# 발견적 해법을 이용한 분산 컴퓨터 시스템 설계
# - A Heuristic for the Design of Distributed Computing Systems -

손 승 현*
Son, Seung Hyun
김 재 련**
Kim, Jae Yearn

## Abstract

Geographically dispersed computing system is made of computers interconnected by a telecommunications network. To make the system operated efficiently, system designer must determine the allocation of data files to each node. In designing such distributed computing system, the most important issue is the determination of the numbers and the locations where database files are allocated. This is commonly referred to as the file allocation problem(FAP)[3].

The proposed model is a 0/1 integer programming problem minimizing the sum of file storage costs and communication(query and update) costs.

File allocation problem belongs to the class of NP-Complete problems. Because of the complexity, it is hard to solve. So, this paper presents an efficient heuristic algorithm to solve the file allocation problem using Tabu Search Technique.

By comparing the optimal solutions with the heuristic solutions, it is believed that the proposed heuristic algorithm gives good solutions. Through the experimentation of various starting points and tabu restrictions, this paper presents fast and efficient method to solve the file allocation problem in the distributed computing system.

## 1. 서론

과거에는 중앙 사이트에서 데이터를 처리,저장, 및 검색하는 중앙 집중 컴퓨터 시스템(Centralized Computer System)을 운영하여 왔다. 이러한 시스템은 일괄적으로 중앙 집중 컴퓨터에서 프로그램과 데이터 등을 처리하기 때문에 중앙 사이트에서의 통신과 처리는 모두 시스템 병목(Bottle-Neck)이 되어, 정보를 얻고자 할 때 접근 속도가 떨어지며, 데이터 파일의 가용도(Availability) 역시 떨어지게 된다. 이러한 문제점들에 대한 대안으로서 분산 컴퓨터 시스템(Distributed Computing System)이 부각되었다.

분산 컴퓨터 시스템은 통신 네트워크(Communication Network)로 연결된 지역적으로 널리 위치하고 있는 Computing 노드들의 집합으로 이루어져 있으며, 각각의 노드에서는 필요한 프

* 한양대학교 산업공학과 석사과정
** 한양대학교 산업공학과 교수

로그램은 물론 데이터의 처리,저장 및 통신 능력을 가지고 있다[6]. 이러한 분산 컴퓨터 시스템의 장점으로는 가용도 및 용량의 단계적 확장 용이성, 효율성과 융통성 등을 들 수 있다.

분산 컴퓨터 시스템에서 시스템 설계자는 네트워크 상의 여러 노드에 데이터 파일을 할당하게 된다. 합리적인 데이터 파일 정책(File Allocation Policy)은 시스템 운용 비용과 통신 비용의 절감 등의 경제적 이익은 물론이고, 온 라인(On-Line) 상에서의 반응시간(Response Time)의 단축 및 시스템의 가용도를 높여주게 된다[4].

본 연구에서는 이러한 합리적인 데이터 파일 배정(File Allocation Problem ; FAP)에 관한 연구를 하고자 한다. 본 논문에서 제시된 분산 컴퓨터 시스템에서는 다음의 세 가지 사항을 결정하게 된다.

1) 시스템 안에서의 각각의 데이터 파일에 대한 복사본의 개수.
2) 각각의 데이터 파일이 저장되는 노드 선정.
3) 요구되어 지는 데이터 파일이 그 지역에서 처리되지 못할 때, 그 데이터 파일을 다른 노드에서 가져올 때의 경로 선정.

위 사항을 바탕으로 다음과 같은 설계 목적을 가지게 된다.

낮은 시스템 운용 비용(파일 저장 비용 과 조회 및 갱신 통신 비용)
온 라인 상에서의 빠른 응답 시간
데이터 파일의 가용도를 높임

분산 컴퓨터 시스템에서 데이터 배정에 관한 문제를 푸는 방법으로는 최적해를 구하는 방법과 발견적(Heuristic) 기법으로 구하는 방법이 있다. 화일 배정 문제의 계산량은 일반적으로 NP-Complete문제로 알려져 있다. 따라서 파일 배정 문제를 최적해 기법으로 구하는 데에는 많은 어려움이 있다. 특히, 문제 크기가 큰 경우에는 최적해를 구하는 기법으로는 구하지 못하기 때문에 발견적 기법의 개발이 필요하게 된다.

본 연구에서는 발견적 기법 중의 하나인 Tabu Search기법을 이용한 앨고리듬을 개발하여 분산 컴퓨터 시스템에 맞는 보다 효율적인 방법을 제시한다.

