Comparison of Artificial Life Techniques for Market Simulation

Feng Gao, *Student Member*, *IEEE*, G. Gutierrez-Alcaraz, *Member*, *IEEE*, and
Gerald B. Sheble, *Fellow*, *IEEE*
Iowa State University, Ames, IA, USA
gaofeng@iastate.edu, ggutier@iastate.edu, gsheble@iastate.edu

Abstract

Electricity industries worldwide are undergoing a period of profound upheaval. Conventional vertically integrated mechanism is replaced by a competitive market environment. A pure operating cost optimization is not enough to model the distributed, large-scale complex system. A market simulator will be a valuable training and evaluation tool to assist sellers, buyers & regulators to understand system's dynamic performance and make better decisions avoiding bunch of risks. The objective of this research is to model market players by adaptive multi-agent system, compare the performances of different artificial life technique such as Genetic Algorithm (GA), Evolutionary Programming (EP) and Particle Swarm (PS) in simulating players' behaviors, identify the best method to emulates real rational participants.

1. Introduction

Electricity industries worldwide are undergoing a period of profound upheaval. Conventional vertically integrated mechanism is replaced by a competitive market environment. A pure operating cost optimization is not enough to model the distributed, large-scale complex system. A market simulator will be a valuable training and evaluation tool to assist sellers, buyers & regulators to understand system's dynamic performance and make better decisions avoiding bunch of risks.

Evolutionary computation is a general term for several computational techniques which take their inspiration from natural selection in the biological world and use this mechanism of EVOLUTION as key elements in their design and implementation [1]. There are a variety of evolutionary computational models that have been proposed and studied which are referred to as evolutionary algorithms. ALIFE is a common acronym to tie all of the ideas based on biological emulation, including evolutionary computation. It is the attempt to simulate, or in some case emulate, the governing principles of life. Artificial life techniques, including artificial neural networks, have found success in solving several different complicated centralized non-convex optimization problems and for emulating intelligent market participants' individual decentralized optimization problem.

Post *et al.* [2] proposed a market-based auction pricing mechanism for the interchange of electricity, where a generic transportation model is assumed and solved by linear programming. Richter and Sheble [3] presented a Genetic Algorithm based framework to evolve utility bidding strategies in a double side auction marketplace and developed a market simulator by Pascal language. Agent technology is a means to help in the coordination and negotiation between the market players. Praca *et al.* [4] presented a multi-agent architecture for all market-acting entities such as producers, distributors and regulators. A Java software pack of electricity market simulator is implemented in [5].

The objective of this paper is to determine how to model market players as adaptive multi-agent system, to compare the performances of different artificial life techniques (Genetic Algorithm (GA), Evolutionary Programming (EP) and Particle Swarm (PS)), to simulate corporate (players') behaviors, to identify the best method to emulates real rational decision making by participants.

The rest of the paper is organized as follows: Market structure and rules are introduced in Section II. Section III presents the multi-agent system and evolutionary algorithms which are applied to simulate players' behaviors. A sample result is shown in Section IV. Finally, conclusions are summarized in Section V.

2. Market Structure

0-7695-2507-5/06/$20.00 (C) 2006 IEEE

The electricity market structure normally consists of two types of players: buyers and sellers in addition to the central broker. Transmission companies, TRANSCOs, are considered exogenous to the market. Each seller is modeled as a corporation with variable and fixed costs of production. Each buyer is modeled as a corporation with fixed and variable costs of delivery. Each buyer and seller has a fixed amount to buy or sell. Each buyer and seller is given a network location which is used to determine the transportation capability from each buyer to each seller individually.

