Chapter 3

CGSA for Constrained ED Problem

In this chapter, a combined genetic and simulated annealing algorithm (CGSA) for constrained ED problem is described. Other algorithms are developed to solve the constrained ED problem with monotonically and non-monotonically IC functions for comparisons as follows:

- Zoom brute force (ZBF)
- Zoom dynamic programming (ZDP)
- Genetic algorithm based on simulated annealing solution (GA-SA)
- Local search (LS)
- Merit order loading (MOL)

In Section 3.1, the CGSA procedures and details of each process are proposed. Section 3.2 describes the ZBF including the zoom feature. The ZDP with the numerical example are explained in Section 3.3. Section 3.4 describes GA-SA whereas Section 3.5 describes LS method. Lastly, MOL is explained in Section 3.6.

3.1 Combined genetic and simulated annealing algorithm (CGSA)

The combined genetic and simulated annealing algorithm (CGSA) combines the strength of GA and SA together. CGSA initializes one feasible individual and randomly generates the rest of *NP*-1 individuals in the initial population. At each generation (iteration), CGSA performs fitness function evaluation, tournament selection, uniform crossover, mutation, and elitism processes. For every epoch generations, the current best feasible individual is selected as an initial feasible solution for fine-tuning by SA.

To obtain a higher solution quality than GA-SA proposed in [8], CGSA vastly improves over GA-SA as follows. First, CGSA uses SA as an advanced local search algorithm to search for a better solution at every specified epoch GA generation and at the last GA generation whereas GA-SA simply uses SA to provide a good initial feasible solution for GA process. Furthermore, CGSA uses tournament selection and uniform crossover with sub-elitism whereas GA-SA uses basic roulette wheel selection and uniform crossover. The tournament selection can increase the diversity of individual population in the mating pool and the sub-elitism guarantees that the offspring individuals cannot be worse than their parent individuals.

3.1.1 Initialization

To ensure that there is at least one feasible solution in the CGSA population, the initial power output of N-1 generating units are calculated from

$$P_i(t) = \frac{P_{i,high}(t) \times P_D(t)}{\sum_{j=1}^{N} P_{j,high}(t)}, \quad i = 1,...,R-1,R+1,...,N. \tag{3.1.1}$$

P_R is calculated from quadratic as follows [6]:

$$A \cdot P_R^2(t) + (B-1) \cdot P_R(t) + C + P_D(t) - \sum_{\substack{i=1 \\ i \neq R}}^{N} P_i(t) = 0. \tag{3.1.2}$$

Where,

$A = B_{RR}$,

$$B = \sum_{\substack{j=1 \\ j \neq R}}^{N} B_{Rj} P_j(t) + \sum_{\substack{i=1 \\ i \neq R}}^{N} P_i(t) B_{iR} + B_{R0},$$

$$C = \sum_{\substack{i=1, \\ i \neq R}}^{N} \sum_{\substack{j=1, \\ j \neq R}}^{N} P_i(t) B_{ij} P_j(t) + \sum_{\substack{i=1, \\ i \neq R}}^{N} B_{io} P_i(t) + B_{00}.$$

The R^{th} dependence reference unit is varied from 1 to N. The feasible power outputs with the least total generator fuel cost is encoded into a binary-string CGSA initial individual while the rest NP-1 individuals are randomly generated.

3.1.2 Solution coding

Each generating unit power output of N-1 free units is encoded in a binary string normalized over its operating range [7]. The concatenated binary coding method stacks each normalized string of unit power output in series with each other to construct the string individual as shown in Figure 3.1. In this paper, each unit string is assigned by 16 bits; thus a string individual has $16 \times (N\text{-}1)$ bits.

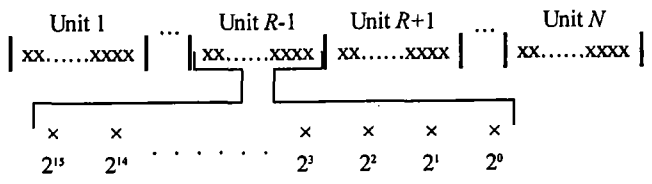

Figure 3.1 $16 \times (N\text{-}1)$ bit concatenated encoding scheme

To obtain the actual power output of each generating unit for fitness function evaluation, each string is decoded to the decimal value by,

$$P_i(t) = P_{i,low}(t) + D_i \times \left[\frac{P_{i,high}(t) - P_{i,low}(t)}{2^{16} - 1} \right], \; i = 1, \ldots R\text{-}1, R\text{+}1, \ldots N. \tag{3.1.3}$$

3.1.3 Fitness function evaluation

Since not all *NP* individuals are feasible, the fitness function of the j^{th} individual at time period t accounting for the cost function and the power balance equation is

$$f_j(t) = \frac{1}{2Cost_j(t)} + \frac{1}{2Pow_j(t)}, j = 1, \ldots, NP. \tag{3.1.4}$$

Where,

$$Cost_j(t) = 1 + \frac{(\sum_{i=1}^{N} C_i(P_i^j(t)) - Cost_{\min})^2}{Cost_{\max} - Cost_{\min}},$$

$$Pow_j(t) = 1 + (\sum_{i=1}^{N} P_i^j(t) - P_D(t) - P_L(t))^2 / P_D(t),$$

After decoding the j^{th} individual to $[P_1^j(t), \ldots, P_{R-1}^j(t), P_{R+1}^j(t), \ldots, P_N^j(t)]$ and substituting in Equation (3.1.4), if $P_R^j(t) < P_{R,low}(t)$, set $P_R^j(t) = P_{R,low}(t)$ and if $P_R^j(t) > P_{R,high}(t)$, set $P_R^j(t) = P_{R,high}(t)$. $[P_1^j(t), \ldots, P_N^j(t)]$ is used to evaluate the fitness value in Equation (3.1.4). Even though the power balance may not be satisfied, the fitness value is ensured to be in the range of 0 to1. Note that the fitness value of the feasible solution will not be less than 0.5.