## 2. 수리적 모형

### 2.1 가정

분산 컴퓨터 시스템 하에서의 파일 배정 문제를 수리적으로 정식화 하기 위한 가정은 다음과 같다.

1) 프로그램 파일은 모든 노드에 존재한다고 가정한다.
2) 질의가 발생 하였을 때, 발생한 노드에서 처리하지 못할 때는 필요한 파일 복사본이 있는 일정한 노드로만 전달되고 그 곳에서 처리 된다고 가정한다.
3) 데이터 파일의 조회 및 갱신은 모두 실시간(On-Line)으로 이루어 진다고 가정한다.
4) 각각의 노드에서 컴퓨터가 조회나 갱신을 처리하는 시간은 0이라고 가정한다.
5) 장거리 통신 지연이 실시간(On-Line) 응답 시간의 주요 결정 요인이라고 가정한다.
6) 각각의 노드 쌍( i-j ) 사이의 경로 정책은 이미 세워졌으며, 네트워크 상에서의 각각의 통신 채널 능력도 알고 있다고 가정한다.
7) 통신 라인에 의해 연결되어 있는 모든 노드 사이의 최악의 응답 시간 및 파일 가용도(Availability)를 알고 있다고 가정한다. 최악의 응답 시간에 대한 추정치는 모의실험이나 과거의 경험으로 부터 얻을 수 있다.

### 2.2 기호 정의

기호를 설명하면 다음과 같다.

$N$ = 네트워크 상에서의 총 노드의 개수.

$F$= 총 데이터 파일의 개수.

$L$ = 네트워크 상에서의 총 링크의 개수.

$C^l$ =링크 l의 능력. ( kilobits/sec)

$P^l$ = 링크 l을 요구하는 두 노드 i-j의 집합.

$Q_i^d$ = 파일 d에 대한 노드 i에서의 조회량. ( Kilobits)

$\bar{Q}_i^d$= 최고 부하 시간(Peak Load Period)동안 파일 d에 대한 노드 i에서의 조회량. (Kilobits/sec)

$U_i^d$= 파일 d에 대한 노드 i에서의 갱신량. (Kilobits)

$\bar{U}_i^d$= 최고 부하시간(Peak Load Period)동안 파일 d에 대한 노드 i에서의 갱신량. (Kilobits/sec)

$t_{ij}$ = 노드 i와 j사이의 통신비용. ( $/Kilobits)

$w_{ij}$ = 노드 i와 j사이의 최악의 응답시간. (sec)

$a_{ij}$ = 노드 i와 j사이에 성공적으로 통신할 확률.

$T_i^d$ = 파일 d에 대한 노드i에서의 조회에 대한 최대 허용 응답시간. (sec)

$A_i^d$ = 파일 d에 대한 노드 i에서의 조회에 대한 최소 허용 파일 가용도.

결정변수(Decision Variables)

$X_i^d$ = 1 : 파일 d의 복사본이 노드 i에 보관되어 있는 경우.
= 0 : 그렇지 않은 경우.

$Y_{ij}^d$ = 1 : 노드 i에서 발생한 파일 d에 대한 조회(Query)가 노드 j로 전송되는 경우.
= 0 : 그렇지 않은 경우.

2.3 수리 모형

파일 보관 비용 : $$Z1 = \sum_{j=1}^{N} \sum_{d=1}^{F} C_j^d * X_j^d \quad (1)$$

조회 통신 비용 : $$Z2 = \sum_{i=1}^{N} \sum_{d=1}^{F} \sum_{j=1}^{N} Q_i^d * t_{ij} * Y_{ij}^d \quad (2)$$

갱신 통신 비용 : $$Z3 = \sum_{i=1}^{N} \sum_{d=1}^{F} U_i^d * (\sum_{j=1}^{N} t_{ij} * X_j^d) \quad (3)$$

목적식은 다음과 같다.

$$Z = Z1 + Z2 + Z3$$

수식을 단순히 하기 위해 $Cx_j^d, Cy_{ij}^d$ 를 정의한다.

$$Cx_j^d = C_j^d + \sum_{i=1}^{N} U_i^d * t_{ij} \qquad \forall j, d$$

$$Cy_{ij}^d = Q_i^d * t_{ij} \qquad \forall i, j, d$$

따라서, 목적식은 다음과 같이 된다.

$$Z=\sum_{j=1}^{N}\sum_{d=1}^{F}(Cx_j^d * X_j^d)+\sum_{i=1}^{N}\sum_{d=1}^{F}\sum_{j=1}^{N}(Cy_{ij}^d * Y_{ij}^d) \quad (4)$$

본 연구에서의 0-1 정수 계획 모형은 다음과 같다.

