데이터베이스를 연계한 발전기 기동정지계획 어플리케이션 개발

오승렬* 백영식* 송경빈** 김재철*
경북대학교* 계명대학교 **

Development of Application for Unit Commitment using the Database

Seung-Yul Oh* Young-Sik Baek* Kyung-Bin Song** Jae-Chul Kim*
Kyungpook National Univ* Keimyung Univ**

Abstract - This paper presents a Case-Sort method to solve the unit commitment problem using database in electric power systems. The formulation of the unit commitment may be described as nonlinear mixed integer programming. However, it is hard to optimize a problem with discrete and continuous variables in a large-scale system at the same time. The Case-Sort method is based on the unit [MW] generation cost considered drive hour. Then, this paper shows effectiveness and economical efficiency of the proposed algorithm.

1. 서 론

전력계통의 효율적, 경제적인 운전을 위해서는 경제급전을 수행하기 전에 발전기의 기동 정지 계획이 먼저 수립되어야 한다. 화력발전기에 있어 기동정지계획은 수요, 운전예비력, 각 발전기의 제약조건 하에서 발전기의 총 연료비를 최소화하는 최적화 문제로 구성되며, 이는 각각의 발전기에 대한 유지, 보수계획, 수요예측, 신뢰도 해석 등이 이루어진 후에 시행된다. 이러한 최적화 문제를 접근하는 방법으로 지금까지 개발된 방법에는 여러 가지가 있으나 대규모 계통에서의 경제성과 효율성을 동시에 만족하기에는 그 부족함이 있다고 본다. 이에 본 논문에서는 투입시간을 고려한 단위[MW]당 발전비용에의한 순위*(이하Case-Sort) 를 결정하여 화력 발전기의 기동 정지 계획문제를 해결하는 새로운 알고리즘을 개발하였다. 또한 제시된 알고리즘의 경제성과 효율성을 입증하기 위해 같은 조건에서 수행한Genetic Algorithm Solution, Lagrangian Relaxation Method 그리고 Dynamic Programming의 결과를 비교 검토하였다.

2. 본 론

2.1 기동정지계획의 정식화

기동정지계획문제는 고찰대상 기간 중에 화력발전기의 총 연료비용을 최소화하는 최적화문제이며, 여기에서 총 연료비의 구성은

- 연료비용
- 기동비용
- 정지비용

으로 이루어지며, 연료비용은 열 비율과 연료가격에 의해 계산되며, 기동 비용은 발전기가 정지된 시간에 대한 함수로 표현되어진다. 다음으로 정지비용은 각 발전기의 매 정지시 일정한 비용으로 고정되어 있다.
또한 기동정지계획은 최적화 과정에서 다음과 같은 제약조건들을 만족 시켜야한다.

(1). 계통제약조건
① 수급조건
② 운전예비력 조건.

(2). 발전기제약조건
① 발전기 초기상태
② 발전기 상/하한 조건
③ 최소 운전/정지 시간
④ 발전기 상태제약조건(must-run, fixed-MW, unavailable, available)

2.1.1 목적함수

목적함수는 고찰대상기간 동안 전체 기동정지 대상 발전기의 연료비와 기동 및 정지 비용을 합한 총 연료비 F로 구성된다.

$$\text{Minimize } F = \text{Minimize} \sum_{t=1}^{DH} \sum_{i=1}^{NOU} (C_i^t U_i^t + SUC_i^t + SDC_i^t)$$

여기서, $C_i^t = \alpha_i (P_i^t)^2 + \beta_i P_i^t + \gamma_i$

$\alpha_i, \beta_i, \gamma_i$: i 발전기의 연료비 계수

P_i^t : t 시간대에서의 i-발전기의 출력[MW]

U_i^t : t 시간대에서의 i-발전기의 상태(1:기동 0:정지)

DH : 고찰 기간 (t = 1 . . . . T)

NOU : 기동정지대상 발전기 (i = 1 . . . . Ni)

SUC_i^t : t 시간대에서의 i-발전기 기동비용

$$SUC_i = \sigma_i + \delta_i \left[1 - \exp\left(\frac{-T_{off,i}}{\tau_i} \right) \right]$$

σ_i : hot start up cost of i_th unit .
δ_i : cold start up cost of i_th unit .
$T_{off,i}$: shut down hour of i_th unit.
τ_i : cooling time constraint of i_th unit.

