Adaptive Variable Parameter Selection Probability of Genetic Algorithm for Fuzzy Modeling

Makoto Ohki, Shigehiko Hasegawa, Naruhito Kishimoto and Masaaki Ohkita
Department of Electrical and Electronic Engineering, Faculty of Engineering, Tottori University
101, 4 Koyama-Minami, Tottori, Tottori 680-8552 Japan
Phone: +81-857-31-6701, Fax.: +81-857-31-0880
email: mohki@ele.tottori-u.ac.jp

ABSTRACT: We propose a new mutation operator with an adaptive variable parameter selection probability in genetic algorithms (GA). The GA is applied to optimize the fuzzy reasoning in this paper, or fuzzy modeling. There are two kinds of problem in the fuzzy modeling, configuring fuzzy rules and optimizing shaps of membership functions which are considered to be combinatorial and numerical optimization problems. Although GA can be simultaneously applied to both two kinds of problem, GA has not been applied to optimize the shape of the membership functions in many cases, because GA can not quickly obtain a solution which can be satisfied. To quickly obtain such a solution using GA, we have already proposed a variable bit-selection probability in the mutation operator. However, a parameter selection probability, one of factors to configure the mutation operator, greatly affects progress of the optimization. To improve this difficulty, we propose the adaptive variable parameter selection probability which serves to a quick convergence to optimized solutions.

KEYWORDS: genetic algorithm, fuzzy modeling, mutation operator, parameter selection probability

1. INTRODUCTION

A problem on a fuzzy modeling[1] consists of two kinds of problem, configuring fuzzy rules and optimizing shaps of membership functions which are considered to be combinatorial and numerical optimization problems. The genetic algorithm (GA) [2, 3] is able to be applied to both these problems. In many research works, the GA is applied only to optimize the configuration of the fuzzy rules, while another optimizing algorithm, such as a steepest descent method, is applied to optimize shapes of the membership functions. In their background, since a mutation operator of the GA always globally searches in a solution space, the GA slowly optimizes the fuzzy reasoning. In the case when the fuzzy model is applied to such a field of control, the global optimum is not always required. There are many cases that a local optimum gives appropriate performance. To obtain such a local optimum, the authors have proposed a method whose bit-selection probability (BSP) in the mutation operator is varied depending on progress of the optimization, where we call such a BSP variable bit-selection probability (VBSP) [4, 5, 6, 7]. Since the VBSP allows the GA to search around a good solution candidate in a mean of a humming distance, the local optimum can be quickly obtained.

We have investigated into a parameter selection probability (PSP), one of factors configuring the algorithm, and tried some values of the PSP. This investigation shows that the progress and the result of the optimization depends on the value of the PSP. Besides, the adequate value of the PSP is different to individual problem. To obtain the adequate value of the PSP, preliminary implementations by trial and error is required. In this paper, we propose a method whose PSP is adaptively varied depending on the progress of the optimization to improve this difficulty. Now, we call such PSP adaptively variable PSP (APSP).

2. FUZZY MODELING BY THE GA

In this paper, the simplified fuzzy reasoning is optimized. One of rules in the simplified fuzzy reasoning is represented by the following expression:

$$rule\ \langle i_1 i_2 ... i_N \rangle : if\ x_1\ is\ A_{1i_1}\ and, ...,\ and\ x_N\ is\ A_{Ni_N}\ then\ y\ is\ w_{\langle i_1 i_2 ... i_N \rangle}. \tag{1}$$

where $x_1, ..., x_N$ denotes input variables, $A_{1i_1}, ..., A_{Ni_N}$ denote MSFs defined for their input variables, y denotes an output variable or reasoning result and $w_{\langle i_1 i_2 ... i_N \rangle}$ denotes a consequent crisp value of the rule $\langle i_1 i_2 ... i_N \rangle$.

