병렬 타부 탐색을 이용한 발전기 기동정지계획의 최적화

(Optimization of Unit Commitment Schedule using Parallel Tabu Search)

이 용 환 † 황 준 하 †† 류 광 렬 ††† 박 준 호 ††††

(Yong Hwan Lee) (Junha Hwang) (Kwang Ryel Ryu) (Junho Park)

요 약 발전기 기동정지 계획은 하나의 전력시스템을 형성하는 다수의 발전기에 대해서 주어진 여러 제약을 따르는 일간 또는 주간의 기동 및 정지시간을 결정하는 작업으로 다양한 제약과 방대한 탐색공간으로 인해 최적의 경제적 계획 수립이 매우 어려운 대규모 최적화 문제이다. 타부 탐색은 보통의 지역적 탐색법에 비해 국지적 최적해에 빠질 위험이 적고 다른 전역적 탐색기법에 비해 대상문제에 관한 지식을 충분히 활용하기에 유리하여 많은 최적화 문제에 사용되고 있다. 그러나 규모가 방대하면서 많은 제약조건이 존재하는 대규모 최적화 문제들은 타부 탐색으로도 빠른 시간내에 최적의 해를 찾아내기 힘들다. 본 논문은 대규모 최적화 문제의 하나인 발전기 기동정지 계획 문제를 타부 탐색의 병렬화를 통해 해결함으로써 탐색 소요시간의 단축과 함께 해의 질 또한 향상시킬 수 있음을 보여준다.

키워드 : 병렬타부탐색, 메타휴리스틱 알고리즘, 발전기 기동정지 계획 문제, 최적화

Abstract The unit commitment problem in a power system involves determining the start-up and shut-down schedules of many dynamos for a day or a week while satisfying the power demands and diverse constraints of the individual units in the system. It is very difficult to derive an economically optimal schedule due to its huge search space when the number of dynamos involved is large. Tabu search is a popular solution method used for various optimization problems because it is equipped with effective means of searching beyond local optima and also it can naturally incorporate and exploit domain knowledge specific to the target problem. When given a large-scaled problem with a number of complicated constraints, however, tabu search cannot easily find a good solution within a reasonable time. This paper shows that a large-scaled optimization problem such as the unit commitment problem can be solved efficiently by using a parallel tabu search. The parallel tabu search not only reduces the search time significantly but also finds a solution of better quality.

Key words : Parallel Tabu Search, Meta heuristic algorithm, Unit Commitment Problem, Optimization

· 본 연구는 한국과학재단 목적기초연구(과제번호 R01-1999-00216)지원으로 수행되었음.

† 비 회 원 : 부산대학교 컴퓨터공학과
yhlee1@pusan.ac.kr
†† 정 회 원 : 금오공과대학교 컴퓨터공학부 교수
jhhwang@cespc1.kumoh.ac.kr
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
부산대학교 컴퓨터및정보통신연구소 연구원
krryu@pusan.ac.kr
†††† 비 회 원 : 부산대학교 전자전기정보컴퓨터공학부 교수
parkjh@pusan.ac.kr

논문접수 : 2002년 3월 2일
심사완료 : 2002년 6월 27일

1. 서 론

산업 전반에 걸쳐 많은 최적화 문제들이 있고, 그 중 많은 문제들이 방대한 해공간과 많은 제약조건으로 인해 최적의 해를 찾는 것이 계산적으로 불가능하다. 따라서 최적의 해를 찾기보다는 최적에 가까운 해를 찾고자 하는 노력이 이루어져 왔다. 최적화 문제의 현실적 해결을 위하여 hill-climbing 탐색, greedy 탐색과 같은 휴리스틱 기법들과 유전알고리즘, 타부(tabu) 탐색, ant colony 알고리즘, simulated annealing등의 많은 메타

휴리스틱 기법들이 개발되고 사용되었다. 휴리스틱 기법들은 빠른 시간에 해를 찾아내는 장점이 있으나 쉽게 지역적 최적해에 빠진다. 따라서 탐색 시간이 다소 길어지더라도 보다 세련된 탐색전략을 구사하여 전역적 탐색을 할 수 있는 메타 휴리스틱 기법에 기반한 방법들이 많이 이용된다.

본 논문에서는 대규모 최적화 문제 중 하나인 발전기 기동정지계획 수립을 위하여 타부 탐색을 적용한다. 타부 탐색[1]은 지역적 탐색을 바탕으로 빠른 탐색을 수행하면서 적응성 있는 메모리(adaptive memory)를 이용하여 전역적 탐색을 가능하게 한다. 타부 탐색이 다른 메타휴리스틱 기법보다 빠른 탐색을 수행하지만 고난도의 대규모 최적화 문제에서는 탐색 소요시간이 길어지며, 지역적 탐색을 바탕으로 하기 때문에 여전히 전역적 탐색에 한계가 있다. 따라서 본 논문에서는 타부 탐색을 병렬화 함으로써 이러한 한계를 극복하고 보다 우수한 탐색 성능을 보일 수 있는 방안을 제시하고자 한다.

발전기 기동정지계획의 수립에 대한 기존의 방법들로는 우선순위법[2], 라그랑지 미정계수법(lagrangian relaxation method)[3]등이 있다. 우선순위법은 휴리스틱에 기반한 방법으로 매우 빠른 탐색을 하지만 국지적 최적해에 빠지는 단점이 있고, 라그랑지 미정계수법 또한 비교적 빠른 시간에 해를 구할 수 있지만 대상 문제의 복잡성과 비 선형적 특성으로 인해 수렴하지 못하는 경우가 발생하고, 찾아내는 해 또한 국지적 최적해이다. 보다 전역적인 탐색방법으로 simulated annealing[4], 전문가 시스템[5], 신경회로망[6], 유전알고리즘[7, 8]등을 이용한 방법이 제시된바 있다. 이들 방법은 대상문제의 규모가 커짐에 따라 탐색시간이 지나치게 증가하므로 10개에서 38개의 발전기를 대상으로 한 소규모 기동정지계획 문제에 대해서만 적용되었다. 본 논문에서 제안하는 병렬 타부탐색 기법은 실험결과 152대의 발전기가 개입된 대규모 문제에 대해 우수한 해를 찾아주는 것으로 확인되었다.