SDC_i^t : t 시간대에서의 i-발전기 정지비용

2.1.2 적용된 제약조건

● 수급 제약조건
계통에 투입된 발전기의 출력의 합은 고려대상기간 동안의 예측된 각 부하를 만족시켜야 한다.

$$\sum_{i=1}^{N_i} P_i^t U_i^t = D^t \quad (D^t : \text{t 시간대의 부하 })$$

● 운전예비력 제약조건
계통에서는 발전기 탈락 등의 사고 및 급격한 부하증가에 대비하기 위하여 충분한 운전예비력을 확보하여야 하며, 발전기가 낼 수 있는 최대출력은 수요와 운전예비력 요구량을 합한 것 보다 커야한다.

$$\sum_{i=1}^{N_i} P_i^t U_i^t \geq D^t + SR^t \quad (SR^t : \text{t시간대에서의 운전예비력})$$

● 출력 상하한치 제약
발전기 출력에는 하한값 (P_i^{min})과 상한값 (P_i^{max}) 의 제약이 있어, 이 제약 내에서만 발전이 가능하다.

$$P_i^{min} \leq P_i \leq P_i^{max}$$

● 최소 운전 및 정지 시간
발전기는 일단 운전을 시작하면 최소한 어느 일정시간 동안은 운전을 계속하여야 하며 또한 정지하게 되면 최소한 어느 일정시간 동안은 정지하여야 한다.

$$T_i^{up} \geq T_i^{min\ up} \quad , \quad T_i^{down} \geq T_i^{min\ down}$$

● 발전기 초기상태
현재까지의 정지 또는 기동 되어온 시간을 고려한 발전기 기동정지계획이 수립되어야 한다.
● 발전기 가용상태
유지보수 등으로 투입이 불가능한 발전기를 제외한 상태에서 기동정지 계획이 수립되어야 한다.

2.2 발전기 기동정지계획 어플리케이션 구성

본 프로그램은 Visual Studio 6.0 에서 만들어 졌으며, 기본적으로 데이터의 입력은 프로그램 화면상에서 Data Base와 Data File에 의해 선택적으로 입력이 가능하며, 생성된 Data Base와 Data File은 화면상에서 추가, 삭제, 변경이 가능하도록 설계되어 있다. 또한 다른 프로그램간의 호환성 높이기 위해 프로그램은 기능에 따라 모듈화 되어 있다.아래의 그림 2는 본 프로그램의 전체적인 계층구조를 도식화하여 나타낸 것이다.