The broker matches the bids from each buyer to each seller by ordering the bids inversely by type of player, similar to the Florida Coordination Group approach as described in [6]. Thus the highest buyer is matched with the lowest seller. The broker then verifies that the transaction can occur based on the remaining network capability between these players. If sufficient capacity exists, and then the transaction is committed, the remaining transportation matrix is updated to reflect the actual flow based on the contract, and the next buyer and seller are matched. A buyer or seller may have to engage in more than one transaction if the amount bid is not equal or if the network restricts the flow. The broker always commits transactions to sell all bid amounts unless restricted by the network. The broker continues to match bids until the surplus profit is zero. The surplus profit is the difference between the buyers' and the sellers' bids. The transaction price is always the difference between the two bids, divided by two. This is similar to the power pool split savings approach that many regions have been using for years [7].

3. Market Simulation

As a complex system, electricity market is composed of many interacting units, such as Generation Companies (GENCOs), Transmission Companies (TRANSCOs), Distribution Companies (DISCOs), Load Service Entities (LSEs), and Independent System Operator (ISO) etc. The overall interaction patterns generated among the constituents can exhibit extreme complexity. A complex system like Electricity Market will show emergent properties arising from the interactions of individual companies (players) that are not properties of the individuals. It is impossible to analytically predict market performance directly from the structure of the constituents and the form of their interaction rules. Multi-agents system is a promising tool in which each player is changing its mode of behavior over time in reaction to (and possibly also in anticipation of) the modes of behavior expressed by other players. Market simulation does imply that agents "optimize". We simulate not an individual person's behavior but a business or a company's function in the market. It is not a simple operation of "addition" or "multiply". It is the individual actions of each company that embodies the simulation of the complete markets. Consequently, the fact that agents are evolving together over time provides a way to observe and understand the dynamic behaviors of electricity market.

3.1. Multi-Agents System

The term "agent" is usually used to describe self – contained program, which can control their own actions, based on their perceptions of their operating environment [8]. Usually agents have some of these characteristics, (a) Autonomous (b) Intelligent (c) Rational (d) Learning ability (e) Social incorporation ability. Multi-agent systems model complex distributed systems as a set of software agents that interact in a common environment. The decomposition of a system into a number of agents lets the system react and adapt better in a changing environment [9].

An agent that can respond to its environment is called adaptive. There are four primary ways of adapting [9]:

1. Reaction – A direct, predetermined response to a particular event or an environmental signal. Typically expressed in the form "when event, if condition(s), then action";
2. Reasoning – A more advanced form of reactive response that uses a set of inferences rules;
3. Learning – Changing that occurs during the lifetime of an agent;
4. Evolution – Changing in a population that occurs over successive generations of agents.

Each agent has its own competencies and knowledge, but it needs to interact with other agents to solve complex problems, avoid conflicts, acquire and share information, among others. Researchers have proposed two models of software agents [9].

1. *Cognitive* approach: each agent contains a symbolic model of the outside world, about which it develops plans and makes decisions in the traditional (symbolic) AI way;
2. *Reactive* approach: simple-minded agents react rapidly to asynchronous events without using complex reasoning.

3.2. Evolutionary Algorithm

Evolutionary algorithms share some common conceptual base of simulating the evolution of individual structures via processes of selection and reproduction. Several different types of evolutionary algorithms were developed independently. These include genetic programming (GP), evolutionary programming (EP), evolutionary strategies (ES) and genetic algorithms (GA) [10]. This paper focuses on the following algorithms: Genetic Algorithms, Evolutionary Programming, and Particle Swarm.

3.2.1. Genetic Algorithm

Genetic algorithms (GAs), developed by Holland (1975), have traditionally used a more domain independent representation, namely, bit-strings. Genetic algorithms are performance driven method of finding useful structures with a computer, based loosely on the theory of evolution. In evolution successful creatures mate, blending their genes, said genes then undergoing a small number random changes via mutation. A GA uses selection to pick parents in some sort of relation to their quality. It blends structures via a process called crossover. Small changes are accomplished by mutation [7].