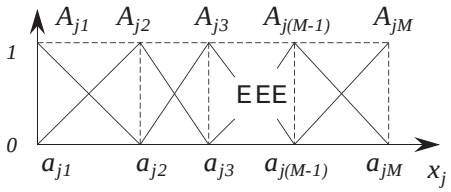


Figure 1: Antecedent MSFs.

```
procedure Convetional_GA
var
  q : integer ; {generation}
const
  Pp = P1 ; {PSP}
  Pb = P2 ; {BSP}
begin
  q := 0 ;
  while q<qf do begin {qf:finishing generation}
    Get performance of each individual gk ;
    Select parents e, s1, s2, ... ;
    Apply crossover operator ;
    Apply mutation operator ;
    q := q+1
  end
end.
```

Figure 2: Procedure of the Conventional GA.

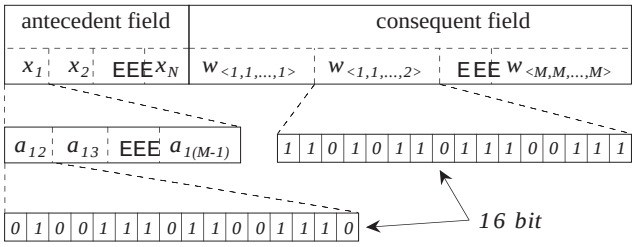


Figure 3: Genotype coding.

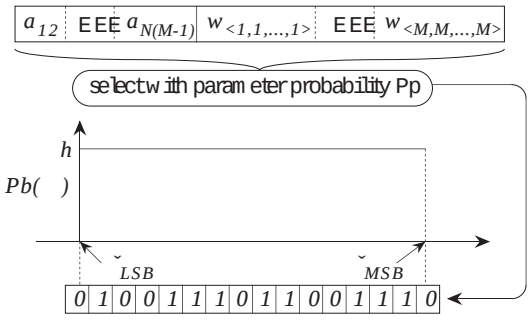


Figure 4: Conventional mutation operator.

The identifier of the rule $\langle i_1 i_2 ... i_N \rangle$ shows that the MSF A_{ji_j} is defined for the input variable x_j in this rule. The MSF of the antecedent part is defined by a triangular function shown in Fig. 1. The reasoning result, y, are defined by using a compatibility of the antecedent part, $\mu_{\langle i_1 i_2 ... i_N \rangle}$, by the following equation:

$$y = \sum_{\langle i_1 i_2 ... i_N \rangle} \mu_{\langle i_1 i_2 ... i_N \rangle} \cdot w_{\langle i_1 i_2 ... i_N \rangle} \Big/ \sum_{\langle i_1 i_2 ... i_N \rangle} \mu_{\langle i_1 i_2 ... i_N \rangle} \tag{2}$$

where the compatibility $\mu_{\langle i_1 i_2 ... i_N \rangle}$, is defined by multiplication of the values of MSF, $\prod_{j=1}^{N} A_{ji_j}(x_j)$.

The fuzzy reasoning configured with such a set of the fuzzy rule is coded into a genotype shown in Fig.3. The gene, as an individual, consists of $N \cdot (M-2)$ parameters a_{ji_j} $(j = 1, 2, \cdots, N \ and \ i_j = 2, \cdots, M-1)$ giving the antecedent MSFs and M^N parameters $w_{\langle i_1 i_2 ... i_N \rangle}$ $(i_j = 1, 2, \cdots, M)$ giving the consequent crisp values. Each parameter is given by a 16 bit length binary number, for example, a short integer number in the C language can be also handled as a 16 bit length binary number. In fact, a scaling factor is required to convert the parameters to the phenotype parameters representing the fuzzy model.

The simple GA shown in Fig.2 is used for the optimization. In the GA, the following two genetic operators are applied. A crossover operator is implemented as follows. Consider two individuals applied to the crossover operator. Two boundaries between the parameters each other are selected in the antecedent and the consequent fields respectively. The series of parameters divided by those crossover boundaries are substituted each other. Then new two individuals are obtained.