다음 장에서는 대상 문제인 발전기 기동정지계획 문제에 대해서 설명한다. 3, 4, 5장에서는 타부 탐색의 적용방법에 대해서 소개하고, 6장에서는 실 데이타를 이용한 실험을 통하여 다른 탐색 방법과 성능을 비교한다. 마지막으로 7장에서는 결론과 향후 연구방향에 대해서 토의한다.

2. 발전기 기동정지계획 문제

발전기 기동정지계획이란 하나의 전력시스템을 형성하는 여러 대의 발전기에 대해서 안정적인 전력 공급체계를 확보하면서 총 발전비용을 최소화하기 위해 각 발전기별로 일간 또는 수일간의 기동 및 정지시간을 결정하는 작업이다. 전력 수급량이 증가하고 발전기 수가 늘어나면서 종류도 다양해짐에 따라 효율적인 기동정지계획의 수립은 경제성의 관점에서 중요한 문제로 부각되었다. 그러나, 각 발전기마다 고려해야 할 제약조건이 많을 뿐 아니라 여러 발전기 간에 존재하는 제약조건들도 많아 효율적인 발전계획을 도출하는 것이 상당히 어렵다.

발전기 기동정지 계획 문제에서 고려해야 할 제약 조건들은 다음과 같다.

(1) 발전 요구량: 매 시간대별로 요구되는 발전량으로 부하수요와 송전손실량을 함께 만족시킬 만큼 전력을 공급하여야 한다.

(2) 운전 예비력: 예상하지 못한 전력부하의 급증이나 일부발전기의 고장에 대비해 항상 어느 정도의 예비 전력을 공급하여야 한다.

(3) 발전기의 초기 조건: 계획 시점에서의 각 발전기의 기동 정지 상태 및 지속시간 등을 고려하여야 한다.

(4) 최소기동시간, 최소정지시간: 각 발전기는 그 종류에 따라 기동 정지 상태를 전환하기 전에 이전상태에서 지속하여야 하는 최소 요구시간이 있다.

(5) 기타 발전기의 최대, 최소 발전 한계량, 상시운전 등의 여러 제약사항이 있다.

(3), (4)번 제약조건은 발전기 별로 반드시 지켜야 하는 절대적 제약으로 어느 하나의 발전기라도 지켜지지 않으면 계획에 따라 발전을 할 수가 없다. 반면 (1), (2)번의 제약은 지키지 않는다 하여 계획대로 발전이 되지 않는 것은 아니다. 그러나 경제성을 위해서는 요구량(발전 요구량 + 운전 예비력)보다는 많은 양을 발전하면서 요구량과 실 발전량의 차이는 최소가 되도록 하여야 한다. 실 발전량이 요구량보다 적게 되면 대규모 정전과 같은 사고 발생의 우려가 있다.

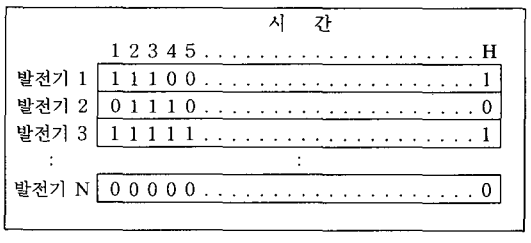


그림 1 발전기 기동정지 계획의 표현

발전기 기동정지계획을 표현하기 위한 한 방안으로서 각 발전기마다 시간별로 기동정지 여부를 이진값(기동

1, 정지 0)으로 나타내어 보면 그림 1과 같이 된다. 예를 들어 특정발전기의 24시간 계획이 000000 111111 000000 000000 이라면 0시부터 6시까지 정지하고 6시부터 12시까지 기동, 다시 12시부터 24시까지 정지하는 것을 의미하게 된다. 만약 이 발전기의 최소정지시간이 6시간이고 최소기동시간은 12시간이라면 이 계획은 최소기동시간을 만족하지 못하기 때문에 잘못된 계획이 된다. 최소시간 제약에 의해서 특정 시간의 기동정지 여부를 수정하기 위해서는 주위의 몇 시간이 동시에 수정되어야 한다. 따라서 동시에 여러 시간대의 발전량에 변화가 발생되고 발전비용에도 많은 차이가 난다. 이와 같이 발전기 기동정지계획은 해의 일부분의 변화에 의해서 평가값이 대폭 변화하는 불규칙적인 탐색공간을 가지고 있어 최적해의 탐색이 매우 힘든 문제이다.

```
TabuSearch()
{
    ..... /* 초기화 */
    curSol = InitSolution(); /* 초기해 생성 */
    curEvalVal = Evaluate(curSol); /* 평가, 발전비용계산 */
    while(!SatisfyTerminateCondition()) /* 종료조건을 만족할 때까지
    반복 */
    {
        evalValOfBestNeighbor = INF; /* 매우 큰 값 */
        while(!CheckAllNeighborSol()) /* 이웃해를 모두 검사할
        때까지 반복 */
        {
            neighborSol = MakeNeighbor(curSol); /* 하나의 이웃해
            생성 */
            evalVal = Evaluate(neighborSol); /* 평가 */
            /* 타부에 대한 벌점 부가, 재방문 방지 */
            if(IsTabu(neighborSol))
                evalVal += TABU_PENALTY;
            /* 빈도수 기반 메모리를 통한 다각화 */
            if(IsDiverseMode()) /* 다각화 모드이면 */
            {
                /* 방문빈도수에 비례하는 벌점 부가 */
                evalVal += FREQ_PENALTY * Freq(neighborSol);
            }
            /* 가장 좋은 이웃해 기록 */
            if(evalVal < evalValOfBestNeighbor)
            {
                bestNeighborSol = neighborSol;
                evalValOfBestNeighbor = evalVal;
            }
        }
        /* 가장 좋은 이웃해(bestNeighborSol)로의 이동 */
        UpdateTabuList(bestNeighborSol); /* 타부리스트 갱신 */
        UpdateFrequencyMemory(bestNeighborSol); /*
        빈도수기반메모리 갱신 */
        MoveSolution(curSol, bestNeighborSol); /* curSol =
        bestNeighborSol */
    }
}
```