Minimize : Z

subject to :

$$\sum_{j=1}^{N} Y_{ij}^d = 1 \qquad \forall i, d \quad (5)$$

$$X_j^d - Y_{ij}^d \geq 0 \qquad \forall i, d, j \quad (6)$$

$$\sum_{j=1}^{N} Y_{ij}^d * w_{ij} \leq T_i^d \qquad \forall i, d \quad (7)$$

$$\sum_{j=1}^{N} Y_{ij}^d * a_{ij} \geq A_i^d \qquad \forall i, d \quad (8)$$

$$\sum_{ij \in P^l}(\bar{Q}_i^d * Y_{ij}^d + \bar{U}_j^d * X_j^d) \leq C^l \qquad \forall i \qquad l \quad (9)$$

$$X_i^d, Y_{ij}^d \in \{0,1\} \quad (10)$$

제약식 (5)는 파일 d에 대한 노드 i에서의 모든 질의 및 갱신 요구를 만족시키고 제약식(6)은 파일d에 대한 복사본이 노드j에 있을 경우에만 노드 i에서 발생한 조회가 노드 j로 보내도록 한다. 제약식 (7)은 파일 d에 대한 노드 i에서의 조회에 관한 최악의 응답 허용 시간을 최대 허용시간 이하로 보장한다. 제약식(8) 역시 파일 d에 대한 노드 i에서의 파일 가용도가 최소 가용도 이상을 보장하도록 해준다. 마지막으로, 제약식(9)는 링크 용량이 초과되지 않도록 보장해 주며, 제약식 (10)은 결정변수가 0 또는 1의 값을 가지도록 한다.

본 모델에서는 Ghosh와 Murthy[3]의 모형에서 고려하지 않은 최고 부하 시간(Peak Load Period)때의 링크 요량을 초과 하는 지에 대한 요소로 조회량(Query Volume) 외에 갱신량(Update Volume)까지도 고려 함으로서 보다 현실에 맞는 모델을 제시하였다.

## 3 . 해법 개발

제3장에서는 제2장에서 제시한 수리적 모형에 Tabu Search 기법을 적용하여 문제를 푸는 방법을 제시한다.

### 3.1 Tabu Search의 기본 개념

Tabu Search는 Glover[2]에 의해 처음으로 연구되기 시작한 일종의 탐색 기법으로 모형화가 쉬우며 조합 최적화 문제에서 빠른 시간에 근사 최적 해를 찾는데 접합한 것으로 알려져 있다. 이러한 Tabu Search 기법은 언덕 오름 탐색 전략(Hill-Climbing Search Strategy)을 사용하여 국부 최적해(Local Optimum)에서 벗어나고 해의 순환(Cycling)을 방지 할 수 있는 기법이다. 이는 해를 탐색한 과정을 기억시켜 다시 되돌아 가는 것을 금지(Tabu)시킴으로써 가능하다. Tabu Search를 특정 문제에 적용할 때, 해 공간의 탐색 성능은 현재해로 부터 이웃해들을 생성하는 이동(Move) 방법, 한번 갔던 해 공간은 일정 기간 동안은 찾아가지 못하도록 하는 Tabu 목록 크기(Tenure), Tabu로 지정된 이동을 해제 할 수 있는 열망 수준 등에 영향을 받게 된다. 해를 찾아갈 때에는 현재 해의 이웃해이며 Tabu 목록에 포함되어 있지 않은 모든 이웃해 중 운영 비용을 최소화 시키는 해를 선택한다. 그리고 그 해는 Tabu 목록에 첨가 시킨다. 또한 Tabu목록에 있다고 하더라도 열망 수준을 만족한다면 더 좋은 해가 나올 가능성이 있기 때문에 이 해 역시 새로운 해가 될 수 있도록 한다. 이러한 과정을 주어진 횟수 동안 반복 실시하여 해를 찾아가는 기법이 바로 Tabu Search의 기본 개념이다.

