JOURNAL OF INFORMATION, KNOWLEDGE AND RESEARCH IN ELECTRONICS AND COMMUNICATION

OPTIMIZATION OF STATE REDUCTION FOR STOCHASTIC FINITE STATE SYSTEM

[1] SAUMYA DAS, [2] OM PRAKASH, [3] DR. BHAVIN S. SEDANI

[1] Assistant Professor, SMIT, SMU
[2] Assistant Professor, JJTU, Rajasthan
[3] Head and Associate Professor, Department of Electronics and Communication Engineering, VVP Engineering College, Rajkot - Gujarat

ABSTRACT *: This paper deal with Real world complex problems have no predefined procedure for solving. It is impossible to define and determine the exact solution of these problems. These are called Stochastic or probabilistic system. Consider a Fossil Fired (Oil, Coal, and Gas) generator power system unit. To get the desired output power at lowest cost, the power system unit is designed as a stochastic finite state system.*

Keywords : Particle Swarm Optimization, Constraint Optimization Problems, Unconstraint Optimization Problem And Economic Load Dispatch Problem.

INTRODUCTION

In the theory of computation, a **stochastic finite state system** or **nondeterministic finite automaton (NFA)** is a finite state machine where for each pair of state and input symbol there may be several possible next states. This distinguishes it from the deterministic finite automaton (DFA), where the next possible state is uniquely determined. Although the DFA and NFA have distinct definitions, it may be shown in the formal theory that they are equivalent, in that, for any given NFA, one may construct an equivalent DFA, and vice-versa: this is the powerset construction. Both types of automata recognize only regular languages. Non-deterministic finite state machines are sometimes studied by the name subshifts of finite type. Non-deterministic finite state machines are generalized by probabilistic automata, which assign a probability to each state transition. More recently, stochastic formulations of receding horizon control are showing great promise in areas such as dynamic unreliable resource allocation [1], supply chains [2], portfolio optimization [3]–[5], dynamic hedging [6], sustainable development [7], and polymerization reactors [8]. The implementation of receding horizon control in the different applications spans a range of approaches, from open-loop control [3], [4], to a linear feedback form of control [8], to a full stochastic programming implementation [6], [9], [10].

Nondeterministic finite automata were introduced in 1959 by Michael O. Rabin and Dana Scott, who also showed their equivalence to deterministic finite automata

Intuitive definition

An NFA, similar to a DFA, consumes a string of input symbols. For each input symbol it transitions to a new state until all input symbols have been consumed.

Unlike a DFA, it is non-deterministic in that, for any input symbol, its next state may be any one of several possible states. Thus, in the formal definition, the next state is an element of the power set of states. This element, itself a set, represents some subset of all possible states to be considered at once.

An extension of the NFA is the **NFA-lambda** (also known as **NFA-epsilon** or the **NFA with epsilon moves**), which allows a transformation to a new state without consuming any input symbols. For example, if it is in state 1, with the next input symbol an *a*, it can move to state 2 without consuming any input symbols, and thus there is an ambiguity: is the system in state 1, or state 2, before consuming the letter *a*? Because of this ambiguity, it is more convenient to talk of the set of possible states the system may be in. Thus, before consuming letter *a*, the NFA-epsilon may be in any one of the states out of the set {1,2}. Equivalently, one may imagine that the NFA is in state 1 and 2 'at the same time': and this gives an informal hint of the powerset construction: the DFA equivalent to an NFA is defined as the one that is in the state q={1,2}. Transformations to new states without consuming an input symbol are called **lambda transitions** or **epsilon transitions**. They are usually labeled with the Greek letter λ or ε.

The notion of accepting an input is similar to that for the DFA. When the last input symbol is consumed, the NFA accepts if and only if there is *some* set of transitions that will take it to an accepting state. Equivalently, it rejects if, no matter what transitions are applied, it would not end in an accepting state.

Formal definition

Two similar types of NFAs are commonly defined: the NFA and the *NFA with ε-moves*. The ordinary NFA is defined as a 5-tuple, (Q, Σ, T, q_0, F), consisting of

- a finite set of states Q

- a finite set of input symbols Σ
- a transition function $T : Q \times \Sigma \rightarrow P(Q)$.
- an *initial* (or *start*) state $q_0 \in Q$
- a set of states F distinguished as *accepting* (or *final*) *states* $F \subseteq Q$.