그림 2 타부 탐색 알고리즘

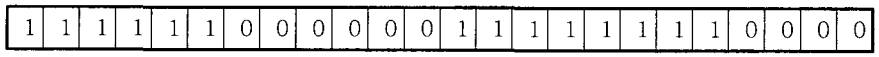
3. 타부 탐색의 적용

타부 탐색은 최적화 문제를 해결하기 위해 흔히 사용하는 메타휴리스틱 기법이다. 그림 2에 기술된 바와 같이 타부 탐색은 보통의 지역적 탐색법처럼 하나의 후보해에서 시작하여 주어진 종료조건을 만족할 때까지 반복적으로 이웃해로 이동을 한다. 그러나 타부 탐색은 국지최적해에 빠지는 것을 피하기 위한 전략들을 사용함으로써 지역적 탐색을 넘어선다. 하나의 해에서 이웃의 다른 해로 이동할 때, 비록 이전의 해보다 좋지 않더라도 이웃해 중 가장 좋은 해로 이동을 하여 국지 최적해를 벗어나기 위한 이동을 시도한다. 그러나 다음의 이동에서 가장 좋은 해들을 계속 선택하다 보면 이전의 국지 최적해로 다시 돌아올 확률이 높으므로, 상당수의 이동이 이루어지는 동안 지난번의 국지적 최적해로의 회귀를 방지하는 수단이 필요하다. 이를 위해서, 타부 탐색은 타부리스트라는 단기 메모리(short term memory)에 가장 최근 일정 횟수의 이동내역을 기록하고, 이 리스트에 기록된 해(타부 상태의 해)로 이동하려는 경우 매우 큰 벌점을 부가하여 그러한 이동을 억제한다. 타부 리스트는 보통 FIFO(first-in-first-out)기반의 큐(queue)의 형태로 구현된다. 즉, 이동시 이동내역이 타부리스트의 끝으로 추가됨과 동시에 리스트 제일 앞의 이동내역은 삭제된다. 타부리스트에 새로운 내역이 추가될 때마다 타부리스트의 모든 내역들은 한 칸씩 앞으로 옮겨가게 되고 가장 앞에 있던 내역은 삭제되어 타부리스트는 항상 가장 최근 일정 횟수의 이동내역을 유지하게 된다.

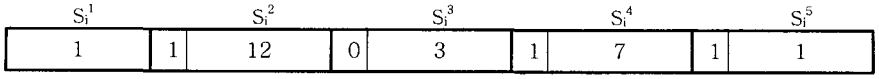
타부리스트를 이용하여 탐색을 국지적 최적해로부터 어느 정도 벗어나게 할 수는 있으나 타부리스트만을 가지고는 대규모의 해공간에서 다양하고 다각화적인 탐색을 유도하기는 힘들다. 따라서 타부탐색에서는 보다 전역적인 탐색을 위하여 빈도수 기반 메모리(frequency based memory)등의 장기 메모리(long term memory)를 이용하여 탐색의 다각화를 시도할 수 있다. 빈도수 기반 메모리에는 현재까지의 이동과정에서 선택된 해의 패턴들의 빈도수를 기록하고 이웃해 생성 시 생성된 해의 패턴에 따라 빈도수에 비례하는 벌점을 주어 빈번하게 탐색한 영역으로의 탐색을 지양하고 탐색하지 않는 영역으로의 탐색을 유도하여 전역적인 탐색이 이루어지도록 한다.

3.1 해의 표현

타부 탐색을 적용하기 위한 후보해의 표현 방법으로 가장 쉽게 생각할 수 있는 것은 발전기 기동정지 계획의 표현 방법 그대로 단위시간별 상태를 표시하여 2진부호화하는 방법이다. 즉, 그림 3(a)와 같이 발전기별로 시간당 하나의 비트씩 할당하여 이진값으로 나타내는 방법이다. 이때 이웃해는 임의의 한 비트의 값을 변경함으로써 생성할 수 있다. 그림 3(a)에서 최소정지시간과 최소기동시간이 모두 6이라고 하면 그림의 계획은 최소시간 제약을 따르고 있는 계획이 된다. 그러나 이웃해 생성을 위하여 예를들어 7시의 계획이 변경된다면 최소정지시간을



(a) 단위시간별 상태 표시에 의한 해의 표현



(b) 부스트링별 상태지속시간 표시에 의한 해의 표현

그림 3 후보해의 표현 방법

어기는 해가 된다. 따라서 위의 방법을 따를 경우 이웃해 생성과정에서 최소정지시간과 최소기동시간을 지키지 못하는 해가 나타날 수 있으며 이러한 최소시간 제약을 만족하는 실행 가능한 해(feasible solution)를 찾는 것이 힘들어 진다. 본 논문에서는 최소정지시간과 최소기동시간을 반드시 지키도록 하는 [9]의 방법을 개선하여 사용하였다.

하나의 후보해는 개별 발전기에 대한 스트링(string)의 집합으로 구성되어진다. 하나의 발전기 i에 대한 기동정지계획의 스트링 S_i는 그림 3(b)와 같이 여러 개의 부스트링(substring)으로 구성된다. 하나의 부스트링은 1비트의 상태플랙과 정수값을 가지는 지속시간부로 이루어진다. 단, 첫 번째 부스트링은 상태플랙을 가지지 않는다.