Crossover

Crossover selects bits from each of two chromosomes to produces new chromosomes. There are several kinds of options for performing crossover. The "No crossover" makes the new chromosomes copies of the old chromosomes. When this is used, all innovation in the search for better bidding strategies results from mutations. One point crossover chooses a random position in the chromosomes and exchanges the bits after that point. Two-point crossover chooses a pair of positions in the chromosomes and exchanges those bits between the positions. Uniform crossover either swaps or leaves alone the bits in the chromosomes at each position with 50-50 probabilities. This form of crossover is probably best for efficient search but is computationally intensive, as it requires a lot of random numbers.

Mutation

After crossover has produced new genes the resulting new genes are mutated. The mutation rate is the probability, independent for each bit, that the bit will be flipped. Mutation provides an ongoing source of exploration in the search for better bidding strategies. This is absolutely necessary in a situation where the measure of quality is not absolute.

Selection

Normally there are three selection techniques, proportional selection, rank selection, and roulette selection. All of them choose potential parental strategies in proportion to a number derived from the agent's profit [7].

In the research reported in [7], parameters used to develop GENCOs' bids are evolved using a GA**.** Each member of the GA population corresponds to a GENCO participating in an auction. There are three distinct evolving parts, or genes, for each of the GENCOs. First, the number of 1 *MW* contracts to offer at each round of bidding is evolving. This gene is filled with integer values. Valid integers are between 0 and a maximum value that corresponds to that GENCOs' maximum capability divided by the number of rounds of bidding. Secondly, bid multipliers for each round of bidding are evolved. These bid multipliers are used in combination with the GENCOs' costs and their expected equilibrium price to develop a bid. This gene is represented by binary strings that are mapped to a value between the GENCOs' cost and forecasted equilibrium price during the bidding process. Thirdly, there is a gene that selects which prediction technique to use to forecast the equilibrium price. This is an integer valued gene with valid integers being from 0 to 4, since there are 5 classical prediction techniques from which each GENCO may choose. Additional forecasting methods can be incorporated easily

Roulette selection is a parent selection method that chooses more highly fit creatures with a greater probability than the lesser fit creatures. This fitness bias is more pronounced (especially when population sizes are small) than other parent selection methods like tournament selection.

Based on sensitivity tests, three-point crossover was selected to create the children. Crossover is used on both the number of contracts desired and the bid multiplier. The bid multipliers for each GENCO are concatenated together into one string prior to crossover, and three crossover locations are selected randomly from a uniform distribution over the chromosome's entire length.

The standard bit-flip mutation operator is used on the binary strings representing the bid multipliers. The number of contracts gene has the possibility of being mutated by two different mutation operators. The first mutation operator (mutation-A) adds an integer to the existing integer. If the result is not a valid integer, the value is wrapped around, i.e. if the result is greater than the maximum, then the maximum is subtracted from it. The second mutation operator (mutation-B) shuffles the values among the different loci. If a good number is found in one locus, it can spread to other locations more quickly. Mutation on the prediction technique selection gene involves randomly selecting one of the valid predication techniques.

The fitness of each creature is exactly equal to its profit after participating in an auction. A generation level for each GENCO can be determined by the number of contracts that the GENCO was able to obtain during the auction process. Profit becomes the total revenue to generate at that level, minus the total cost. Total revenue is equal to the *sum* of the contract price multiplied by the number of contracts over all rounds of bidding.

At each generation, one half of the population is replaced with the children. Although the parents were not taken strictly from the top half of the population, it is always the creatures on the bottom of the population that are replaced each generation [3].

3.2.2. Evolutionary Programming

Evolutionary Programming (EP) is a technique in the field of evolutionary computation. EP was proposed as a Finite State Machine (FSM) model by L. J. Fogel in 1960s [11]. In that model, the mutation operator of the state of machine is a kind of a uniform distribution. In 1990s, the thinking of evolutionary programming was extended by D. B. Fogel and then EP was made an optimization tool. Now, EP has become a powerful optimization tool and has been applied to many real problems. The purpose of EP is to do a stochastic search in order to seek an optimal solution to an optimization problem [12].