*주1 : Unit Commitment Algorithm

*주2 : Economic Dispatch Algorithm

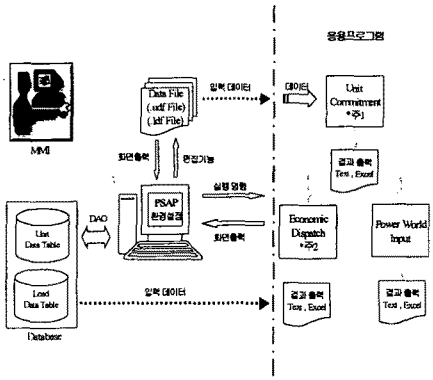

그림 1. 프로그램 계층구조

2.2.1 화면구성

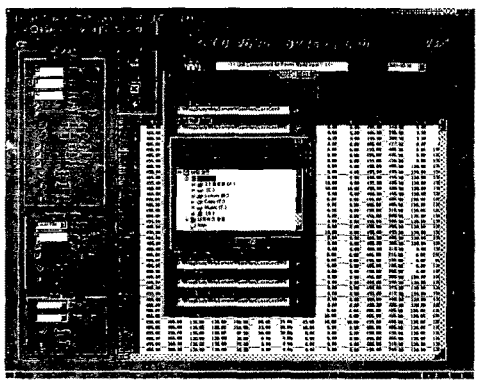

그림 2 어플리케이션 메인 화면

그림은 어플리케이션의 메인 화면과 환경설정을 위한 대화상자로, 실제 Unit Commitment, Economic Dispatch, Power World Simulator의 입력을 위해 프로그램을 실행하는 부분이다. 화면의 왼쪽 부분은 발전기기동정지 계획을 위해 필요한 조건을 입력하는 부분이며, 화면 오른쪽 부분은 실행결과를 화면상에서 볼 수 있도록, Microsoft Flexgrid Control을 이용하여 화면에 출력해 줌으로서 프로그램 진행과정을 파악하는데 좀더 용이하게끔 하였다. 또한 환경설정 기능은 사용자에 맞게끔 입출력 경로지정을 가능하게 하며, 이는 프로그램 이동시 더욱 편리한 기능을 제공한다. 이외의 화면으로는 Data Base 내부의 발전기와 부하 데이터의 수정, 삽입, 삭제 등 Database를 조작하는 부분과 프로그램의 입력형식의 하나인 파일 입력에 사용되는 Unit Data File과 Load Data File의 편집 기능을 담당하는 부분으로 구성된다. 본 프로그램에서는 발전기 데이터와 부하데이터를 쉽게 구분하기 위해서 Unit Data File은 확장자 *.udf , Load Data File은 확장자가 *.ldf로 이루어져 있다. 이는 다른 파일과의 구별이 용이하도록 하고 입력파일 선택시 필터링 효과를 위해 만들어진 확장자명 이다. .

2.2.2 Case-Sort에 의한 발전기 기동정지계획

매 시간 부하에 대해 투입 시간을 고려한 단위[MW]당 발전비용이 가장 적은 발전기를 투입하고 동일시간대에 부족부하에 대해 다시 이용 가능한 모든 발전기의 단위[MW]당 발전비용을 계산, 부하를 만족 할 때까지 이 과정을 반복한다.

※ 투입 예상 시간 Tin 동안

$$\text{Hourly Total Cost}_i = a_i P_i^2 + \beta_i P_{i+} \gamma_i + \text{SrtCost}_i / \text{Tin}_i = \text{HTC}_i [\$]$$

$$\text{단위[MW]당 발전비용}_i = \frac{\text{HTC}_i}{P_i} [\$]$$

아래의 그림3은 본 알고리즘을 이용한 발전기 기동정지 계획의 Flow Chart를 나타낸 것이다.

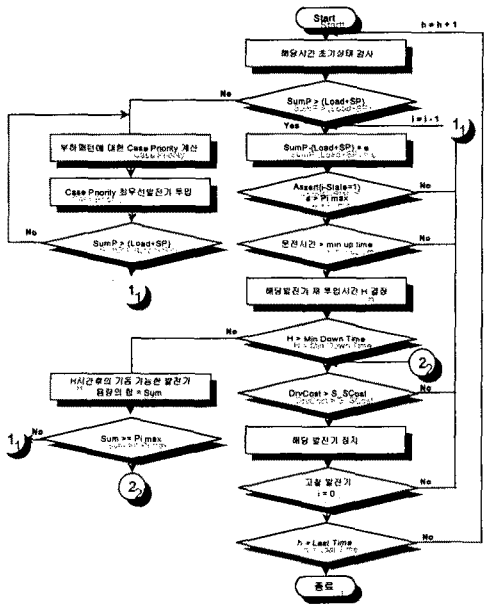

그림 3 Case-Sort 방법을 이용한 발전기 기동정지계획 순서도

2.2.3 Newton method를 이용한 경제급전

$$g(x+\Delta x) = g(x) + [g'(x)]\,\Delta x = 0$$

$$g(x) = \begin{bmatrix} g_1(x1, x2, x3) \\ g_2(x1, x2, x3) \\ g_3(x1, x2, x3) \end{bmatrix} \quad (3)$$