3.1.4 GA operators

The CGSA operators consist of GA and SA operators. GA is a search algorithm based on the mechanics of natural selection and natural genetics [10]. GA is widely used for searching a global optimum solution. For CGSA, GA operators are the tournament selection, uniform crossover with sub-elitism, mutation and elitism.

a) Tournament selection

In the selection process, the tournament selection is used instead of the roulette wheel selection to increase the diversity of individual population in the mating pool [11]. A binary tournament size consisting of two individuals is employed. Initially, *NP*/2 groups of individuals are randomly selected from the total *NP* individuals without replacement. These groups participate in the tournament in which the winning individual of each

tournament, having the higher fitness value than the other, will be replaced in the parent individual. The process repeats twice to obtain the total *NP* parent individuals in the mating pool. Figure 3.2 shows the tournament selection.

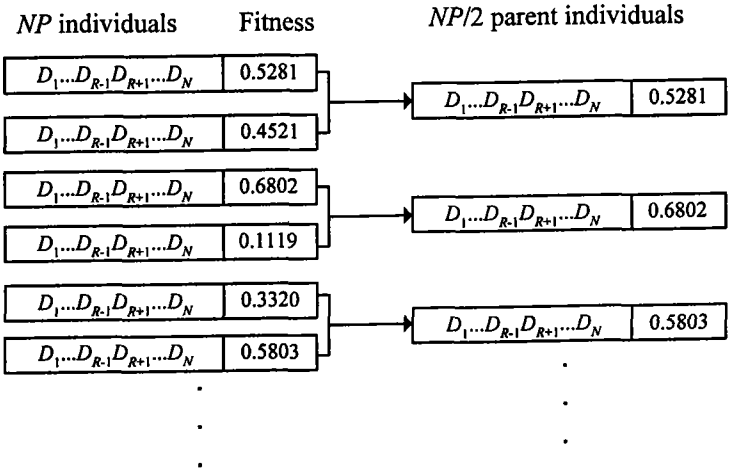

Figure 3.2 Tournament selection

b) Uniform crossover with sub-elitism

Uniform crossover is a process of exchanging bits between two parent string individuals in the mating pool [7]. More specifically, arbitrary positions on both individuals are chosen for crossing locations, where the exchanges of bits take place. The uniform crossover is shown in Figure 3.3. A $16\times(N\text{-}1)$ bit randomly generated crossing mask is employed to determine the crossing locations. To reproduce two offspring individuals, two parent individuals will exchange their bits at every location, where the corresponding positions in the crossing mask string are one. To guarantee that the offspring individuals cannot be worse than their parent individuals, those parent and offspring individuals are sorted in the descending order of their fitness values. The fittest and the second fittest individuals replace the two offspring individuals (sub-elitism).

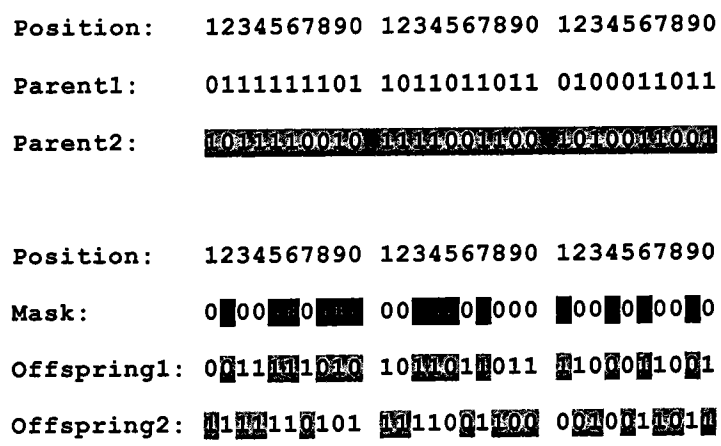

Figure 3.3 Uniform crossover

c) Mutation

Mutation is a process of flipping bits in a randomly chosen offspring at the random positions after performing crossover [10]. The $16\times(N\text{-}1)$ bit string mask representing the bit positions to be flipped is generated according to the probability of mutation. The rate of mutation is much lower than that of crossover since the high mutation rate may deteriorate the offspring individuals. Figure 3.4 shows the mutation process.

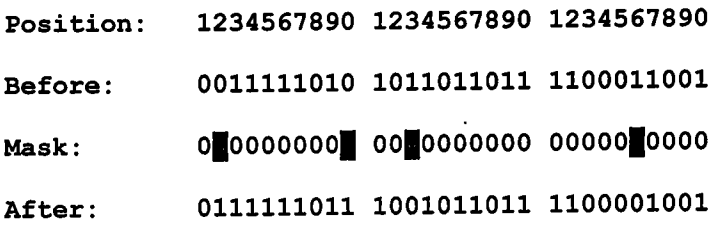

Figure 3.4 Mutation

d) Elitism

The best feasible individual after mutation may not be better than the best feasible individual before mutation. Elitism is used to guarantee that the best feasible string individual survives until the last generation. More specifically, if the best offspring individual at the G^{th} iteration is worse than the best feasible parent individual, the best feasible parent individual will randomly replace one of the offspring individuals. That is, the offspring individuals in the G^{th} iteration become the parent individuals in the $(G+1)^{th}$ iteration. With the elitism, the CGSA guarantees to produce the best feasible solution that

is not worse than the initial feasible solution. The current best feasible solution will be subsequently fine-tuned by SA for every epoch generations.