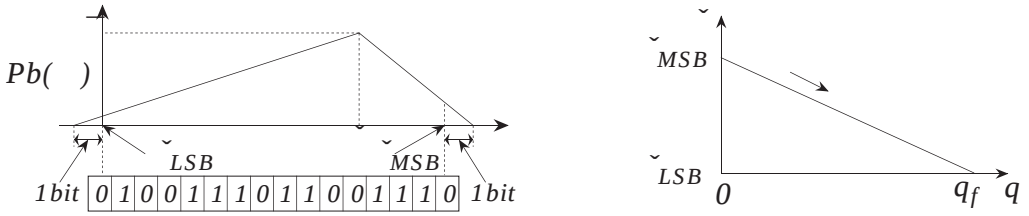
A mutation operator is implemented as shown in Fig.4 where the horizontal axis, ξ, denotes the bit position of the parameter, ρ_{LSB} and ρ_{MSB} denote the bit positions of the least significant bit (LSB) and most significant bit (MSB) respectively and h denotes the height of the function $Pb(\xi)$ which gives the value of the BSP. Some of the parameters are selected according to the PSP, Pp. The value of the bit of these selected parameters are inverted according to the BSP, Pb. The PSP and the BSP in the conventional GA are given by constant. Then the conventional GA globally searches in a solution space.

3. THE GA USING THE VBSP

Since the GA always globally searches in a solution space by means of such a fixed BSP, the optimization becomes slow. It is caused by the conventional GA whose mutation operator can not search near a good solution found at the current generation. There are many cases that some local optima give appropriate performance.

To add such a local search function to the GA, we have proposed the VBSP which varies depending on the number of generation[6].

The VBSP is defined by a triangular function shown in Fig.5 (a). Depending on the number of the generation, the bit position of the top of the triangular function, ρ, is varied by a linear function shown in Fig.5 (b). While the optimization progresses for q_f generations, the bit position ρ gradually moves from ρ_{MSB} to ρ_{LSB}.



(a) Shape of the VBSP. (b) The top position of the VBSP, ρ.

Figure 5: Definition of the VBSP.

4. NUMERICAL EXAMPLE

A numerical example is examined to investigate effectiveness of the VBSP. In the numerical example, the fuzzy reasoning possess two input variables, x_1 and x_2, and one output variable y. These two input variables vary in a semi-open interval, $x_1, x_2 \in [0, 1)$. An input domain D_0 is divided into two domains, D_1 and D_2, to define an elite and two semi-elites. In these domains, the individuals are performed by an squared error given by the following equation:

$$e_k(g) = \sum_{x_1, x_2 \in D_k} (y_o - y_f)^2 \Big/ |D_k| \tag{3}$$

where k denotes the number of those domains, g denotes an individual to be performed, $|D_k|$ denotes the number of data to be performed, y_o denotes the objective result and y_f denotes the fuzzy reasoning result. The elite, E, has contents of an individual which gives the smallest value of the squared error, e_0, in the domain D_0 through the optimization. The semi-elites, S_1 and S_2, have contents of individuals which give the smallest value of the functions, e_1 and e_2, in the domains D_1 and D_2, at the current generation respectively.

A population in this numerical example consists of 18 individuals. Each individual in genotype is coded by the same manner shown in the previous section.

The following three functions are adopted to be approximated by the fuzzy reasoning model.