각 부스트링에서의 상태플랙은 기동정지 여부를 나타내고, 지속시간부는 자신이 나타내는 정수값에 발전기 i 고유의 최소정지(기동)시간을 더하여 실제 지속시간을 나타낸다. 즉, 각 부스트링의 의미는 상태플랙이 나타내는 상태를 지속시간만큼 지속함을 의미하는 것이다. 첫 번째 부스트링은 계획시점 이전의 상태를 고려하여 해석되어야 하므로 따로 상태플랙을 가지지 않는다. 이전 상태가 기동상태이면 지속시간만큼 기동을 하고 정지상태이면 지속시간만큼 정지를 한다. 지속시간부가 가질수 있는 값의 범위는 0부터 발전기 i의 최소정지시간과 최소기동시간 중에서 큰 값까지로 하였다. 이와 같이 지속시간부의 범위를 결정하게 되면 최소정지시간이나 최소기동시간 이상의 모든 지속시간을 표현할 수 있다. 스트링 S_i의 부스트링의 개수 n_i는 다음의 식으로 구한다.

$$n_i = \lceil \frac{N_H}{t_i^{\min}} \rceil + 1$$

단,

N_H : 기동정지계획 대상 시간

$t_i^{\min}$: 발전기 i의 최소정지시간과 최소기동시간 중에서 작은 값

3.2 이웃해 생성

타부 탐색과 같은 지역적 탐색법에서는 이웃해를 어떤 방식으로 생성시키는가에 따라 탐색성능이 달라진다. 일반적으로 이웃해는 현재해의 일부분을 변경하여 생성한다. 본 논문에서의 이웃해 생성 방식은 지속시간부의 값을 한 시간 증감하거나 상태플랙의 값을 전환하여 생성하였다.

그림 4에서 (b)는 지속시간을 한 시간 증감하여 생성된 후보해를 보여주고 있다. (a)의 첫 번째 부스트링의

(a) 현재해

(b) 지속시간부의 수정에 의해 생성된 이웃해

(c) 상태플랙의 수정에 의해 생성된 이웃해

그림 4 이웃해 생성

경우 지속시간부는 1을 나타내고 있기 때문에 한 시간을 증감하면 2 또는 0이 될 수 있다. 따라서 (b)의 그림과 같은 이웃해들이 생성된다. 나머지 부스트링에 대해서도 같은 방식으로 이웃해를 생성한다. (c)의 그림은 현재해의 두 번째, 세 번째 부스트링에서 상태플랙의 값을 전환하여 생성된 이웃해를 보여주고 있다. 다른 부스트링에 대해서도 같은 방식으로 이웃해를 생성한다. 모든 발전기에 대한 스트링마다 위의 방식으로 이웃해들을 생성할수 있다. 그러나 현재해로부터 모든 가능한 이웃해를 생성하게 되면 이웃해의 개수가 너무 많아 많은 시간이 소요된다. 따라서 본 연구에서는 모든 가능한 이웃해를 생성하지 않고 매계변수로 주어진 개수만큼 무작위적으로 이웃해를 생성하였다.

3.3 타부리스트

현재해로부터 위의 방법에 의해 이웃해를 생성하고 나면 가장 좋은 이웃해로 이동을 한다. 타부 탐색에서는 타부리스트를 관리하여 현재해가 이동을 할 때마다 이동내역을 타부리스트에 삽입하여 이미 방문한 해로의 재 방문을 막고 탐색의 다각화를 유도한다. 타부리스트에는 방금 방문한 해 자체를 저장할 수도 있으나 일반적으로 이동시 해에서 실제 변화가 나타난 부분(속성)만을 저장한다. 본 연구에서는 이웃해 생성시 변화가 가해진 곳의 발전기번호(스트링번호), 부스트링번호, 변화위치(상태플랙 혹은 지속시간부) 세가지 속성을 타부리스트에 저장하였다. 그리고 inc_list와 dec_list의 두 개의 타부리스트를 두어 inc_list에는 지속 시간을 한 시간 증가시키거나 상태플랙을 0에서 1로 전환하여 생성된 경우를 삽입하여 추후에 지속시간을 한 시간 감소시키거나 상태플랙을 1에서 0으로 전환하여 원래의 해로 돌아가는 것을 방지하도록 하고, dec_list는 지속 시간을 한 시간 감소시키거나 상태플랙을 1에서 0으로 전환하여 생성된 경우를 삽입하여 추후에 지속시간을 한 시간 증가시키거나 상태플랙을 0에서 1로 전환하여 원래의 해로 돌아가는 것을 방지하였다.

3.4 탐색의 다각화

타부 리스트만 이용해서는 타부 탐색이 국지적 최적해를 벗어나 전역적 탐색을 하도록 이끌기 어렵다. 따라서 보다 전역적 탐색을 이끌기 위한 다각화 방안이 필요하다. 본 연구에서는 다각화 방안으로서 빈도수 기반 메모리를 이용하였다. 발전기와 시간대에 대한 빈도수 기반 메모리를 두어 각 발전기와 각 시간대에 대해서 계획 변경 횟수를 기록하여 둔다. 빈도수 기반 메모리의 값은 매 이동시마다 계획의 변경이 발생한 발전기와 시간대에 대해서 빈도 값을 증가시킨다. 빈도수 기반 메모리를 지속적으로 관리하다가 탐색이 장시간 동안 진전이 없을 때 일정 기간 동안 다각화를 적용하는데, 그림 2에서 보는 바와 같이 다각화시에는 변경 빈도수가 높은 발전기나 시간대의 계획을 수정하여 만들어진 이웃해에 대해서 그 빈도수에 비례하는 벌점을 부여함으로써 상대적으로 많이 탐색하지 않은 공간으로의 탐색을 유도한다.