3.1.5 SA operators

The concept of the simulated annealing method is based on the physical annealing process of solids [12]. SA became popular about a decade ago in solving various optimization problems because the characteristic of reducing the chance of getting stuck in a poor local optimum. For CGSA, at every epoch generations, the best feasible individual is encoded to be SA initial solution. The SA process is used to find a local optimum solution. When SA is terminated, the SA solution is encoded back into the CGSA individual. The SA processes are:

a) Perturbation

At the k^{th} iteration and m^{th} trial , randomly select P_R be a dependent reference power output and let the power outputs of the remaining N-1 units be a vector $\boldsymbol{P}^{(k,m)}$. Subsequently, the vector of perturbation $N^{(k,m)}$ consists of uniform random variables which are generated within the upper bound of $+\sigma_k$ and lower bound of $-\sigma_k$, where σ_k is the control parameter at the k^{th} iteration, which is calculated from

$$\sigma_k = 0.95^{(k-1)} * \sigma_1. \tag{3.1.5}$$

The upper and lower bounds are equal to $Min\{(\boldsymbol{P}^{(k,m)} + \sigma_k), P_{i,high}(t)\}$ and $Max\{(\boldsymbol{P}^{(k,m)} - \sigma_k), P_{i,low}(t)\}$, respectively. A new loading vector $\boldsymbol{P}'$ of N-1 generating units in the neighborhood is $\boldsymbol{P}^{(k,m)}+N^{(k,m)}$ while P_R' is calculated directly from the quadratic Equation (3.1.2). If P_R' satisfies its operating limit constraint in Equation (2.7), $\boldsymbol{P}'$ and P_R' are accepted as a feasible candidate solution. Otherwise, discard $\boldsymbol{P}'$ and P_R' and repeat randomly selecting P_R and adding a new randomly generated $N^{(k,m)}$ vector to $\boldsymbol{P}^{(k,m)}$ until P_R' is feasible. Figure 3.5 shows the addition of the loading vector $\boldsymbol{P}^{(k,m)}$ and $\boldsymbol{N}^{(k,m)}$.

$$\begin{bmatrix} P_1(t) \\ \vdots \\ P_{R-1}(t) \\ P_{R+1}(t) \\ \vdots \\ P_N(t) \end{bmatrix} + \begin{bmatrix} N_1 \\ \vdots \\ N_{R-1} \\ N_{R+1} \\ \vdots \\ N_N \end{bmatrix} = \begin{bmatrix} P_1'(t) \\ \vdots \\ P_{R-1}'(t) \\ P_{R+1}'(t) \\ \vdots \\ P_N'(t) \end{bmatrix}$$

Figure 3.5 Perturbation

b) Acceptance criterion

The probabilistic acceptance criterion for SA is given as shown below [4]:

$$p(\Delta F_t) = \frac{1}{1 + e^{(\Delta F_t / \sigma_k)}}. \tag{3.1.6}$$

The new loading vector P' is accepted as a current solution if either

1. $\Delta F_t \leq 0$ or

2. $\Delta F_t > 0$ and $p(\Delta F_t) >$ rand (0,1).

Otherwise, discard $\boldsymbol{P}'$ and retain $\boldsymbol{P}^{(k,m)}$ and P_R.

The best solution from the total number of trials of the current iteration will be used as an initial solution for the next iteration. The iterative process is terminated when σ_k is less than 1. Subsequently, the SA solution replaces the best individual in GA population.

3.1.6 Procedures of CGSA

Step 1: Read the unit operating limits, input-output coefficients, ramp rate limits, and fuel cost of each unit, a forecast load demand at t =1, transmission loss B-matrices, and initial power outputs $P_i(0) = P_{i,min}$.

Step 2: Specify the population size (NP), maximum GA generation limit (G_{max}), uniform crossover probability, mutation probability, epoch generation, maximum SA trial limit (m_{max}), and initial value of control parameter (σ_1).

Step 3: Initialize the time period t to one.

Step 4: Determine $P_{i,high}(t)$ and $P_{i,low}(t)$.

Step 5: Calculate an initial feasible solution from Equation (3.1.1) and encode into a binary string initial individual whereas the rest of NP -1 individuals are randomly generated.

Step 6: Initialize the GA generation counter (G) to one.

Step 7: Perform the following GA process:

Step 7.1: Evaluate the fitness values of all *NP* individuals.

Step 7.2: Perform the tournament selection.

Step 7.3: Perform the uniform crossover to reproduce *NP* offspring individuals.

Step 7.4: Perform the mutation.

Step 7.5: Perform the elitism.

Step 8: If the division of G by the specified epoch value is not an integer number and $G < G_{max}$, go to Step 13.

Step 9: Select the fittest feasible offspring individual as an initial solution, $\boldsymbol{P}^{(1,1)}$ and P_R.

Step 10: Set $\boldsymbol{P}_{best} = \boldsymbol{P}^{(1,1)}$ and $P_{R,best} = P_R$, and initialize the SA iteration counter (k) to 1.

Step 11: Perform the following SA process:

Step 11.1: Calculate σ_k.

Step 11.2: Initialize the SA trial counter (m) to 1.

Step 11.3: Randomly selects the R^{th} reference unit. Obtain $\boldsymbol{P}'$ by adding a randomly generated $N^{(k,m)}$ within its limits to $\boldsymbol{P}^{(k,m)}$. If the calculated $P_R^{'}$ is not feasible, discard $\boldsymbol{P}'$ and $P_R^{'}$ and repeat this Step until $P_R^{'}$ is feasible.

Step 11.4: If $\Delta F_t \leq 0$, set $\boldsymbol{P}^{(k,m+1)} = \boldsymbol{P}'$ and $P_R^{(k,m+1)} = P_R^{'}$, update the current best solution by letting $\boldsymbol{P}_{best} = \boldsymbol{P}'$ and $P_{R,best} = P_R^{'}$, and go to *Step* 11.6.