$$g'(x) = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} \end{bmatrix} \quad (4)$$

$$\Delta x = -[g'(x)]^{-1} g(x) \quad (5)$$

여기에 g 함수 대신 gradient vector ∇L_x 를 대입.

$$\Delta x = -\left[\frac{\partial}{\partial x} \nabla L_x\right]^{-1} \nabla L \quad (6)$$

경제급전문제는 다음 식으로 나타내어진다.

$$L = \sum_{i=1}^{N} F_i(P_i) + \lambda \left(P_{load} - \sum_{i=1}^{N} P_i \right) \quad (7)$$

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial P_1} \\ \frac{\partial L}{\partial P_2} \\ \frac{\partial L}{\partial P_3} \\ \frac{\partial L}{\partial \lambda} \end{bmatrix} = \begin{bmatrix} \frac{d}{\partial P_1} F_1(P_1) - \lambda \\ \frac{d}{\partial P_2} F_2(P_2) - \lambda \\ \frac{d}{\partial P_3} F_3(P_3) - \lambda \\ P_{load} - \sum_{i=1}^{N} P_i \end{bmatrix} \quad (8)$$

$$\left[\frac{\partial}{\partial x} \nabla L_x\right] = \begin{bmatrix} \frac{d^2L}{dx_1^2} & \frac{d^2L}{dx_1 dx_2} & \cdots \\ \frac{d^2L}{dx_2 dx_1} & \cdots & \cdots \\ \vdots & \vdots & \vdots \\ \frac{d^2L}{d\lambda dx_1} & \cdots & \cdots \end{bmatrix} \quad (9)$$

$$x^1 = x^0 + \Delta x \quad (10)$$

2.2.4 사례연구1

Table1은 본 논문에서 제안한 알고리즘의 유용성을 입증하기 위해서 실행한 Simulation결과이다. 이는 비교대상 논문과의 타당성 있는 결과 비교를 위해 발전기, 부하 데이터 및 제약조건들은 참고문헌[1]의 내용을 그대로 적용하였다.

ⓐ Total Operating Cost

Unit	D.P solution	Lagr.Rel. solution	Genetic Algorithm best	Genetic Algorithm worst	Case-Sort
	Operating Cost $			(dif %)	
10	565825 (-0.11)	565,825 (-0.11)	565,825 (-0.11)	570,032 (+0.63)	566,406 (base)
20	-	1,130,660 (+0.32)	1,126,243 (-0.06)	1,132,059 (+0.44)	1,127,108 (base)
40	-	2,258,503 (+0.30)	2,251,911 (+0.01)	2,259,706 (+0.36)	2,251,675 (base)
60	-	3,394,066 (+0.61)	3,376,625 (+0.09)	3,384,252 (+0.32)	3,373,363 (base)
80		4,526,022 (+0.61)	4,504,933 (+0.14)	4,510,129 (+0.25)	4,498,593 (base)
100		5,657,277 (+0.63)	5,627,437 (+0.09)	5,637,914 (+0.28)	5,622,017 (base)

ⓑ Total CPU Time

Unit	GA (1996년)	Case-Sort(2001년)
	Average CPU Time (sec)	
10	221	0.05
20	733	0.05
40	2627	0.11
60	5840	0.28
80	10036	0.6
100	15733	0.93

Table1. Simulation results for up to 100-unit systems.

2.2.5 사례연구2

다음의 결과는 참고문헌[6]의 데이터를 이용한 Simulation 결과이다. 이 역시 결과의 타당성을 위해 비교 논문과의 같은 조건하에서 실행된 결과이다.