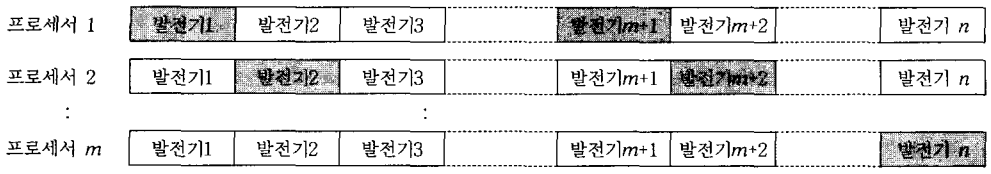
4. 타부 탐색의 병렬화

타부 탐색은 지역적 탐색을 기반으로 하여 빠른 탐색을 하면서 여러 가지 전략을 사용하여 탐색을 다각화함으로써 전역적 탐색을 할 수 있다. 그러나 대규모 최적화 문제는 복잡하면서 방대한 탐색 공간을 가지기 때문에 일반적인 타부 탐색으로는 적절한 시간 안에 탐색을 끝내기 어렵고 충분히 전역적 탐색을 하기도 힘들다. 따라서 본 연구에서는 타부 탐색을 병렬화함으로써 이러한 문제점을 해결하고자 하였다.

4.1 병렬화 모델

타부 탐색의 병렬화 모델은 크게 전역 모델과 분산 모델로 구분할 수 있다. 전역 모델은 하나의 관리프로세서와(master)와 여러 개의 작업프로세서(slave)로 구성되며, 관리프로세서가 타부리스트와 빈도수 기반 메모리를 유지하면서 탐색의 전 과정을 관리, 수행하고 이웃해들의 생성 및 평가와 같이 계산량이 많은 작업을 여러 작업프로세서에 나누어 할당하여 처리하는 방법이다. 관리프로세서가 각 작업프로세서에 작업을 할당하면 작업프로세서들은 관리프로세서로부터 현재해를 받아 이웃해들을 생성하고 그 중 가장 좋은 이웃해를 관리프로세서로 전달한다. 각 프로세서로부터 이웃해들을 받은 관리프로세서는 그 중 가장 좋은 이웃해를 선택하여 새로운 현재해로 이동을 한다. 이 모델은 구현은 쉬우나, 작업이 먼저 끝난 작업프로세서는 타 프로세서들의 작업이 끝날 때 까지 기다려야 하는 문제가 있다. 따라서 각 작업프로세서의 작업이 비슷한 시간에 끝이나도록 동기화를 엄격히 유지해야 한다는 단점이 있으며[10], 또한 병렬화를 통한 계산시간의 단축 효과 외 다른 이점을 기대하지 못한다.

분산 모델에서는 여러 프로세서가 각기 독립적인 타부 탐색을 수행하면서 일정 주기로 우수한 해를 서로 교환하면서 탐색을 수행한다. 이 모델은 주로 각 프로세서가 서로 다른 탐색 전략을 구사하도록 하여 프로세서간 탐색의 다양화를 유지하여 전역적인 탐색이 효과적으로 이루어지도록 한다. 또한 전역모델처럼 타 프로세서의 작업시간에 대한 의존성의 거의 없어 프로세서간



각 프로세서는 음영부분의 영역만 수정하여 이웃해 생성

그림 5 탐색 공간의 분할

의 동기화 유지가 필요치 않다. 실제로 분산 모델의 경우 계산 시간의 단축뿐만 아니라 보다 우수한 해를 찾게 되는 효과를 얻은 경우가 보고된 바 있다[11, 12]. 이 외에도 문제를 분해(problem decomposition)하여 부문제(subprobelm)를 병렬로 풀게 하는 접근 방안 등이 있다[13]. 본 논문에서는 타부 탐색의 병렬화 모델로서 분산 모델을 채택하였다.

4.2 탐색 영역의 배분

분산 모델의 적용 시 가장 중요한 요소는 프로세서간 탐색의 다양화를 유지하는 것이다. 프로세서간의 다양화 유지를 위해서는 다음과 같은 전략들을 생각할 수 있다. 첫 번째 방법은 프로세서마다 탐색 전략을 달리하는 것이다. 즉, 타부 리스트의 크기, 이웃해 생성방법, 집중화·다각화 전략 등을 프로세서마다 달리하여 서로 다른 방향으로의 탐색을 유도하는 방식이다. 두 번째 방법은 직접적으로 탐색공간을 분할하는 방식이다. 즉, 프로세서마다 이웃해 생성범위를 제한하여 특정 영역만 탐색을 하도록 하는 방식이다. 본 연구에서의 다양화 유지 전략으로는 후자의 방법이 더 효과적인 것으로 판명되었다.

타부 탐색의 병렬화 시 탐색을 수행하는데 소요되는 시간을 프로세서 수에 비례해서 단축하기 위해서는 병렬화하지 않았을 때 적정한 것으로 판단된 생성 이웃해의 수를 프로세서의 수로 나눈 만큼 각 프로세서에서 이웃해를 생성하도록 하는 것을 생각할 수 있다. 그러나 각 프로세서에서 너무 적은 수의 이웃해만 생성하게 하면 대상 탐색 영역을 세밀하게 탐색하지 못할 우려가 있다. 그러나 본 논문에서 제안하는 방법에 따라 프로세서마다 탐색공간을 제한을 하게 되면 적은 수의 이웃해로도 충분한 탐색을 할 수가 있게 된다. 발전기 기동정지 계획에서의 탐색 영역의 분할은 발전기별 분할과 시간대별 분할을 생각할 수 있다. 발전기 기동정지 계획문제의 경우에는 부스트링들을 수정하여 이웃해를 생성하는데 각 부스트링이 나타내는 시간대는 유동적이기 때문에(예를들어 발전기 i의 두 번째 부스트링과 발전기 j의 두 번째 부스트링이 동일시간대의 상태를 표시하지 않는다.) 시간대별로 탐색 공간을 분할하는 것이 힘들다. 따라서 탐색 영역의 분할은 발전기별로 하였다. 그림 5는 발전기별로 각 프로세서의 탐색영역을 분할한 예이다. 이처럼 각 프로세서에서의 이웃해 생성 영역을 서로 분할하게 되면, 전체영역에 대해서 생성했던 이웃해를 일부 영역에 대해서만 생성하므로 그 영역에 대해서는 보다 세밀한 탐색이 가능해진다. 그리고 전체 프로세서의 관점에서 보면 각 프로세서가 서로 다른 영역을 탐색하므로 탐색의 다양성 유지가 저절로 이루어지게 된다.