Step 11.5: If $p(\Delta F_t) >$ random(0,1), set $\boldsymbol{P}^{(k,m+1)} = \boldsymbol{P}'$ and $P_R^{(k,m+1)} = P_R^{'}$. Otherwise, set $\boldsymbol{P}^{(k,m+1)} = \boldsymbol{P}^{(k,m)}$ and $P_R^{(k,m+1)} = P_R^{(k,m)}$.

Step 11.6: If $m < m_{max}$, let $m = m+1$ and return to *Step* 11.3. Otherwise, set $\boldsymbol{P}^{(k+1,1)} = \boldsymbol{P}_{best}$, and $P_R^{(k+1,1)} = P_{R,best}$.

Step 11.7: If $\sigma_k > 1$, let $k = k+1$ and return to *Step* 11.1.

Step 12: If $G < G_{max}$, encode $\boldsymbol{P}_{best}$ and $P_{R,best}$ to a binary string individual and replace the best individual in *NP* population. Otherwise, go to Step 14.

Step 13: Set $G = G+1$ and return to Step 7.

Step 14: $\boldsymbol{P}_{best}$ and $P_{R,best}$ are the solution at time period t. If $t < T$, let $t = t+1$, read the power load demand at time period t, and return to Step 4. Otherwise, CGSA is terminated.

3.2 Zoom brute force (ZBF)

The zoom brute force (ZBF) method is developed to determine the optimal solution for benchmarking purpose. ZBF searches for the optimal solution by examining every possible combination of discretized solutions. At the first iteration, the step size is Δ_1. At the subsequent iterations, the searching step size of the k^{th} iteration, Δ_k, is calculated from

$$\Delta_k = \Delta_{k-1}/K_\Delta. \tag{3.2.1}$$

The zoom feature is applied to the BF method to reduce the computing time. The example of the zoom feature is given in Figure 3.6. Let the load demand be 600 MW, the horizontal axis is P_1 and the vertical axis is P_2. The square mesh represents the boundary of searching area whereas the black dot represents the candidate solution (P_1, P_2). Initially, P_1 = 300, P_2 = 300, and Δ_1 = 100 MW. For the first iteration, the best solution is (400,200). For the second iteration, the value of Δ_2 is equal to 25 MW since K_Δ = 4, and the searching boundary of each generating unit is reduced; however, the matrix dimension remains the same. The i^{th} matrix dimension is calculated from

$$MD_i = (P_{i,max} - P_{i,min})/\Delta_1. \tag{3.2.2}$$

With the fixed matrix dimension, the upper and lower bounds of unit i^{th} at iteration k are calculated from

$$UB_i^k = Min\{P_{i,best} + (0.5 \times MD_i \times \Delta_k), P_{i,high}(t)\} \tag{3.2.3a}$$

$$LB_i^k = Max\{P_{i,best} - (0.5 \times MD_i \times \Delta_k), P_{i,low}(t)\} \tag{3.2.3b}$$

Note that UB_i^k and LB_i^k of the first iteration are set to $P_{i,high}(t)$ and $P_{i,low}(t)$, respectively. It is assumed that the ramp rate limit constraints are satisfied, the upper and lower bounds of P_1 in the second iteration are 450 MW and 350 MW whereas those of P_2 are 250 and 150 MW, respectively. Table 3.1 shows the best solution, searching boundaries of each unit, and the searching step sizes of P_1 and P_2 for the first two iterations.