Here, $P(Q)$ denotes the power set of Q. The *NFA with ε-moves* (also sometimes called *NFA-epsilon* or *NFA-lambda*) replaces the transition function with one that allows the empty string ε as a possible input, so that one has instead

$T : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$.

It can be shown that ordinary NFA and NFA with epsilon moves are equivalent, in that, given either one, one can construct the other, which recognizes the same language.

1.2 Properties of NFA-ε

For all $p, q \in Q$, one writes $p \xrightarrow{\epsilon} q$ if and only if q can be reached from p by going along zero or more ε arrows. In other words, $p \xrightarrow{\epsilon} q$ if and only if there exists $q_1, q_2, \cdots q_k \in Q$ where $k \geq 0$ such that

$q_1 \in T(p, \epsilon), q_2 \in T(q_1, \epsilon), \cdots q_k \in T$

.

For any $p \in Q$, the set of states that can be reached from p is called the **epsilon-closure** or **ε-closure** of p, and is written as

$E(\{p\}) = \{q \in Q : p \xrightarrow{\epsilon} q\}$.

For any subset $P \subset Q$, define the ε-closure of P as

$$E(P) = \bigcup_{p \in P} E(\{p\})$$

.

The epsilon-transitions are transitive, in that it may be shown that, for all $q_0, q_1, q_2 \in Q$ and $P \subset Q$, if $q_1 \in E(\{q_0\})$ and $q_2 \in E(\{q_1\})$, then $q_2 \in E(\{q_0\})$.

Similarly, if $q_1 \in E(P)$ and $q_2 \in E(\{q_1\})$ then $q_2 \in E(P)$

Let x be a string over the alphabet Σ∪{ε}. An NFA-ε M accepts the string x if there exist both a representation of x of the form $x_1x_2 \ldots x_n$, where $x_i \in (\Sigma \cup \{\varepsilon\})$, and a sequence of states $p_0, p_1, \ldots, p_n$, where $p_i \in Q$, meeting the following conditions:

1. $p_0 \in E(\{q_0\})$
2. $p_i \in E(T(p_{i-1}, x_i))$ for $i = 1, \ldots, n$
3. $p_n \in F$.

Implementation

There are many ways to implement a NFA:

- Convert to the equivalent DFA. In some cases this may cause exponential blowup in the size of the automaton and thus auxiliary space proportional to the number of states in the NFA (as storage of the state value requires at most one bit for every state in the NFA)
- Keep a set data structure of all states which the machine might currently be in. On the consumption of the last input symbol, if one of these states is a final state, the machine accepts the string. In the worst case, this may require auxiliary space proportional to the number of states in the NFA; if the set structure uses one bit per NFA state, then this solution is exactly equivalent to the above.
- Create multiple copies. For each n way decision, the NFA creates up to $n - 1$ copies of the machine. Each will enter a separate state. If, upon consuming the last input symbol, at least one copy of the NFA is in the accepting state, the NFA will accept. (This, too, requires linear storage with respect to the number of NFA states, as there can be one machine for every NFA state.)
- Explicitly propagate tokens through the transition structure of the NFA and match whenever a token reaches the final state. This is sometimes useful when the NFA should encode additional context about the events that triggered the transition. (For an implementation that uses this technique to keep track of object references have a look at Tracematches.)

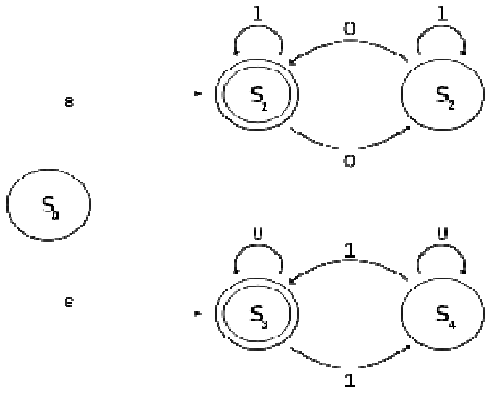
Example

The following example explains a NFA M, with a binary alphabet, which determines if the input contains an even number of 0s or an even number of 1s. (Note that 0 occurrences is an even number of occurrences as well.) Let $M = (Q, \Sigma, T, s_0, F)$ where

- $\Sigma = \{0, 1\}$,
- $Q = \{s_0, s_1, s_2, s_3, s_4\}$,
- $E(\{s_0\}) = \{ s_0, s_1, s_3 \}$
- $F = \{s_1, s_3\}$, and
- The transition function T can be defined by this state transition table:

	0	**1**	**ε**
S_0	{}	{}	$\{S_1, S_3\}$
S_1	$\{S_2\}$	$\{S_1\}$	{}
S_2	$\{S_1\}$	$\{S_2\}$	{}
S_3	$\{S_3\}$	$\{S_4\}$	{}
S_4	$\{S_4\}$	$\{S_3\}$	{}

The state diagram for M is:



M can be viewed as the union of two DFAs: one with states $\{S_1, S_2\}$ and the other with states $\{S_3, S_4\}$.
The language of *M* can be described by the regular language given by this regular expression:

$$(1^*(01^*01^*)^*) \cup (0^*(10^*10^*)^*)$$

Application of NFA-ε

NFAs and DFAs are equivalent in that if a language is recognized by an NFA, it is also recognized by a DFA and vice versa. The establishment of such equivalence is important and useful. It is useful because constructing an NFA to recognize a given language is sometimes much easier than constructing a DFA for that language. It is important because NFAs can be used to reduce the complexity of the mathematical work required to establish many important properties in the theory of computation. For example, it is much easier to prove the following properties using NFAs than DFAs:

- The union of two regular languages is regular.
- The concatenation of two regular languages is regular.
- The Kleene Closure of a regular language is regular.

State Reduction and State Assignment

If one is able to reduce the total number of states, one may be able to save on the number of flip-flops required for a design. This is the optimal situation. For example if a finite state machine drops from 7 states to 4 states and compact state assignments are used, the design drops from three flip-flops to two flip-flops. A sub optimal situation is when the number of states is reduced, but the number of flip-flops is not. This does add don't cares to the combinational logic that generates the next state equations. This will most likely drop the over all cost of the finite state machine. Once the number of states is at a minimum, then a judicious assignment of states may
further reduce the cost of the next state equations and/or the cost of the output equations. A set of heuristic rules is proposed where each rule is directed toward the reduction of the combinational logic in the finite state machine design. As opposed to compact state assignments, one may propose a one-hot state assignment. One-hot is a set of state assignments in which a unique bit is one in the assignment for each state. This often leads to a reduction in the logic cost for the outputs, because in one and only one state a given output is asserted.

Given a stochastic machine description,the state set can always be partitioned into classes of equivalent states by a finite number of calculations. If equivalence classes containing two or more states are found, it should be possible to condense the machine description in such a way has to leave the family of distinct input-output relations invariant.
If to each state of a stochastic machine M there corresponds an equivalent state of machine N and to each state of N there corresponds an equivalent state of machine M, Which say that M & N are state equivalent machines. Among the machines which are state equivalent to a given machine M those having the smallest no of states are called Reduced Forms of M. A machine for which any two states are distinguishable is said to be in reduced form. The terminology is consistant since the reduced forms of any machine M are precisely those machines which are state equivalent to M and in reduced form.
troduction
Optimization problem formulation
Economic load dispatch (ELD) is an important topic in the operation of power plants which helps to build up effective generating management plans. The ELD problem has non-smooth cost function with equality and inequality constraints which make it difficult to be effectively solved [14]. Real cost functions are more complex than conventional second order cost functions when multi-fuel operations, valve-point effects, accurate curve fitting, etc., are considering in deregulated changing market [16].
The ELD problem may be expressed by minimizing the fuel cost of generator units under constraints. Depending on load variations, the output of generators has to be changed to meet the balance between loads and generation of a power system. The power system model consists of *n* generating units already connected to the system [13]. The fuel-cost function without valve-point loadings of the generating units is given by

$$\mathrm{F}(x_i) = a_i x_i^2 + b_i x_i + c_i \quad Rs/h \tag{7}$$

A cost function is obtained based on the ripple curve for more accurate modeling which contains higher order nonlinearity and discontinuity due to the valve point effect and should be refined by a sine function. The ELD problem can be expressed as [11,1 2, 17]:
Minimize operation cost;

$$F = \sum_{i=1}^{n} (c_i + b_i x_i + a_i x_i^2) + \left| e_i \sin\left(f_i(x_i^{min} - x_i)\right)\right| {}^{Rs}/_{h} \tag{8}$$