Solution	Hybrid GA	Normal GA	Case-Sort
Cost	3,826,775 (-0.40%)	3,834,467 (-0.20%)	3,842,168 (base)
Cpu Time	20 minutes	12 Hour	1 sec

Table2. Simulation results for 110-unit systems.

위에 나타난 사례연구 1,2의 결과와 같이 본 연구에서 제안한 Case-Sort Method는 기존의 알고리즘을 사용한 결과와 비교해볼 때 비용 면에서는 큰 차이를 보이진 않는다. 오히려 [사례 연구 1]에서는 GA(best)의 10기, 20기 경우를 제외한 나머지 방법보다는 경제적인 결과를 보이고 있다. 하지만 계산시간 면에서 본다면 그 차이는 명료해진다. D.P Solution의 경우를 볼 때 발전기 20기에서부터는 탐색 조합의 수가 급격히 증가하면서 해를 구하지 못함을 볼 수 있다. 나머지 방법들 또한 그 결과는 산출했지만 계산시간의 엄청난 소요로 인해 On-Line 운전시에는 원하는 시간 내에 변경된 해를 얻을 수 있다는 보장이 없다. 비록 비교대상논문이
수년 전에 발표된 결과라 하지만 현재에도 그 시간상의 차이는 본 알고리즘의 결과와 현저한 차이를 보이고 있다. 그런 면에서 볼 때 본 논문에서 제안한 방법은 실계통에서의 On-Line 운전시 매우 만족할만한 결과를 얻을 수 있으리라 기대 된다.

3. 결 론

최근에 연구가 활발히 진행되고있는 Genetic Algorithm을 이용한 발전기 기동정지 계획의 경우 세대수에 따라 더욱 근접한 최적해에 도달할 수 있지만 그에 따른 계산시간이 현저하게 많이 소요된다는 취약점을 가지고 있다. 여러 가지 요인으로 인해 항상 부하 변동 또는 발전기 탈락사고 등을 내포하고있는 실계통 에서는 무엇보다 상황에 따른 신속한 대처가 중요시된다. 다시 말해 계산상에서의 소요시간은 크면 클수록 실계통에서의 On-Line 운전시에는 커다란 단점으로 지적 될 수 있다. 이에 본 논문에서는 전력계통에서의 경제적 운용을 위한 화력 발전기의 기동정지계획에 대하여 기존의 알고리즘보다 더욱 간략하면서도 경제적인 목표와 효율적인 면에서의 우수성을 여러 가지 사례연구를 통한 결과비교로서 이를 입증하였다.

[참 고 문 헌]

[1] S.A. Kazarlis, A.G. Bakirtzis, V. Petridis "A Genetic Algorithm Solution to The Unit Commitment Problem" IEEE Trans. Vol .11 1996

[2] Allen J.Wood, Bruce F. Wollenberg " Power Generation Operation and Control"John Wiley & Sons, New York

[3] Chuan-Ping Cheng, Chih-Wen Liu, Chun-Chang Liu "Unit Commitment by Lagrangian Relaxation and Genetic Algorithm" IEEE Trans. Vol.15 2000

[4] A.G Bakirtzis and C.E.Zoumas "Lamda of Lagrangian relaxation solution to unit commitment problem" IEE Pro-Gener.Trans. Distrib. Vol.147 March 2000

[5] K.A Juste, H.Kita, E.Tanaka, and J.Hasegawa "An Evolutionary Programming Solution to the Unit Commitment Problem" IEEE Trans. Vol.14 1999

[6] S.O Orero , M.R Lrving "Large scale unit commitment using a hybrid genetic algorithm" Elsevier Science Ltd. 1997

[7] C.Wang S.M.Shahidehpour "Effects of Ramp-Rate Limits on Unit Commitment and Economic Dispatch" IEEE Trans. Vol. 8 1993