$$y = f_1(x_1, x_2) = 0.5x_1 + 1.2x_2, \tag{4}$$

$$y = f_2(x_1, x_2) = sin(3\pi x_1) - (x_2 - 0.5)^2, \tag{5}$$

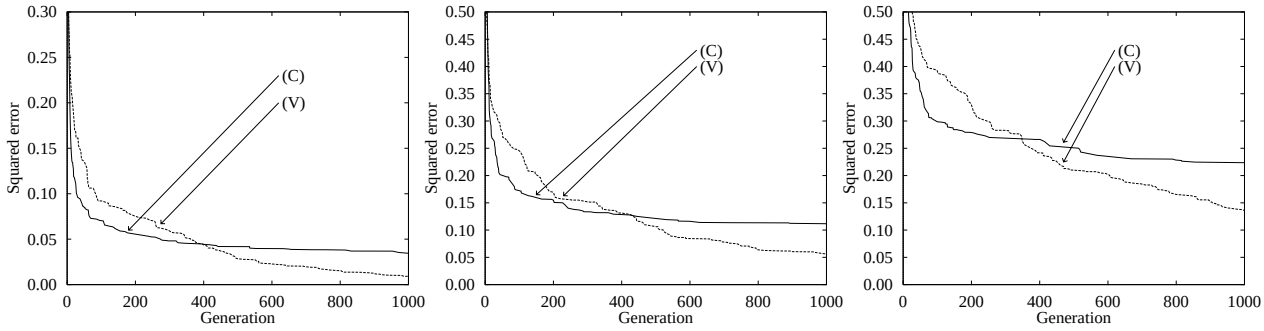
and

$$y = f_3(x_1, x_2) = \gamma_1(x_1) + \gamma_2(x_2) \tag{6}$$

where

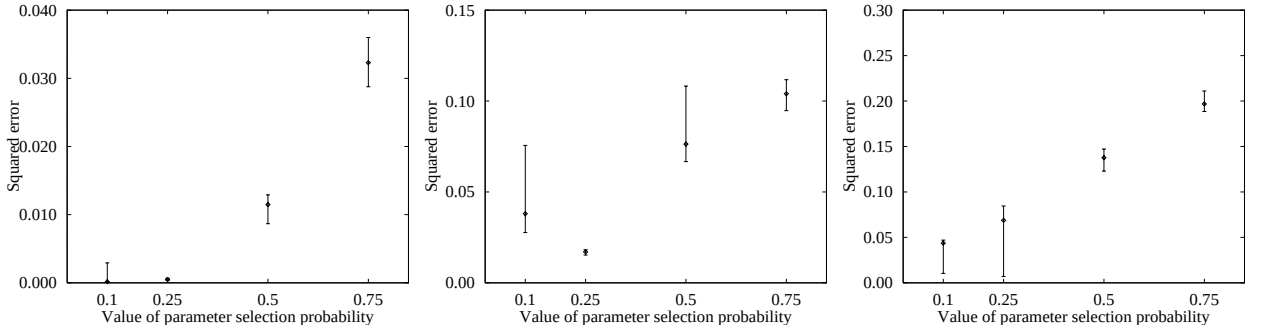
$$\gamma_1(x_1) = \begin{cases} -4x_1 + 1 & (0 \le x_1 < 0.5), \\ 0 & (0.5 \le x_1 < 0.75), \\ -4(x_1 - 0.75) + 1 & (0.75 \le x_1 < 1), \end{cases} \quad \gamma_2(x_2) = \begin{cases} -1 & (0 \le x_2 < 0.5), \\ 1 & (0.5 \le x_2 < 0.75), \\ -8(x_2 - 0.75) + 1 & (0.75 \le x_2 < 1). \end{cases}$$

Figures 6 (a)–(c) illustrate optimization progresses in the examples of function approximation. In order to consider that the GA behaves as a stochastic model, ten times of the optimizations under same condition have been executed. Each curve in Fig.6 shows average squared error of ten times of the optimization. In these optimizations, the values of PSP and BSP are set to 0.25 and 0.25 respectively, where the combination of these values gives best result for the case using the conventional fixed BSP. In Fig.6, (F) and (V) denote progresses



(a)results for Eq.(4) (b)results for Eq.(5) (c)results for Eq.(6)

Figure 6: Optimization progresses for each function approximation.



(a)results for Eq.(4) (b)results for Eq.(5) (c)results for Eq.(6)

Figure 7: Averaged values and standard deviations of the squared errors given after the optimization.

in the case using the VBSP and the conventional fixed BSP respectively. These results shows that the VBSP is more effective to quickly obtain good result than the convention fixed BSP.

On the other hand, we have investigated how the optimization behaves under different values of the PSP as shown in Figs.7. Fig.7 shows that the results of the optimization greatly depend on the value of the PSP. This means that a designer requires to decide the value of the PSP by trial and error in a designing an optimization algorithm even using the VBSP. Besides, there is not sufficient evidence that the value of the PSP obtained by the trial and error always allows the GA to quickly converge to a good result.