The schematic diagram of the EP algorithm for optimization is depicted in Figure 1 [12]:

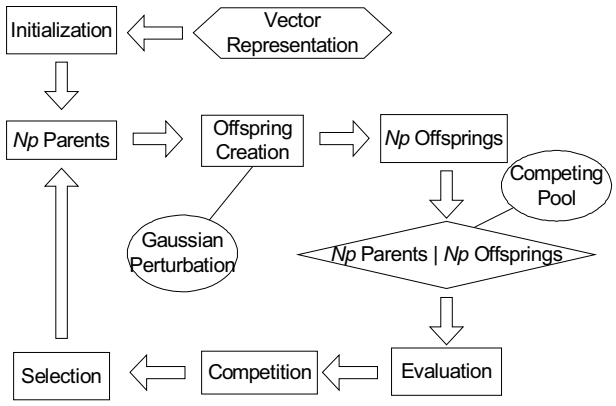

Figure 1: Schematic Diagram of EP algorithm

Vector Representation of Decision Variables [13]:

The real-valued decision variables that need to be determined using the EP algorithm are represented as an n-dimensional vector p which is associated with an n-dimensional objective function $f(p)$. Each vector p is a member of the population that will be evolved during EP computation. Each objective function $f(p)$ is a value of member p that will be calculated during EP computation. Each member p and its objective function $f(p)$ is a probable solution for the optimization problem.

Parent Creation:

An initial starting population consisting of parent individuals $p(i),\ i=1,2,\ldots Np$ is formed by generating a uniform random number (that represents an operating point within the operating range) for each component within each $p(i)$.

Offspring Creation:

An offspring population consisting of offspring individuals (equal to that of parent individuals) $p'(i),\ i=1,2,\ldots Np$ is formed from parent population by disturbing individual parent. The disturbance is done by adding a Gaussian Random Variable of zero mean and pre-calculated Standard Deviation *(the mutation factor)* to each component of $p(i)$.

Objective Function Evaluation:

The fitness of each parent individual $p(i)$ is evaluated by calculating the objective function value of each component. Just like for parent population, the fitness of each offspring individual $p'(i)$ is evaluated by calculating the objective function value $f'(p(i))$ of each component $p'(i)$.

Competition:

The competition scheme is designed such that $p(i)$ and $p'(i)$ members of parent and offspring population stochastically compete with all the members of the two populations based on their objective function values $f(p(i))$ and $f'(p(i))$. The purpose of the competition scheme (or decision rule) is to form a competing pool such that the members (within the two populations) with better solutions have more chances of survival as compared to other members.

Selection:

Based on the competition scheme, the top Np members out of $2*Np$ members from the

competing pool are selected as the survival population to be used as the parent population for next iteration.

Stopping Rule:

The process of forming new population and selecting the ones with better solutions is continued until a specified number of iteration *(Ng)* is reached or the function value of the best solution is not further improved.

3.2.3. Particle Swarm

Particle Swarm (PS) is a recently proposed algorithm by James Kennedy and R. C. Eberhart in 1995, motivated by social behavior of organisms such as bird flocking and fish schooling. PS algorithm is not only a tool for optimization, but also a tool for representing sociocognition of human and artificial agents, based on principles of social psychology. PS as an optimization tool provides a population-based search procedure in which individuals called particles change their position (state) with time. In a PS system, particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own and a neighboring particle's experiences, making use of the best position encountered by itself and its neighbor. Thus, a PS system combines local search methods with global search methods, attempting to balance exploration and exploitation [14].

PS shares many similarities with evolutionary computation techniques such as GA. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PS has no evolution operators such as crossover and mutation. In PS, the potential solutions, called particles, fly through the problem space by following the current optimal particles.

In each iteration particles are updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called *Pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value obtained so far by any particle in the population. This best value is a global best and called *Gbest*. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called *Lbest*.