본 연구에서는 탐색공간의 분할을 위해서 전체 발전기를 일정등분으로 나누어 각 프로세서에 할당하였다. 이때 특정 종류의 발전기들이 한 등분으로 들어가지 않고 각 등분에 고루 분포하도록 분배를 하였다. 이와 같은 방법으로 각 프로세서는 특정 영역에 대해서 집중적 탐색을 하면서, 일정주기마다 이웃 프로세서로부터 우수한 해를 받아 들여 그 해로부터 다시 자신이 담당한 영역을 탐색함으로써 전체적으로는 다양성을 유지하면서 효율적인 탐색이 가능하게 하였다.

4.3 연결구조

병렬 타부 탐색에서 프로세서들간의 연결구조(topology)는 성능을 좌우하는 중요한 요소 중 하나이다. 일반적으로 연결구조가 조밀하면 프로세서간의 이웃해 전파가 너무 과도하게 이루어짐에 따라 다양성이 유지되지 못하여 해의 질이 떨어지게 되는 문제가 있다. 반면 연결구조가 느슨하면 프로세서간의 다양성의 유지에는 유리하나 우수한 해의 전달 속도가 지나치게 저하된다는 문제가 있다. 본 논문에서는 하이퍼큐브의 형태로 구현하였다. 그림 6에서 보는 바와 같이 하이퍼큐브는 모든 프로세서에서의 링크의 수가 $\log_2 n$(n은 프로

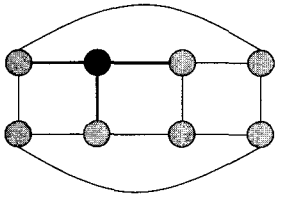


그림 6 하이퍼큐브 연결 구조

세서의 개수)이 되도록 연결하는 구조(예를 들어 프로세서의 수가 8개이면 각 프로세서에서의 링크의 수는 3개가 된다.)로서 실험 결과 지나치게 조밀하지도 지나치게 느슨하지도 않은 적절한 연결성을 보이는 것으로 확인되었다.

5. 비용 함수의 효율적인 계산

발전기 기동정지 계획에 의해 산출되는 총 발전 비용은 아래의 식(1)에 나타난 것처럼 연료비용, 기동비용, 제약조건 위배시 부가되는 벌점의 합으로 구성된다.

$$cost = \sum_{i=1}^{N_H} FC_i + \sum_{i=1}^{N_P} ST_i + \sum_{i=1}^{N_C} \mu_i V_i \qquad (1)$$

여기서,

N_H : 총 발전계획 대상 시간,

N_P : 총 계획 대상 발전기 수

N_C : 제약의 수,

FC_i : 시간 i의 연료비용

ST_i : 발전기 i의 기동비용,

μ_i : 제약조건 i의 위반에 대한 가중치

V_i : 제약조건 i의 위반 여부

식(1)의 기동비용은 발전기마다 주어진 매개변수 값에 따라 발전기의 기동 정지 여부에 따라 결정되는 비용으로 타 발전기와는 독립적으로 계산이 가능하다. 그러나 연료비용은 같은 시간에 기동되는 모든 발전기들을 고려하여 가장 경제적으로 유리한 방향으로 각 발전기의 출력이 결정되고 그 출력에 따라 비용이 계산된다. 각 발전기의 출력은 전력공학적 지식을 바탕으로 한 알고리즘에 의해 결정되는 것으로[14] 본 논문에서는 자세한 설명을 생략한다. 연료비용의 경우 하나의 발전기의 계획이 바뀌게 되면 그 시간대에서의 모든 발전기에 대한 비용이 변하게 된다. 식(1)의 마지막 항은 제약조건을 위반할 때 부여되는 벌점이다. 기동비용은 발전기 별로 계산하고, 연료비용은 시간별로 계산한다. 벌점은 발전기별 제약과 발전기 간의 제약에 따라, 발전기 별로 계산되는 것도 있고 시간별로 계산되는 것도 있다. 그러나 발전기별 제약인 발전기의 초기 조건 고려와 최소시간 제약은 후보해의 표현에 의해서 만족되고 있으므로 비용계산시 발전기 간의 제약만 고려하면 된다.

본 연구에서 평가를 위한 비용함수는 앞에서 본것처럼 연료비용, 기동비용, 제약을 어길 시 발생하는 벌점을 합산하여 계산하였다. 타부 탐색도중 이웃해를 평가하기 위해서 다시 이 모든 값을 처음부터 다시 계산하는 것은 시간적 낭비이다. 따라서 현재해와 이웃해 사이의 변화 부분에 대해서만 재 평가를 하는 것이 효율적이다. 발전기 기동정지 계획에서 이웃해를 평가하기 위해서는 기동비용은 변화가 나타난 발전기에 대해서 재 계산을 하면 되고, 연료비용과 제약 위배에 대한 벌점은 변화가 나타난 시간대에 대해서 재 계산을 하면 된다. 어느 한 이웃해의 계획이 그림 7과 같고 현재해에서 달라진 부분이 X표시 된 부분이라면 이 이웃해의 평가를 위해서는 X표시된 부분의 가로방향으로 기동비용을 재 계산하고 세로방향으로 연료비용과 제약에 대한 벌점을 재 계산하면 된다. 나머지 비용은 현재해의 비용을 그대로 이용하면 된다. 그러나 본 연구의 실험시 사용한 데이타는 일부 발전기들의 특성에 의하여 연료비용이 계획의 변화가 나타난 시간의 전후 한시간까지 달라지게 된다. 따라서 전후 1시간에 대해서도 연료비용을 재 계산하였다.

6. 실 험

병렬 타부 탐색의 성능 평가를 위하여 실제 발전기 152대의 일간 계획을 대상으로 서로 다른 발전요구량을 가지는 5개의 데이타에 대해서 실험을 하였다. 실험 환경으로는 Parsytec사의 CC16시스템을 사용하였다.