5. ADAPTIVE VARIABLE PARAMETER SELECTION PROBABILITY

To improve the problem shown in the previous section, a method to adaptively change value of the PSP depending on optimization progress condition or the APSP is proposed in this section. Procedures for the GA with the APSP is shown in Fig.8 and listed as follows:

(1) Define the initial value of the PSP.

(2) Let α_i change in the squared error while the i-th interval where the interval consists of T generations.

(3) If a condition $\alpha_i < \beta\alpha_{i-1}$ is true, then decrease the value of the PSP by multiplication of θ and previous value of Pp. Now, we call β and θ a control coefficient and a decrement coefficient.

(4) If the value of PSP becomes lower than the threshold Pp_{th}, set the value of the PSP 0.25.

(5) Repeat the procedures (2), (3) and (4) until final generation.

Figure 9 shows comparisons among the cases using the conventional fixed BSP, the VBSP and the APSP. In this example, configuring parameters of the APSP, β, θ, Pp_{ini} and Pp_{th} are 0.8, 0.8, 0.005 and 0.25 respectively. As shown in Fig.9, the cases using the APSP for various function approximation are advantage to quickly obtain good solutions. This is due to that PSP has been adaptively varied to suitable value through the optimization, and then the optimization has been executed under good condition.

```
procedure New_GA
var
  q, i : integer ; Pp : real ;
const
  Pb = P_2 ;
begin
  q := 0 ; Pp := Pp_ini ; i := 0 ;
  while q < q_f do begin
    Get performance of each individual g_k ;
    Select parents e, s_1, s_2, ... ;
    Apply crossover operator ;
    if (q%T = 0) and (q > 1) then begin
      ¿_i := |E(e_q-T, D_all) - E(e_q, D_all)| ;
      if ¿_i < ¿_i-1 then Pp := ˘Pp;
      if Pp < Pp_th then Pp := 0.25;
      i := i+1
    end;
    Apply mutation operator ;
    q := q+1
  end
end.
```

Figure 8: Procedure of the GA with the APSP.

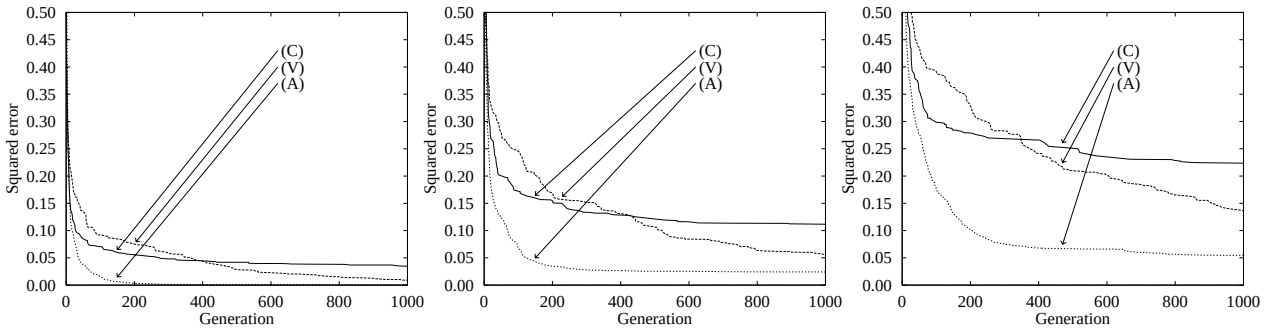
Figures 10–13 show averaged values and the standard deviations of the squared error after the optimization for various values of each parameter to configure the APSP. These results show that those parameters to configure the APSP can be decided in wide efficient ranges.

6. CONCLUSION

In this paper, we have proposed the APSP for the fuzzy modeling using the GA. In this method, the value of the PSP is adaptively varied depending on the optimization progress condition. Comparing efficiency of the APSP with the conventional GA and the VBSP proposed in the previous paper[6], effectiveness of the APSP is represented. Besides, Each parameters configuring the APSP can be defined in wide effect range in value and then can be decided without trial and error. This shows that the algorithm using the APSP is useful to apply to any optimization problems.