Assuming a physical *n*-dimensional search space, the position and velocity of individual *i* are represented as the vectors $X_i = (x_{il}, \ldots x_{in})$ and $V_i = (v_{il}, \ldots v_{in})$, respectively, in the PS algorithm. Let $Pbest_i = \left(x_{il}^{Pbest}, \ldots x_{in}^{Pbest}\right)$ and $Gbest_i = \left(x_{il}^{Gbest}, \ldots x_{in}^{Gbest}\right)$, respectively, be the best position of individual *i* and its neighbours' best position so far. Using the information, the updated velocity of individual *i* is modified under the following equation:

$$V_i^{k+1} = \omega V_i^k + c_1 rand_1 \left(Pbest_i^k - X_i^k\right) + c_2 rand_2 \left(Gbest_i^k - X_i^k\right) \quad (1)$$

Where:

V_i^k is the velocity of individual *i* at generation *k*

V_i^{k+1} is the velocity of individual *i* at generation *k+1*

ω is the weight parameter

c_1, c_2 represents the weight factors

$rand_1, rand_2$ are random numbers between 0 and 1

X_i^k is the position of individual *i* at generation *k*

X_i^{k+1} is the position of individual *i* at generation *k+1*

$Pbest_i^k$ best position of individual *i* at generation *k*

$Gbest_i^k$ best position of the group until generation *k*

Each individual moves from the current position to the next one by modified velocity (1) using the following equation:

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (2)$$

Particles' velocities on each dimension are clamped to a maximum velocity V_{max}. If the sum of accelerations would cause the velocity on that dimension to exceed V_{max}, which is a parameter specified by the user, then the velocity on that dimension is limited to V_{max}. The market simulation procedure of PS is shown below in detail:

1) Initialization of a generation of particles at random.
2) Each particle is regarded as a bid of a buyer / seller. All of bids consist of initial position vector;
3) Applying Florida Coordination Group or Linear programming [15] to match buyers and sellers' bids. After obtaining market clear price, profits of each buyer / seller can be calculated consequently.
4) According to the calculated profit of each buyer/seller, update $Gbest_i$ and $Pbest_i$ for each generation;
5) Update the velocity vector by (1);

6) Update the position vector by (2). If inequalities are violated, keep the direction of the velocity vector and shrink the magnitude of the velocity vector by a scalar α (0<α<1) each time, until all of inequalities are satisfied;
7) Go to step 3)

4. Comparative Analysis

In Section 4, an algorithm analysis of GA, EP and PS is discussed in detail. All of the three techniques are compared generation by generation. In GA, Mutation and crossover produce new adaptive components in an essentially random fashion. The direction of changing between current and next generation really is stochastic. Figure 2 shows the procedure that new offsprings are created through crossover and mutation operators.

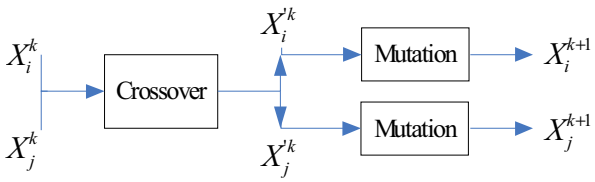

Figure 2: Procedure of New offsprings Creation by GA

In order to relate GA to traditional optimization techniques, such as Newton method etc., it is important to define a "gradient" for GA. Since the direction of changing between current and next generation is uncertain, Simulation is proper to measure the expected "gradient" of GA. Given specific crossover and mutation fashions, execute a large number of simulations, the direction is determined as the difference between X_i^{k+1} and X_i^k.