Subjects to;

i. Power balance constraints

$$\sum_{i=1}^{n} x_i - P_D - P_L = 0 \tag{8a}$$

and,

$$P_L = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i B_{ij} x_j + \sum_{i=1}^{n} B_{oi} x_i + B_{oo} \quad (8b)$$

ii. ii. Generating capacity constraints

$$x_i^{min} \le x_i \le x_i^{max};\ i = 1,2,\dots,n \quad (8c)$$

where,

- a_i, b_i and c_i are the cost coefficients of the i^{th} generator
- n is the number of generators
- x_i is the real power output of the i^{th} generator (MW)
- $F(x_i)$ is the operating cost of unit i (Rs/h)
- P_L is the transmission losses (MW)
- x_i^{max} is the maximum generation output of the i^{th}generator
- x_i^{min} is the minimum generation output of the i^{th} generator
- B_{ij}, B_{oi}andB_{oo} are the B-coefficients
- $P_D$is the total demand (MW).

The key factor in solving an ELD problem is how to handle the several constraints relating to the problem. Over the last few decades, kinds of approaches had been proposed to handle the constraints. These can be grouped into four categories: ideas that preserve the feasibility of solutions, penalty-based approaches, methods that clearly distinguish between feasible and unfeasible solutions, and hybrid techniques [15]. In this thesis the penalty function is adopted to address the constraints in an ELD problem. The introduction of the penalty term enables to transform a constrained optimization problem into an unconstrained one. As a result, the fuel cost function is written as

$$F_m(x_i, r_k) = \sum_{i=1}^{n} F_i(x_i) + r_k . h^2 \quad (9)$$

,

The value of the penalty coefficient r_k is checked at each iteration, and h is the equality constrained defined as

$$h = \sum_{i=1}^{n} x_i - P_D - P_L \quad (10)$$

Most of these methods were based on penalty formulations that transform Eq. (8b) into an unconstrained function $F_m(x_i, r_k)$ as shown in Eq. (9), which consisting of a sum of the objective and the constraints weighted by penalties, and use PSO to minimize $F_m(x_i, r_k)$.

A 13-generators electric power system considering valve-point loading effect has been studied in this test. In this case, the load demand, P_D is taken as 1800MW. 13- generators result are given in Tables 4.8. Although the acquired best solution is not guaranteed to be the global solution, the Particle swarm optimization succeeded in finding the satisfactory solution. The respective operating cost coefficients for each generator are given in Table 1

Generator number	Generator limits		Fuel cost coefficients		
	x_i^{min} (MW)	x_i^{max} (MW)	a_i (Rs/h)	b_i (Rs/MWh)	c_i (Rs/MW²h)
1	00	680	0.00028	8.10	550
2	00	360	0.00056	8.10	309
3	00	360	0.00056	8.10	307
4	60	180	0.00324	7.74	240
5	60	180	0.00324	7.74	240
6	60	180	0.00324	7.74	240
7	60	180	0.00324	7.74	240
8	60	180	0.00324	7.74	240
9	60	180	0.00324	7.74	240
10	40	120	0.00284	8.6	126
11	40	120	0.00284	8.6	126
12	55	120	0.00284	8.6	126
13	55	120	0.00284	8.6	126

Swarm size is taken as 200, plenty parameter is set to 100, error is 10^{-4} for this problem, c_1 and c_2 are taken as 2. The obtained results are presented Table 2.

S.N0	Iterations	F(P)
1	40	22973.91
2	50	22983.79
3	60	23455.60
4	80	22550.63
5	100	22024.40
6	110	22145.90
7	120	18189.83
8	130	18177.25
9	140	18161.07

Table 2: Result of economic load dispatch problem for 13-generators

Economic load dispatch problem for 13-generators data is graphically shown in Figure 1.

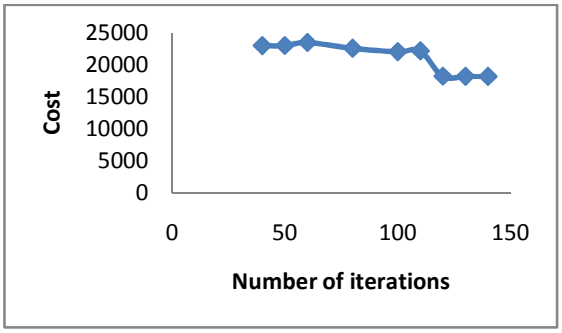


Fig. 1: Variation of cost with respect number of iterations for 13-generator power system

Particle swarm optimization appears to be the most efficient in terms of faster convergence rate and quality of solution, which makes it to be much efficient in finding the global optimum.