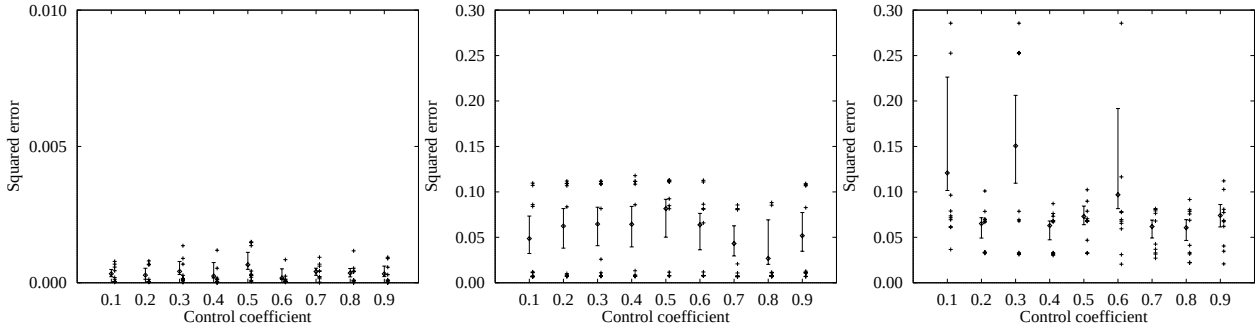
REFERENCES

1. SUGENO M. 1988, Fuzzy Control, Nikkan Kogyo Shinbunsha (in Japanese).
2. GOLDBERG D.E., Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, New York, 1989.
3. SAKAWA M. & TANAKA M., Genetic Algorithm, Asakura-Shoten, Tokyo, 1995 (in Japanese).
4. OHKI M., MORIYAMA T., MIYATA H. & OHKITA M., Self-Tuning of Fuzzy Reasoning with Genetic Algorithm, Proc. of Second World Congress of Nonlinear Analysts (WCNA'96) (Athens, Greece, July 10-17, 1996), part 8, pp.5291-5301, 1997.
5. OHKI M., SHIKATA T, MORIYAMA T. & OHKITA M., A Genetic Algorithm with Modified Bit-Selection Probability for Optimizing the Fuzzy Reasoning, Proc. of 5th European Congress on Intelligent Techniques and Soft Computing, (EUFIT'97), (Aachen, Germany Sept. 8-11, 1997), vol.1, pp.694-698, 1997.
6. OHKI M., MORIYAMA T. & OHKITA M., Optimization of Fuzzy Reasoning by Using Genetic Algorithm with Variable Bit-Selection Probability in its Mutation Operator, Trans. of IEE Japan, vol.119-C, no.7/8, pp.1196-1202, 1998 (in Japanese).
7. OHKI M., HASEGAWA S, MORIYAMA T. & OHKITA M., Genetic Algorithm with Cyclic Variable Bit-Selection Probability Varying Depending on Generation, Proc. of 6th European Congress on Intelligent Techniques and Soft Computing, (EUFIT'98), (Aachen, Germany Sept. 7-10, 1998), vol.1, pp.477-481, 1998.



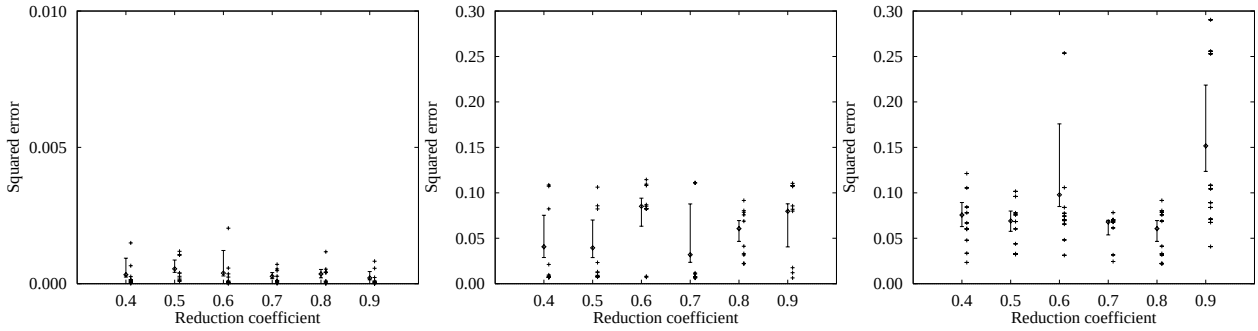
(a)results for Eq.(4) (b)results for Eq.(5) (c)results for Eq.(6)

Figure 9: Comparison of optimization progresses among the conventional fixed BSP, the VBSP and the APSP.