According to the discussion in the previous section, EP can be described by a dynamic equation.

$$X_i^{k+1} = X_i^k + Gaussian(0,(\varphi_i^k)^2) \quad (3)$$

Where X_i^k represents the bidding of individual *i* at generation *k and* X_i^{k+1} is the bidding of individual *i* at generation *k+1*

The standard deviation φ_i^k is the mutation factor and indicates the range the offspring is created around its parent and is given by:

$$\varphi_i^k = \beta\left(\frac{f_{best}^k}{f_i^k}\right) \quad (4)$$

Where:
β represents the scaling factor
f_{best}^k is the best profit at generation *k*
f_i^k is the profit of individual *i* at generation *k*

The magnitude of φ_i^k is proportional to the magnitude of changing in the offspring as compared to the parent. The bigger the value of φ_i^k, the greater is the changing in X_i^{k+1} as compared to X_i^k and vice versa. According to equation (3), the Gaussian random perturbation term defines the direction of changing in EP, as Newton method does.

Particle swarm dynamic equation (1) and (2) essentially provide a clear definition of direction of changing. Plugging equation (1) into (2), equation (5) shows particle swarm updating procedures:

$$X_i^{k+1} = X_i^k + \omega V_i^k + c_1 rand_1\left(Pbest_i^k - X_i^k\right) + c_2 rand_2\left(Gbest_i^k - X_i^k\right) \quad (5)$$

The gradient of PS contains more information compared with GA and EP. Individual particle has a memory of its own optimal value, $Pbest_i^k$. All particles share a common knowledge, *i.e.* $Gbest_i^k$, the optimal value until current generation. All of particles move within a solution space according to historical global & local best solutions.

EP makes use of a Gaussian random perturbation term to update each generation. All of the candidates' updating directions form an *n*-dimension random vector. EP does not apply local information; however each generation does share a common knowledge implicitly. The mutation factor φ_i^k is proportional to the ratio between the best profit and individual profits up to now. The bigger the ratio between the best profit and individual profits, *i.e.* individual profits is far from the best one, the greater is the change in X_i^{k+1} as compared to X_i^k and vice versa.

The "gradient" of GA employs neither global nor local historical information. Crossover and mutation operators bring in new sense of change in direction through the use of a random alteration of solution value. Therefore it is necessary to apply simulation to measure the expected "gradient" of GA.

EP and PS can be more appropriate than GA if solution space is well defined, for both of them utilize previous global information. Furthermore, PS balances local and global searching, displays more advantages. However more operations are needed.

On the other hand, if solution space is highly skewed and tortuous, the previous global information may be wrong and mislead the searching procedure in PS and EP. Thus, GA will outweigh EP and PS in the case.

Table 2 summarizes the major manipulations of three techniques during one generation. N is the population size, M is the number of encoding bits in GA, and L is the number of players. Mutation prediction is applied to accelerate GA [16]. For the sake of simplicity, "One Point Crossover" operation is executed in GA (Table 1).

Position	1	2	3	4	5	6	7	8	9	10
Parent 1	0	1	1	0	1	1	0	0	0	0
Parent 2	1	1	0	1	1	1	1	0	0	1
Cut Pattern	1	1	1	2	2	2	2	2	2	2
Offspring	0	1	1	1	1	1	1	0	0	1

Table 1: One Point Crossover Operation

	GA	EP	PS
# of Calling Random Number Generator	*1.5*N*L*	*N*L*	*2*N*L*
# of Multiplication	_	*3*N*L*	*5*N*L*
Local historical information	*No*	*No*	*Yes*
Global historical information	*No*	*Yes*	*Yes*

Table 2: Compare of Manipulations in GA, EP and PS in One Generation

The major manipulations discussed in the paper include the number of calls to a random number generator (RNG) and the number of multiplication. As an example of the importance of reducing the number of calls to a RNG, one should consider the amount of computer used for even a simple problem like economic dispatch. A company with 100 units, where each unit is represented by a 20-bit gene, there are 400 solutions in the population, would require 800000 call to the RNG for each generation. Since there are a number of generations required to find a solution, our experience is between 100 to 500, the total number of calls to the RNG would be between 80 to 400 million calls. The MATLAB RNG requires a variable amount of time, the average from experiments, is approximately 0.0000158 second. However, this value is very volatile as found by experiment. Thus, approximately 21 minutes to 1.76 hours. Thus, most ED solutions by GA are cut short for practical reasons.