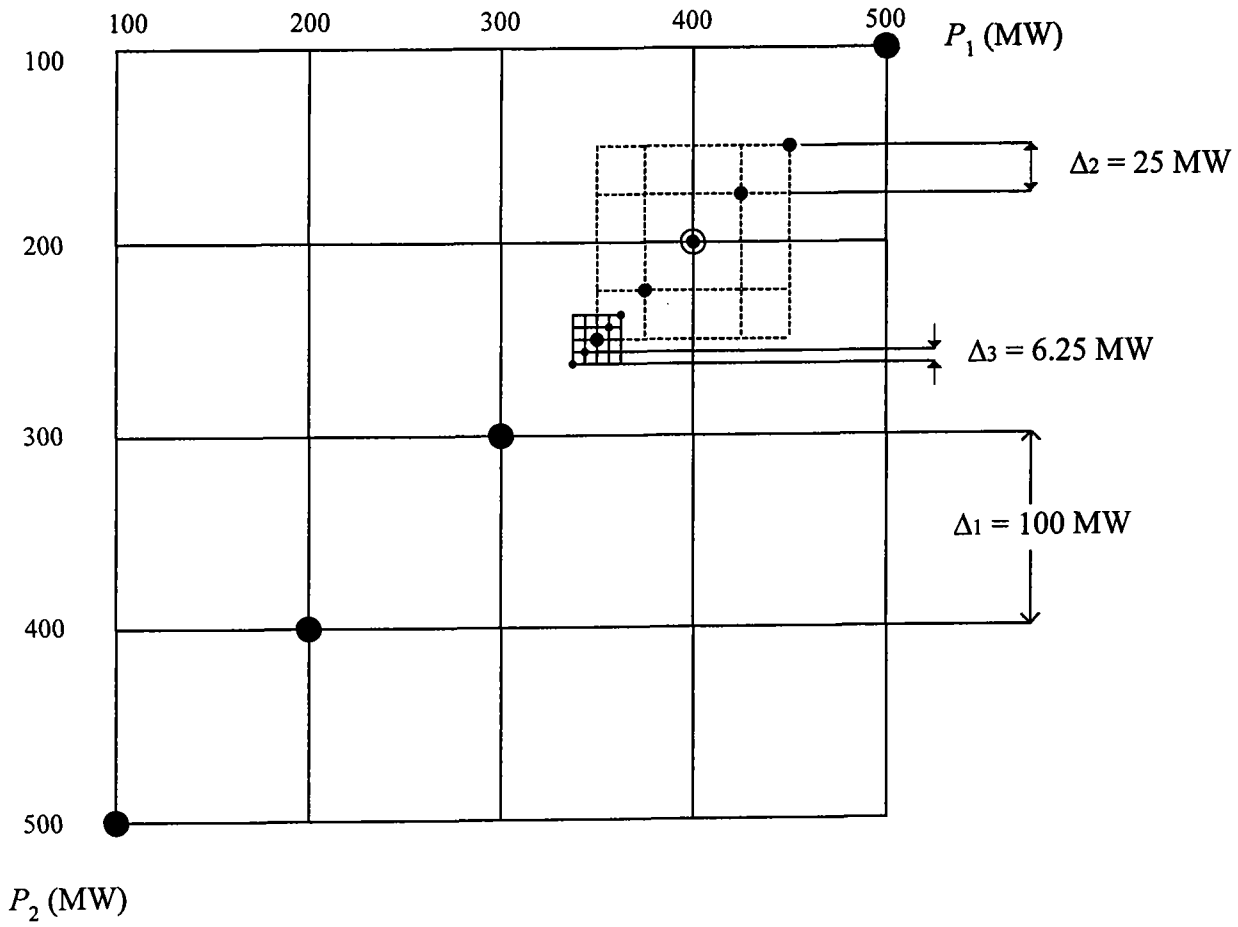

Figure 3.6 Zoom feature

Table 3.1 The best solution, searching boundaries and searching step sizes of P_1 and P_2 for the first two iterations.

Iteration number (k)	Upper bound (MW)		Lower bound (MW)		Step size (Δ_k)	Best solution of the k^{th} iteration (MW)	
	UB_1^k	UB_2^k	LB_1^k	LB_2^k		P_1	P_2
1	500	500	100	100	100	400	200
2	450	250	350	150	25	350	250

For constrained ED problem, the flowchart of ZBF is shown in Figure 3.7. Initially, ZBF reads the input data and calculates MD_i from Equation (3.2.2). At each time period, ZBF determines $P_{i,high}(t)$ and $P_{i,low}(t)$ and read the value of P_D. At each iteration, the R^{th} dependent reference unit is varied from 1 to N. Δ_k, UB_i^k, and LB_i^k are calculated from Equations (3.2.1) and (3.2.3), respectively. To generate a candidate solution, ZBF increases the i^{th} power output, P_i, with the increment of Δ_k whereas the rest power output are not changed. If P_i reaches UB_i^k, ZBF increases P_{i+1} from LB_{i+1}^k to UB_{i+1}^k. For each move, P_R is calculated from Equation (3.1.2).

If P_R is feasible, P_L is calculated from Equations (2.4) and the generator fuel cost is calculated. If P_R is not feasible, it is discarded. Otherwise, if the cost is less than the best cost reached, the best solution reached is updated.

After all possible combinations are searched; the best solution reached is used to set UB_i^k and LB_i^k of the next iteration. ZBF terminates at the last iteration and let the best solution reached be the solution at that time period.