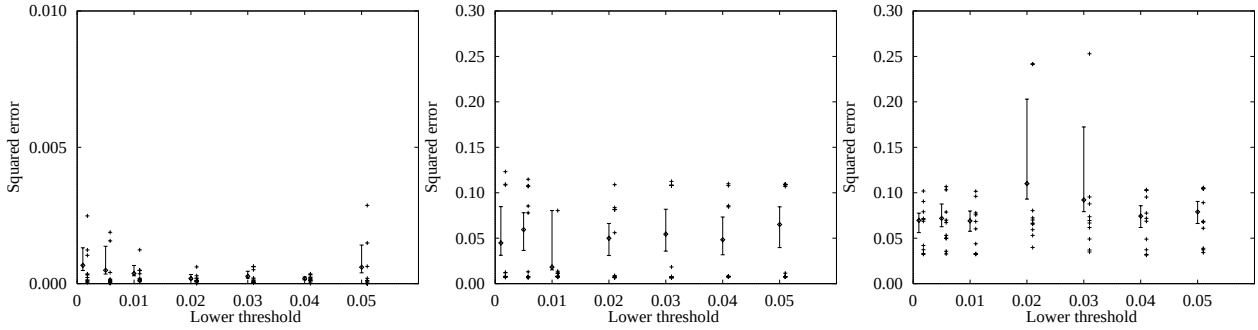
(a)results for Eq.(4) (b)results for Eq.(5) (c)results for Eq.(6)

Figure 10: Comparison of optimization results for various values of the control coefficient β.



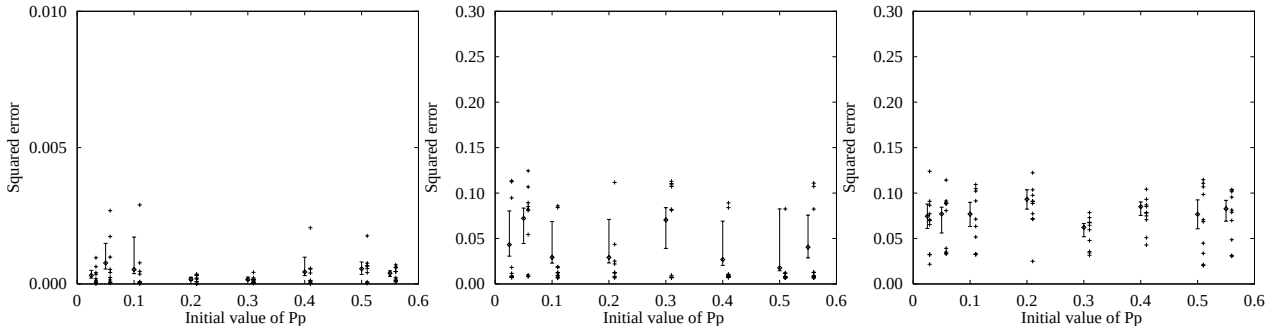
(a)results for Eq.(4) (b)results for Eq.(5) (c)results for Eq.(6)

Figure 11: Comparison of optimization results for various values of the decrement coefficient θ.



(a)results for Eq.(4) (b)results for Eq.(5) (c)results for Eq.(6)

Figure 12: Comparison of optimization results for various lower threshold values of the PSP, Pp_{th}.



(a)results for Eq.(4) (b)results for Eq.(5) (c)results for Eq.(6)

Figure 13: Comparison of optimization results for various initial values of the PSP, Pp_{ini}.