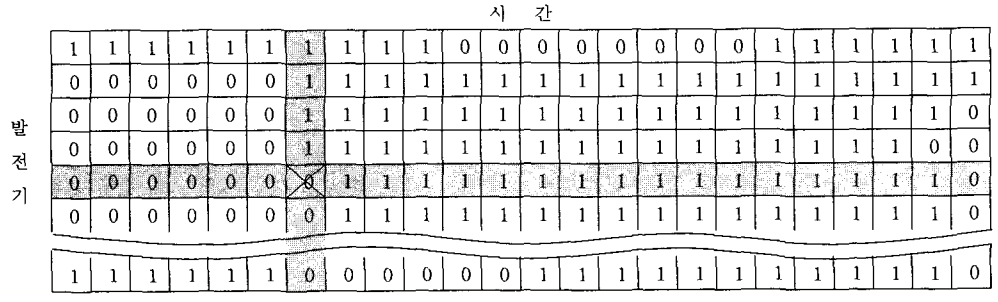


그림 7 점진적 평가

CC16시스템은 PowerPC 604 100MHz CPU 16개로 구성된 전형적인 MIMD(multiple instruction streams-multiple data streams)시스템이다.

본 논문의 병렬 타부 탐색의 성능 검증을 위하여 우선순위법과 유전알고리즘을 구현하였고, 유전알고리즘도 병렬화를 하였다. 우선순위법은 현재 실제 현장에서 발전기 기동정지 계획을 위해 사용하는 방법으로 각 발전기의 연료 효율에 따라 효율이 높은 발전기부터 기동을 시키도록 계획하는 휴리스틱 기법이다. 유전알고리즘[15]은 타부 탐색과 함께 최적화 문제에 많이 이용되는 메타휴리스틱 기법으로 전역적 탐색능력이 뛰어난 알고리즘이다. 병렬유전알고리즘과 병렬 타부 탐색에서의 해의 교환 주기 및 병렬 유전알고리즘의 교환 대상해와 교환해의 수는 다양한 실험을 통하여 가장 좋은 결과를 가지는 값으로 선택하였다. 우선 해의 교환주기는 두 알고리즘 모두 짧을수록 좋은 결과를 나타내었다. 따라서 두 방법 모두 교환주기는 1세대로 하였다. 병렬 유전알고리즘의 교환 대상해 및 교환해의 수에 관한 다양한 실험 결과 해집단에서 가장 우수한 해 1개를 전달할 때 가장 좋은 결과를 나타내었다.

표 1 각 탐색법을 통하여 찾은 최적해의 질(10^{10}원)

	우선 순위법	유전 알고리즘	타부 탐색	병렬유전 알고리즘	병렬타부 탐색
데이타 1	1.0483	1.0404	1.0488	1.0387	**1.0356**
데이타 2	1.0024	0.9967	0.9996	0.9966	**0.9953**
데이타 3	1.0139	1.0072	1.0176	1.0063	**1.0002**
데이타 4	1.0626	1.0526	1.0631	1.0497	**1.0432**
데이타 5	1.0614	1.0476	1.0518	1.0469	**1.0389**

각 데이타에 대한 실험 결과를 살펴보면 표 1과 같다. 우선순위법을 제외한 모든 알고리즘에서 총 이웃해 평가횟수는 동일하도록 하였다. 표 1에서 해의 질은 각 탐색법이 찾아낸 가장 우수한 발전 계획의 발전비용을 원 단위로 나타낸 것이다. 각 데이타에 대해서 가장 우수한 해는 굵은 글씨체로 나타내었고 가장 나쁜 해는 밑줄글씨체로 나타내었다. 표 1을 보면 모든 데이타에 대해서 병렬타부탐색이 가장 우수한 해를 찾아내는 것을 볼수 있다.

각 탐색법의 평균적인 탐색 성능을 비교하기 위하여 5개의 데이타에 대한 실험 결과값을 평균을 구하여 살펴보았다. 표 2에서 보는 바와 같이 유전알고리즘은 해의 질에 있어서 우수한 해를 찾아내지만 탐색 소요 시간이 매우 긴 것을 알 수 있다. 그러나 타부 탐색은 비

표 2 각 탐색법의 평균적인 성능 비교

	해의 질(1010원)	탐색 소요 시간
우선순위법	1.0377	< 1초
유전알고리즘	1.0289	62시간
타부 탐색	1.0362	2시간 16분
병렬유전알고리즘	1.0276	7시간 30분
병렬타부탐색	**1.0226**	30분

록 탐색 소요 시간에 있어서 유전알고리즘보다는 빠르지만 그래도 많은 시간이 소요되고 해의 질 또한 우선순위법과 유사한 수준에 머물고 있다. 이는 타부 탐색이 유전알고리즘만큼 충분히 전역적 탐색을 하지 못하기 때문이다. 그러나 타부 탐색을 병렬화함으로써 해의 질이 향상되고 탐색 소요 시간도 상당히 줄어든 것을 알 수 있다. 타부 탐색이 우선순위법과 유사한 질의 해를 찾아낸 것에 반해 병렬 타부 탐색은 병렬 유전알고리즘보다 우수한 해를 찾아내는 것을 볼 수 있다. 이것은 병렬 타부 탐색이 병렬 유전알고리즘 보다 효율적으로 전역적 탐색을 하고 있음을 보여주는 것이다. 이 같은 결과는 타부 탐색의 병렬화를 위해 탐색 공간을 프로세서별로 분할하여 탐색하고 일정주기마다 해를 교환하도록 하여, 프로세서마다 특정 영역을 집중적으로 탐색을 하면서 프로세서간에는 다양화를 유지함으로써 타부 탐색의 성능이 향상된 것에 기인한 것이다.