CONCLUSION

This paper proposed PSO for power system ELD problem considering as a stochastic finite state system. The results show that the PSO is better than other in terms of the speed and accuracy. it greatly enhances the searching ability and efficiently manages the system constraints. It makes problem easier because the probability of finding a solution by chance is large. The successful optimizing performance on the validation data set illustrates the efficiency of the approach and shows that it can be used as a reliable tool for ELD problem

REFERENCES

[1] D. A. Castanon and J. M. Wohletz,[2002] "Model predictive control for dynamic unreliable resource allocation," in Proc. Conf. Decision Control, Las Vegas, NV, USA, pp. 3754–3759.

[2] P. Seferlis and N. F. Giannelos[2004]"A two-layered optimisation-based control strategy for multi-echelon supply chain networks," Comp. Chem. Eng., vol. 28, pp. 799–809.

[3] F. Herzog [2005] "Strategic Portfolio Management for Long-Term Investments: An Optimal Control Approach," Ph.D. dissertation, ETH Zurich, Zurich, Switzerland.

[4] F. Herzog, S. Keel, G. Dondi, L. M. Schumann, and H. P. Geering,[2006] "Model predictive control for portfolio selection," in Proc. Amer. Control Conf., Minneapolis, MN, pp. 1252–1259.

[5] F. Herzog, G. Dondi, and H. Geering,[2007] "Stochastic model predictive control and portfolio optimization," Int. J. Theor. Appl. Finance, vol. 10, no. 2, pp. 203–233.

[6] P. Meindl and J. A. Primbs,[2004] "Dynamic hedging with stochastic volatility using receding horizon control," in Proc. Financial Eng. Appl., Cambridge, MA, Nov. 8–10, pp. 142–147.

[7] P. D. Couchman, M. Cannon, and B. Kouvaritakis [2006] "MPC as a tool for sustainable development integrated policy assessment," IEEE Trans. Automat. Control, vol. 51, no. 1, pp. 145–149.

[8] D. H. van Hessem and O. H. Bosgra,[2004] "Closed-loop stochastic model predictive control in a receding horizon implementation on a continuous polymerization reactor example," in Proc. Amer. Control Conf., Boston pp. 914–919.

[9] D. M. de la Peña, A. Bemporad, and T. Alamo,[2005] "Stochastic programming applied to model predictive control," in Proc. 44th IEEE Conf. Decision Control Eur. Control Conf., Sevilla, Spain, pp. 1361–1366.

[10] F. Herzog, G. Dondi, S. Keel, L. Schumann, and H. Geering, [2007]"Solving ALM problems via sequential stochastic programming," Quantitative Finance, vol. 7, no. 2, pp. 231–244.

[11] Dhillon, J.S. and Kothari, D.P. [2004], Power System Optimization, Prentice Hall of India.

[12] Dao-Hyun Choi and Se-Young Oh [2000], A new mutation rule for Evolutionary Programming motivated from back propagation learning, IEEE transaction on Evolutionary Programming, vol.4, no.2, pp.188-190.

[13]Kennedy, J., Eberhart, R. C. and Shi, Y. [2001], Swarm intelligence. San Francisco: Morgan Kaufmann Publishers.

[14]Meng, Ke, Wang, H. G., Dong, Z. Y. and Wong, K. P. [2010], Quantum-Inspired Particle Swarm Optimization for Valve-Point Economic Load Dispatch, IEEE transactions on power systems, vol. 25, no. 1, pp.215-222.

[15]Parsopoulos, K. E., and Vrahatis, M. N. [2002], Particle swarm optimization method for constrained optimization problems. In P. Sincak, J. Vascak, V. Kvasnicka & J. and Pospichal (Eds.), intelligent technologies–theory and application (New trends in intelligent technologies). Frontiers in artificial intelligence and applications, vol. 76, pp. 214–220. Amsterdam: IOS Press, ISBN:1-58603-256-9.

[16]Saber A. Y., Chakraborty S., Razzak S.M. Abdur and Senjyu T. [2009], Optimization of economic load dispatch of higher order general cost polynomials and its sensitivity using modified particle swarm optimization, Electric Power Systems Research, vol. 79, no. 1, pp. 98–106.

[17]Vlachogiannis, J. G. and Lee, K. Y. [2009], Economic Load Dispatch—A Comparative Study on Heuristic Optimization Techniques With an Improved Coordinated Aggregation-Based PSO, IEEE Transactions On Power Systems, vol. 24, no. 2, pp. 991-1001