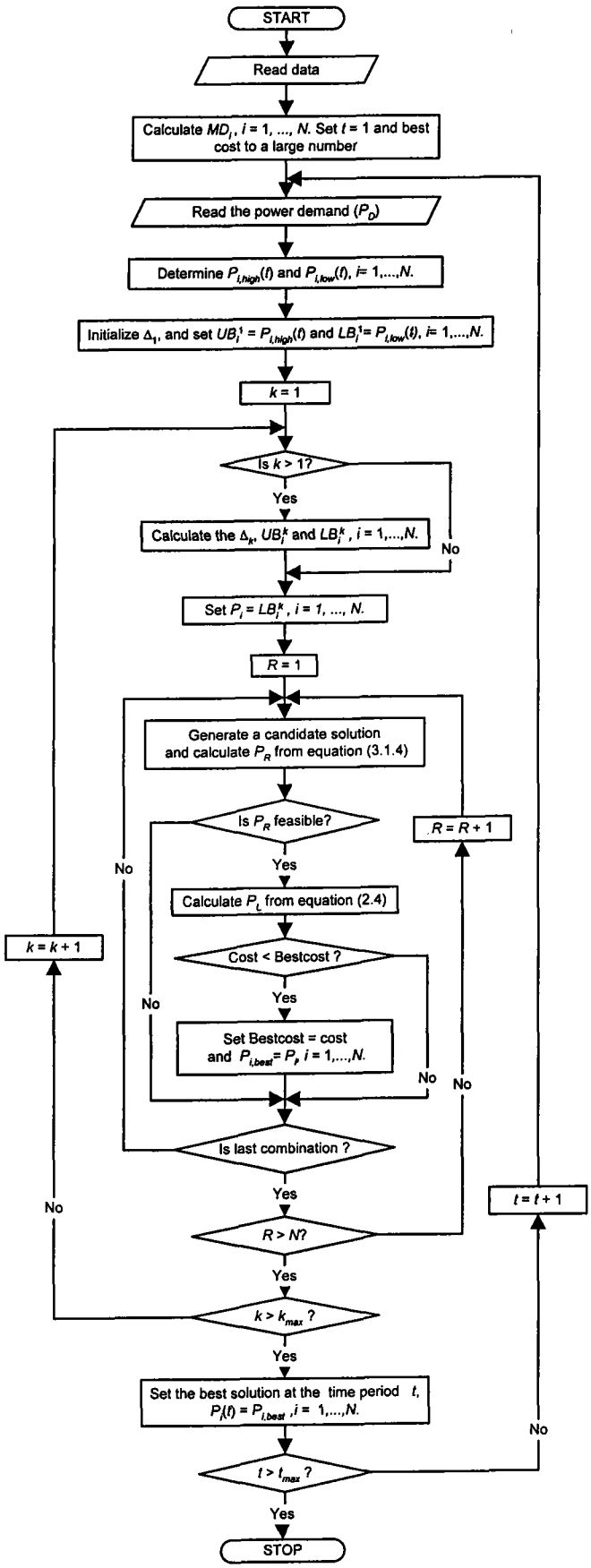

Figure 3.7 Flow chart of ZBF method for constrained ED problem

3.3 Zoom dynamic programming (ZDP)

ZDP is a traditional method used to solve optimization problems. Initially, MD_i is calculated from Equation (3.2.2). For each time period, $P_{i,high}(t)$ or $P_{i,low}(t)$ are determined. P_{GD}^1 is initially set to the power demand and the best cost is set to a very large number. Similar to ZBF, at iteration k, Δ_k, UB_i^k, and LB_i^k are calculated from Equations (3.2.1) and (3.2.3), respectively. P_N is varied from LB_N^k to UB_N^k with the increasing step of Δ_k. For each value of P_N, the DP recursive function obtains a candidate solution as shown in Equations (3.3.1) – (3.3.3).

$$DP_N(P_{GD}^k) = \min \{DP_{N-1}(P_{GD}^k - P_N) + C_N^*(P_N)\}. \quad (3.3.1)$$

$$DP_{N-1}(P_{GD}^k - P_N) = \min \{DP_{N-2}(P_{GD}^k - P_N - P_{N-1}) + C_{N-1}^*(P_{N-1})\} \quad (3.3.2)$$

$$DP_2(P_{GD}^k - P_N - \ldots - P_3) = \min \{C_2^*(P_2) + C_1^*(P_1)\}. \quad (3.3.3)$$

where

$$C_i^*(P_i) = PF_i^k \times C_i(P_i). \quad (3.3.4)$$

$$PF_i^k = 1/(1 - \partial P_L^k / \partial P_i). \quad (3.3.5)$$

If the cost of a candidate solution is less than the best cost reached, it replaces the best cost reached. The best solution reached is used to calculate P_L^{k+1}, PF_i^{k+1} and P_{GD}^{k+1} by Equations (2.4), (3.3.5) and (3.3.6), respectively.

$$P_{GD}^{k+1} = P_D + P_L^{k+1} \quad (3.3.6)$$

The best solution reached at the current iteration is used to set UB_i^k, and LB_i^k for the next iteration. At the last iteration, the best solution is the ZDP solution at that time period.

Note that PF_i^k are neglected at the first iteration therefore Δ_2 is set to Δ_1. With PF_i^1 equals to one, the obtained solution neglects the losses. Subsequently, the deviation of P_L^{k+1} from the power balance constraints is improved during the iterative process. Finally, at the last iteration, the best cost reached is calculated without the penalty factor. The flow chart of ZDP for constrained ED is shown in Figure 3.8.

Example 1: Minimize $\{C_1(P_1) + C_2(P_2) + C_3(P_3)\}$ subject to $P_1 + P_2 + P_3 = 1000$ MW.

The $DP_3(1000)$ function is the minimum cost of three generating units for 1000 MW load demand. The function $DP_3(1000)$ equals $DP_2(1000- P_3) + C_3(P_3)$. The objective is to minimize $DP_3(1000)$ with respect to P_3. With P_3 initially fixed at LB_3^k, the minimum cost of $DP_2(1000-P_3)$ is obtained by varying P_2 such that $C_1(P_1) + C_2(P_2)$ is minimized whereas each value of P_2 generates a candidate solution. Note that P_1 is $1000-P_3- P_2$. By the characteristic of a recursive function, the function $DP_2(1000-P_3)$ returns its minimum cost, $\min\{DP_2(1000-P_3)\}$. If the cost of DP_3 is cheaper than the best cost reached, the best solution reached is updated. Subsequently, P_3 is increased with the step of Δ_k until UB_3^k. Each value of P_3 generates one candidate solution. The best value of P_3 is obtained when DP_3 is minimized, $\min \{DP_2(1000-P_3)+ C_3(P_3)\}$.