5. Conclusions

This paper discusses and compares the multi-agent system and ALIFE algorithms in the application of electricity market simulation. This paper describes how to apply three techniques based on the idea of evolutionary computation to emulate market participants' behaviors. They are Genetic Algorithm, Evolutionary Programming and Particle Swarm. Genetic Algorithm is a promising tool to model market agents for its adaptivity. Evolutionary programming is close to genetic algorithm, but a Gaussian random perturbation takes place of crossover and mutation operators. Gaussian perturbation operator makes evolutionary programming more refined to emulate real players. Particle swarm navigates through the problem solution space by following the current optimal particle and individual previous best solution. Particle swarm is a combinatory method of local search with global search. All of the three techniques are compared according to the characteristics of solution space.

6. References

[1] Spears, W.M., Jong, K.A.D., Baeck, T., Fogel, D.B. and Garis, H.de, "An Overview of Evolutionary Computation," *In Proceedings of the European Conference on Machine Learning*, 1993.

[2] Post, D.L., Coppinger, S.S. and Sheble, G.B., "Application of auctions as a pricing mechanism for the interchange of electric power", *IEEE Transactions on Power Systems*, Volume 10, No. 3, Aug. 1995 Page(s): 1580 – 1584.

[3] Richter, C.W. and Sheble, G.B., "Genetic algorithm evolution of utility bidding strategies for the competitive marketplace", *IEEE Transactions on Power Systems*, Volume 13, Issue 1, Feb. 1998 Page(s): 256 – 261.

[4] Praca, I., Ramos, C. and Vale, Z.A., "Competitive electricity markets: simulation to improve decision-making", *Power Tech Proceedings*, *2001 IEEE Porto* Volume 1, 10-13 Sept. 2001 Page(s): 6.

[5] Lam, Y.C. and Wu, F.F., "Simulating electricity markets with Java", *Power Engineering Society 1999 Winter Meeting, IEEE*, Volume: 1 31 Jan - 4 Feb 1999 Page(s): 406-410.

[6] Wood, A.J. and Wollenberg, B. F., *Power Generation Operation and Control*, 2nd ed., Wiley, New York, 1996.

[7] Sheble, G.B., "Computer Simulation of Adaptive Agents for an Electric Power Auction", EPRI Report, 1997.

[8] Ferber J., *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, 1999.

[9] Deuel R., "Adaptive Agents and Multiagent Systems," *IEEE Distributed Systems Online*, Vol. 5, No. 7, July 2004.

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

[11] Koza, J., Goldberg, D.E., Fogel, D. and Riolo, R., (ed.), *Proceedings of the First Annual Conference on Genetic Programming*, MIT Press, 1996.

[12] Yang, H., Yang, P. and Huang, C., "Evolutionary Programming Based Economic Dispatch for Units with Non-Smooth Fuel Cost Functions", *IEEE Transactions on Power Systems*, Vol. 11, No. 1, Feb 1996, pp. 112-118.

[13] Wong, K. and Yuryerich, J., "Evolutionary-Programming-Based Algorithm for Environmentally-Constrained Economic Dispatch", *IEEE Transactions on Power Systems*, Vol. 12, No. 2, May 1998, pp. 301-306.

[14] Park, J., Lee, K., Shin, J., and Lee, K. Y., "A Particle Swarm Optimization for Economic Dispatch with Nonsmooth Cost Functions", *IEEE Transactions on Power Systems*, Vol. 20, Feb. 2005, pp. 34–42.

[15] Fahd, G., Richards, D. A., and Sheble, G. B., "The Implementation of an Energy Brokerage System using Linear Programming," *IEEE Transactions on Power Systems*, Vol. 7, Feb. 1992, pp. 90–96.

[16] Gao, F, Sheble, G.B., "Economic Dispatch Algorithms for Thermal Unit System Involving Combined Cycle Units", 15th Power Systems Computation Conference, Liège, Belgium, 2005.