetic operators could also have been substituted into the universe to help search for improved solutions.

Our resulting fine-tuned algorithm produced the graph depicted in Fig. 13. The algorithm converged to the solution shown in Fig. 12 in 60 generations. More about the GA employed to solve this problem and the solutions found will be detailed in a separate paper.

VI. Conclusions

GAFrame is a flexible and extensible framework for constructing genetic algorithms. The framework utilizes object-oriented programming methodology, provides access to a graphical user interface, and can be interfaced with other applications. Furthermore, the framework was designed with considerations for user needs, such as ease of use and code reusability.

To develop a new algorithm, the user simply selects the appropriate universe template according to the desired population data type and builds the GA from there, reusing and adding stars as necessary. Generally, fitness functions are incorporated into the evaluation star to tailor the universe to the specific problem at hand.

Both a continuous-time and integer optimization problem were solved by genetic algorithms constructed using GAFrame. For the Economic Dispatch problem examined, resulting solutions yielded objective values within 0.2% of the known optimal value. For the single-stage distribution expansion planning problem we examined, feasible solutions which yielded large cost improvements were found as the algorithm progressed.

Acknowledgment

The authors wish to thank former students at Hong Kong University: E. K. S. Li, C. F. Yung, and F. Lam, for their contributions in extending the Genetic Algorithm Framework to problems in distribution network expansion planning.

References

[1] *The Almagest, Vol. 1-3: Ptolemy 0.6 Manual*: University of California at Berkeley, Department of Electrical Engineering and Computer Science, 1994.

[2] "An overview of the Ptolemy project," The DSP Design Group, Dept. of EECS, UC Berkeley, Mar 7, 1994.

[3] A. Bakirtzis, V. Petridis, and S. Kazarlis, "Genetic algorithm solution to the economic dispatch problem," in *IEE Proc.: Generation, Transmission and Distribution*, vol. 141, July 1994, pp. 377–382.

[4] G. Boone and H.-D. Chiang, "Optimal capacitor placement in distribution systems by genetic algorithm," *Int'l. Journal of Electrical Power and Energy Systems*, vol. 15, no. 3, pp. 155–162, June 1993.

[5] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Ptolemy: A framework for simulating and prototyping heterogeneous systems," *Int'l. Journal of Computer Simulation*, vol. 4, pp. 155–182, Apr. 1994.

[6] D. Choi and J. Hasegawa, "Improvement of genetic algorithm convergence characteristics for distribution system loss minimization reconfiguration," *Trans. of the Institute of Electrical Engineers of Japan*, pt. B, vol. 115-13, no. 3, pp. 252–259, Mar. 1995.

[7] A. S. Chuang, "An extensible framework for genetic algorithms in the Ptolemy environment," Masters of Science Degree Report, Dept. of Electrical Engineering and Computer Science, UC Berkeley, May 1995.

[8] Y. Fukuyama and Y. Ueki, "An application of genetic algorithms to service restoration in distribution system," *Trans. of the Institute of Electrical Engineers of Japan*, vol. 114-B, no. 4, pp. 361–366, Apr. 1994.

[9] ——, "An application of parallel genetic algorithms to generation expansion planning using parallel processors," *Trans. of the Institute of Electrial Engineers of Japan*, vol. 114-B, no. 12, pp. 1250–6, Dec. 1994.

[10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*: Addison-Wesley, 1989.

[11] K. Iba, "Reactive power optimization by genetic algorithm," *IEEE Trans. on Power Systems*, vol. 9, no. 2, pp. 685–692, May 1994.

[12] T. T. Maifeld and G. B. Sheble, "Genetic-based unit commitment algorithm," *IEEE Trans. on Power Systems*, vol. 11, no. 3, pp. 1359–1370, Aug. 1996.

[13] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd ed: Springer-Verlag, 1994.

[14] V. Miranda, J. V. Ranito, and L. M. Proenca, "Genetic algorithms in optimal multistage distribution network planning," *IEEE Trans. on Power Systems*, vol. 9, no. 4, pp. 1927–1933, Nov. 1994.

[15] H. Mori and S. Iida, "Application of a genetic algorithm to meter allocation in electric power systems," in *Proc. of Int'l. Joint Conference on Neural Networks*. New York: IEEE, 1993, vol. 2, pp. 1594–1597.

[16] G. G. Richards and H. Yang, "Distribution system harmonic worst case design using a genetic algorithm," *IEEE Trans. on Power Delivery*, vol. 8, no. 3, pp. 1484–1491, July 1993.

[17] H. Saitoh, K. Inoue, and J. Toyoda, "Genetic algorithm approach to unit commitment problem," in *Int'l. Conference on Intelligent System Applications to Power Systems, EC2*, vol. 2, France, 1994, pp. 583–589.

[18] H. Saitoh, Y. Takano, and J. Toyoda, "Genetic algorithm based method for contingency screening in power systems," *Trans. of the Institute of Electrical Engineers of Japan*, vol. 115-B, no. 1, pp. 9–17, Jan. 1995.

[19] G. B. Sheble and K. Brittig, "Refined genetic algorithm economic dispatch example," *IEEE Trans. on Power Systems*, vol. 10, no. 1, Feb. 1995.

[20] Y. H. Song, F. Li, R. Morgen, and D. T. Y. Cheng, "Comparison studies of genetic algorithms in power system economic dispatch," *Power System Technology*, vol. 19, no. 3, pp. 28–33, 38, Mar. 1995.

[21] S. Sundhararajan and A. Pahwa, "Optimal selection of capacitors for radial distribution systems using a genetic algorithm," *IEEE Trans. on Power Systems*, vol. 9, no. 3, pp. 1499–1507, Aug. 1994.

[22] F. Wen and Z. Han, "Fault section estimation in power systems using genetic algorithm and simulated annealing," in *Proc. of the CSEE*, vol. 14, May 1994, pp. 29–35, 6.

[23] A. J. Wood and B. F. Wollengberg, *Power Generation, Operation and Control*, 2nd ed: John Wiley and Sons, 1996, pp. 31–34.

[24] Q. H. Wu and J. T. Ma, "Genetic search for optimal reactive power distpach of power systems," in *International Conference on Control*, UK: IEE, 1994, vol. 1, pp. 717–722.

[25] Y. Xiaodong, "Application of genetic algorithms to multiple load flow solution problem in electrical power systems," in *Proc. of the 32nd IEEE Conference on Decision and Control*, vol. 4. New York, 1993, pp. 3734–3739.
[26] Y. Xiaodong and N. Germay, "Investigations on solving the load flow problem by genetic algorithms," *Electric Power Systems Research*, vol. 22, no. 3, pp. 151–163, Dec. 1991.
[27] K. Yoshimoto, K. Yasuda, R. Yokoyama, H. Tanaka, and Y. Akimoto, "An approach for transmission expansion planning using neuro computing hybridized with genetic algorithm," *Trans. of the Institute of Electrical Engineers of Japan*, vol. 114-B, no. 10, pp. 1029–1037, Oct. 1994.

Angela S. Chuang received her B.S. and M.S. degrees in Electrical Engineering and computer science at UC Berkeley in 1992 and 1995, respectively. She is currently a Ph.D. candidate in electrical engineering at the same institution. Her research interests include power systems planning, optimization, and game theory.

Felix Wu received the Ph.D. degree from UC Berkeley, where he is currently a Professor of electrical engineering and computer science, on leave at the University of Hong Kong. Dr. Wu is a Fellow of IEEE.