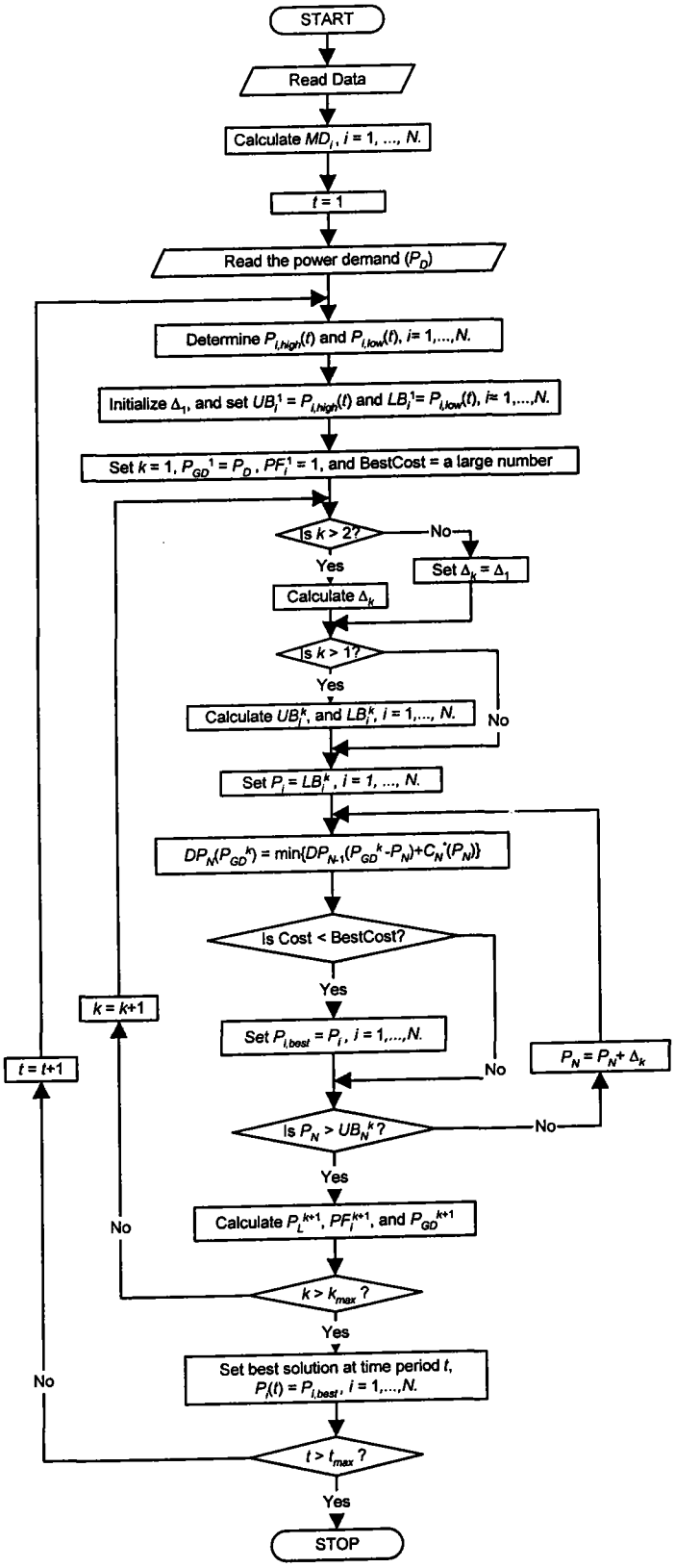

Figure 3.8 Flow chart of ZDP method for constrained ED problem

Example 2: Minimize $\{C_1(P_1) + C_2(P_2) + C_3(P_3)\}$ subject to $P_1 + P_2 + P_3 - P_L = 800$ MW.

Three generating unit system consisting of P_1, P_2 and P_3 are dispatched for the power demand of 800 MW. The operating limits, fuel cost, input-output coefficients are shown in Table 3.2.

Table 3.2 Input-ouput characteristics of three generating units

Unit	P_{min} (MW)	P_{Int} (MW)	P_{max} (MW)	Fuel cost (Baht/Gcal)	Input-output coefficients (Gcal/Hour)		
					A_i	B_i	C_i
P_1	100	-	300	273.80	-123.2693000	3.177074200	-0.002752183
P_2	320	510	-	315.143	180.4651584	1.459778070	0
	-	510~650	-		79.4196883	1.658094200	0
	-	650	700		-141.8048526	1.998281070	0
P_3	376	495		315.143	113.5361104	1.856133141	0
		495	678		206.6947723	1.667493427	0

The B matrix coeffiencts are:

$B_{00} = 0.00201785$, B_{i0}^T = [-0.003830 -0.000171 0.0009450], and

$$[B_{ij}] = \begin{bmatrix} 0.0033800 & 0.0004765 & -0.0002535 \\ 0.0004765 & 0.0026050 & 0.0004505 \\ -0.0002535 & 0.0004505 & 0.0014700 \end{bmatrix}$$

The results are shown in Table 3.3. At the first iteration, P_L^1 is neglected, P_{GD}^1 equals P_D. The solution obtained is used to calculate P_L^2 by substituting P_1, P_2 and P_3 of 104, 320 and 376 MW in Equation (2.4), respectively. As a result, P_L^2 is 6.4184 MW and P_{GD}^2 is 806.4184 MW. At the last iteration, the best power outputs reached are 100.0003, 330.6256 and 376 MW, respectively. The total generator fuel cost is 510,396.82 Baht whereas the power loss is 6.6259 MW.

Table 3.3 ZDP numerical results

Iteration (k)	1	2	3	4	5	6	7
Δ_k (MW)	5	5	1	0.2	0.04	0.008	0.0016
P_1(MW)	104	100.4184	100.6143	100.0151	100.0255	100.0015	100.0003
P_2(MW)	320	330	330	330.60	330.60	330.624	330.6256
P_3(MW)	376	376	376	376	376	376	376
P_L^k (MW)	0.00	6.4184	6.6143	6.6151	6.6255	6.6255	6.6259
ΣP_i	800	806.4184	806.6143	806.6151	806.6255	806.6255	806.6259
C_T(Baht/hr)	508373.11	518784.92	519054.00	518904.35	518918.00	518912.02	510396.82

3.4 Genetic algorithm based on simulated annealing solution (GA-SA)

GA-SA is the GA based on SA solution [8]. GA-SA first runs SA and uses SA solution as one feasible individual in the GA initial population whereas the rest *NP*-1 individuals are randomly generated. For a specified maximum number of generations, GA-SA performs selection, uniform crossover, mutation and elitism. The flowchart of GA-SA for constrained ED problem is shown in Figure 3.9. At each time period, GA-SA initializes the best power-loading vector $\boldsymbol{P}_{best}$ and $P_{R,best}$ with the least total generator fuel cost from Equation (3.1.1). The value of σ_k is calculated at the beginning of iteration by using Equation (3.1.5). In each trial, GA-SA select P_R and generate $N^{(k,m)}$ and $\boldsymbol{P}'$. If P_R' calculated from Equation (3.1.2) is satisfied its ramp rate constraint in Equation (2.7), and the cost of $\boldsymbol{P}'$ is less than the best cost reached, $\boldsymbol{P}'$ and its cost replaces $\boldsymbol{P}_{best}$ and the best cost reached, respectively. Otherwise, $\boldsymbol{P}'$ is checked for the probabilistic acceptance criterion in Section 3.1.5b. When the maximum number of trials is reached, the best solution reached is set to be an initial solution for the next iteration. SA is terminated when σ_k is less than 1 and its solution is encoded into a GA initial individual. The rest *NP*-1 initial individuals are generated by randomly flipping binary bits in the string individuals. GA-SA evaluates the fitness values of *NP* individuals before repeating the roulette wheel selection, uniform crossover, mutation, and elitism. When the maximum number of generations is reached, the fittest individual is selected to be the solution at that time period.