7. 결론 및 향후 과제

본 논문에서는 발전기 기동정지 계획문제에 대해서 타부 탐색을 병렬화 하여 적용함으로써 성능의 향상을 가져올 수 있음을 보여주었다. 발전기 기동정지 계획문제를 해결하기 위한 타부 탐색의 적용방안을 연구하고 또한 타부 탐색의 병렬화를 통한 성능의 향상을 실험을 통하여 보여주었다. 실험 결과 병렬화를 통해 탐색 소요 시간이 단축될 뿐아니라 타부 탐색의 지역적 탐색의 한계를 벗어나 우수한 해를 찾아낼 수 있음을 보였다.

타부 탐색이 지역적 탐색을 신속히 수행하는 장점을 가진 반면 전역적 탐색 능력에 한계가 있는 단점이 있는 것처럼 다른 탐색 기법들도 그마다의 장단점이 있다. 따라서 여러 탐색 방법의 결합에 의한 탐색 성능 향상을 위한 많은 연구가 있었다. 타부 탐색을 병렬화하여 전역적 탐색이 가능해졌지만 다른 알고리즘과의 결합을 통해 탐색 성능을 더욱 향상시키는 방안을 연구할 필요가 있다. 향후 과제로서 타부 탐색의 지역적 빠른 수렴성과 유전알고리즘의 전역적 탐색능력을 결합한 하이브리드 탐색을 병렬화하여 구현함으로써 그 성능을 검증해보고자 한다.

참 고 문 헌

[1] F. Glover and M. Laguna, Tabu search, Kluwer Academic Publishers, 1997.

[2] R. M. Burns and C. A. Gibson, Optimization of Priority List for an Unit Commitment Program," IEEE Power Engineering Society Meeting, paper no. A75453-1, 1975.

[3] F. Zhuang and F. D. Galiana, " Towards a More Rigorous and Practical Unit Commitment by Lagrangian Relaxation," IEEE Trans. on Power System, PWRS-3, No. 2, pp. 763-773, 1988.

[4] F. Zhuang and F. D. Galiana, "Unit Commitment by Simulated Annealing," IEEE Trans, PWRS-5, No. 1, pp. 311-317, 1990.

[5] A. R. Hamdam and K. Mohamed-Nor, "Integrating an Expert System into a Thermal Unit Commitment Algorithm," IEE Proc. Pt. C. Center., Transm. & Distrib., Vol. 138, No. 6, pp. 553-559, 1991.

[6] Ouyang and S. M. Shahidehpour, "A Hybrid Artificial Nueral Network-Dynamic Programming Approach to Unit-Commitment," IEEE Trans. on Power System, PWRS-7, No. 1, pp. 236-242, 1992.

[7] D. Dasgupta and D. R. McGregor, "Thermal Unit Commitment using Genetic Algorithms," IEE Proc. Gener. Trans. & Dist., Vol. 141, No. 5, pp. 459-465, 1994.

[8] S. A. Kazarlis, A. G. Bakirtzia and V. Petridis, "A Genetic Algorithm Solution to the Unit Commitment Problem," IEEE Trans. on PWRS, Vol. 11, No. 1, pp. 83-92, 1996.

[9] K. J. Mun, H. S. Kim, J. H. Park, T. W. Park, S. H. Park, K. R. Ryu, S. H. Chung, A Parallel Genetic Algorithm for the Unit Commitment Problem, Proceedings of the International Conference on Intelligent System Application to Power Systems (ISAP-97), pp. 188-193, Seoul, Korea, 1997.

[10] T. G. Crainic, M. Toulouse and M. Gendreau, Synchronous Tabu Search Parallelization Strategies for Multicommodity Location-Allocation with Balancing Requirements , OR Spectrum, Vol. 17, 1995.

[11] C. Rego and C. Roucairol, A Parallel Tabu Search Algorithm Using Ejection Chains for the VRP, Proceedings of the Metaheuristics International Conference , Breckenridge, Colorado, pp. 253-259, 1995

[12] P. Badeau, M. Gendreau, F. Guertin, J. Y. Potvin and E. Taillard, A parallel tabu search heuristic for the vehicle routing problem with time windows, Transportation Research-C5,pp. 109-122, 1997.

[13] M. Toulouse, T. Crainic and M. Gendreau, Communication Issues in Designing Cooperative Multi-Thread Parallel Searches, Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, Norwell MA, pp. 501-522, 1996.

[14] A. J. Wood and B. F. Wollenberg, Power Generation, Operation, and Control, Wiley- Interscience, 1996.

[15] D. E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison- Wesley, 1989.

이 용 환

1999년 부산대학교 컴퓨터공학과 학사. 2001년 부산대학교 컴퓨터공학과 석사. 2001년 3월 ~ 현재 부산대학교 컴퓨터공학과 박사과정. 관심분야는 인공지능, 최적화, 일정계획

황 준 하

1995년 부산대학교 컴퓨터공학과 학사. 1997년 부산대학교 컴퓨터공학과 석사. 2002년 부산대학교 컴퓨터공학과 박사. 2002년 9월 ~현재 금오공과대학교 컴퓨터공학부 전임강사. 관심분야는 인공지능, 최적화, 일정계획, 기계학습

류 광 렬

1979년 서울대학교 전자공학과 학사. 1981년 서울대학교 전자공학과 석사. 1983년 3월 ~ 1984년 8월 충북대학교 컴퓨터공학과 전임강사. 1992년 University of Michigan 전기 및 컴퓨터공학과박사. 1992년 3월 ~ 1993년 2월 Scientific Research Lab., Ford Motor Company, 선임연구원. 1993년 3월 ~ 현재 부산대학교 컴퓨터공학과 부교수. 관심분야는 인공지능, 기계학습, 최적화, 정보검색

박 준 호

1978년 서울대학교 전기공학과 학사. 1980년 서울대학교 전기공학과 석사. 1987년 서울대학교 전기공학과 박사. 현재 부산대학교 전자전기정보컴퓨터공학부 교수. 관심분야는 전력시스템 최적 운용, 계획, 지능시스템, 병렬처리