Selection of parent individuals to the mating pool is based on the biased roulette wheel selection [10]. The bigger the string individual fitness values the higher the probability to have copies of them in the mating pool. The reproduction probability and cumulative reproduction probability are given as follows:

$$P_{rep,j} = f_j / \sum_{i=1}^{NP} f_i, \tag{3.4.1}$$

$$CP_{rep,j} = \sum_{k=1}^{j} P_{rep,k}, j = 1,...,NP. \tag{3.4.2}$$

The selected parents string individuals to be copied to the mating pool are those having $CP_{rep,j}$ higher than the real number randomly generated between 0 and 1.

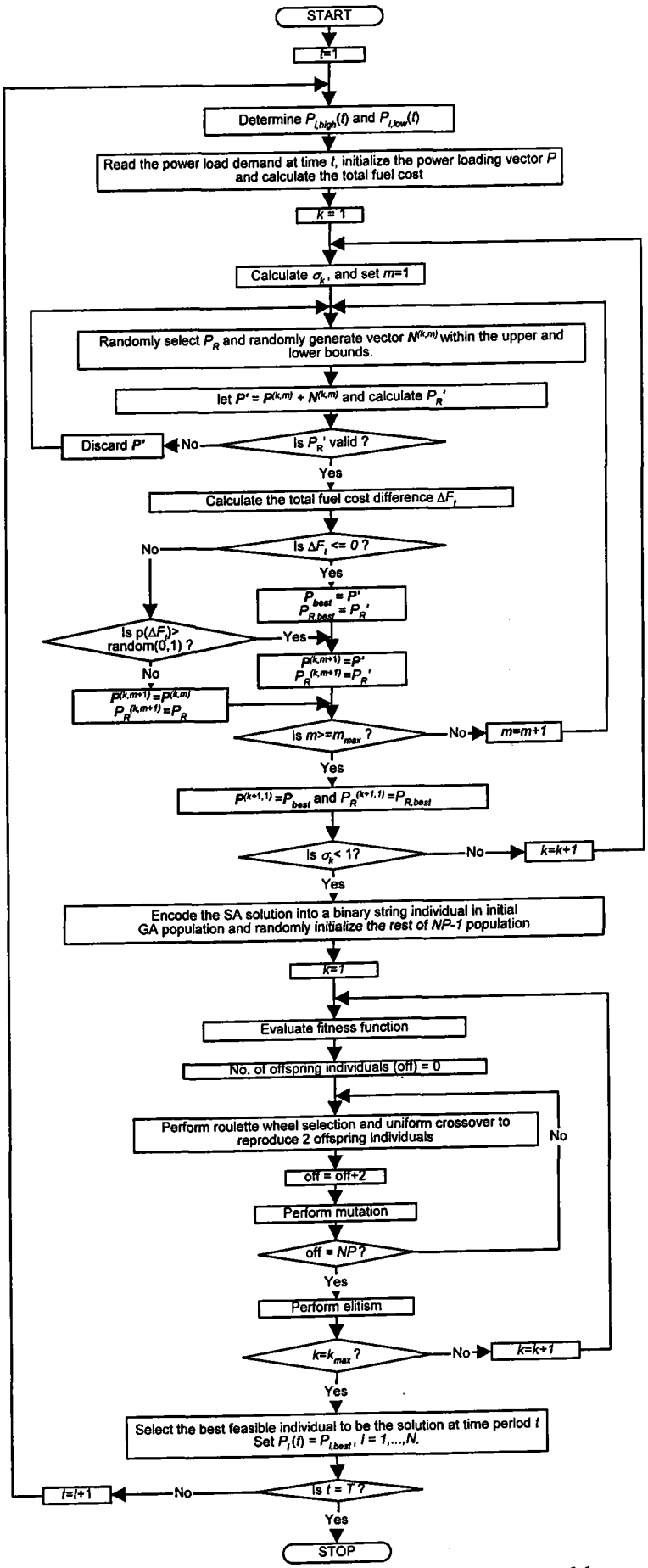

Figure 3.9 Flowchart of GA-SA for constrained ED problem

3.5 Local search (LS)

The local search (LS) technique is the hill-climbing method searching for the local optimal solutions [13]. LS starts searching the neighborhood solution of an initial solution. If either there's no better solution in its neighborhood or the maximum number of iterations is reached, the best solution reached is the LS solution. The different between SA and LS is that LS does not accept the worse solution, leading to the local optimal solution.

3.6 Merit order loading (MOL)

The merit order loading (MOL) determines the full-load average production cost index by

$$CI_i = C_i(P_{i,max}) / P_{i,max}. \tag{3.8}$$

The generators are dispatched in the ascending order of CI_i starting from $P_{i,low}(t)$ to $P_{i,high}(t)$ of each generating unit. This method guarantees a feasible solution. Table 3.4 shows the MOL full-load average production cost index and the dispatch order of 10 generating unit system.

Table 3.4 MOL full-load average production cost index and dispatch order for 10 generating unit system

Unit	CI_I	Dispatch order
RY_CC#1	531.3438	4
RY_CC#2	515.7944	1
RY_CC#3	526.9612	3
RY_CC#4	515.7944	2
BPK_T#1	888.6341	9
BPK_T#2	897.0933	10
BPK_T#3	881.8634	8
BPK_T#4	880.7859	7
KN_CC	621.5732	6
IPT_CC	565.9030	5