### 3.2 Tabu Search를 이용한 효율적 앨고리듬 개발

Tabu Search는 초기 가능해를 생성하여 이를 현재해와 최선해로 두고, 이로부터, 이로부터 이웃해 집합을 생성하여 생성된 이웃해 집합이 Tabu목록에 없거나 Tabu목록에 있지만 열망 수준을 만족하는 이웃해에서 가장 좋은 해를 새로운 현재해로 하고 , 이 현재해를 Tabu 목록에 일정 기간 동안 저장 하면서 이 과정을 종료 조건이 만족할 때까지 반복하는 기법이다.
따라서, 분산 컴퓨터 시스템의 할당 문제에 Tabu Search에서 사용되는 이웃해를 생성하고 초기해 설정, Tabu 목록 및 열망 수준을 적용하여야 한다.

3.2.1이웃해의 선정

Tabu Search에서는 이웃해를 생성하는 방법으로 이동 방법을 많이 사용한다. 이동 방법으로는 흔히 해의 원소를 교환,삽입,삭제,첨가 하는 방법을 사용한다. 4가지 이동 방법 중 삽입 이동이 탐색 효율면에서 가장 좋은 것으로 나타났다. 따라서, 본 연구에서는 효과적으로 이웃해를 생성해 가는 삽입 이동 방법을 사용한다. 그림으로 설명하면 다음과 같다.

노드

| 파일 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 1 |
| 4 | 1 | 1 | 1 | 0 | 0 |
| 5 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 1 | 1 |
| 7 | 1 | 0 | 1 | 0 | 0 |

<그림 3.1>데이터 파일이 각 노드에 할당된 예

노드

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 |

<그림 3.2>파일1의 조회 경로의 예

<그림 3.1> 을 파일 배정의 초기 해로 가정하면 파일1은 노드 1과 2에 할당을 하게 된다. 먼저 파일1에 대해서 삽입 방법을 사용하여 노드1에 있던 1번 파일을 파일1이 없는 3,4,5번 노드로 차례대로 옮겨보고 운영 비용을 계산하여 본다. 만약 비용이 감소 한다면 파일 할당 상태를 계속 바꾸어 준다. 다음 파일1에 대해서 원 상태로 바꾸어 준 다음, 나머지 파일2 - 7번에 대해서도 같은 방법으로 비용을 구한다. 7개의 파일 중 가장 비용을 많이 감소시키는 파일을 현재 해로 선택하고 비용을 구하게 된다. 그리고 이때의 현재해를 Tabu 목록에 기록하게 된다.

데이터를 각 노드에 할당할 때에 우리는 이와 동시에 조회하고자 하는 파일이 그 노드에 없을 시, 그 파일이 있는 노드로 찾아가 정보를 얻기 위한 조회 경로를 결정해야 한다. 예를 들어 < 그림 3.2 > 에서는 파일1이 노드 1과2에 있을 경우, 3번 노드에서는 1번 노드로, 4,5번 노드에서는 2번 노드로 1번 파일 조회를 요구하는 경로를 보여주고 있다.

3.2.2 초기해의 선정

본 연구에서는 다음의 3가지 방법을 사용하여 초기해를 구하게 된다.

1) 임의로(Random) 데이터 파일을 각각의 노드에 배치하는 방법.
2) 각 노드에서 가장 조회를 많이 하는 데이터 파일을 먼저 그 노드에 놓은 후 다른 노드에 그 파일을 추가 하는 방법. ( 이후 이 방법을 조회를 중심으로 한 초기해 방법이라고 부른다.)데이터 파일의 경우에는 하나의 파일을 여러 노드에 할당 할 수 있으므로 어떤 파일을 추가 했을 때, 운영 비용을 감소 시키면 그 노드에도 파일을 할당하게 된다. 이와 같은 작업을 반복 수행하여 더 이상의 파일 삽입으로 인한 운영 비용의 감소가 나타나지 않을 때에는 데이터 파일 삽입을 중단하게 된다. 그 결과, 초기에 각각의 파일 위치 및 개수가 결정되게 된다.
3) 필요로 하는 파일이 그 노드에 없을 때, 그 파일이 있는 노드로 조회를 하게 된다. 그때, 그

노드에서 요구하는 최대 허용 응답 시간(Maximum Acceptable Response Time) 보다 시간이 더 걸리거나 최소 허용 가용도(Minimum Acceptable File Availability) 보다 가용도가 작은 노드로는 아무리 원하는 파일이 있다고 하더라도 그 노드로 조회 경로를 정할 수 없게 된다. 이러한 사실을 이용하여 각각의 파일에 대해 각 노드에서 조회를 할 수 없는 경로를 미리 계산하여 가장 많이 가지 못하는 노드에 데이터 파일을 가장 먼저 배치한 후 다른 노드에 그 파일을 추가 하는 방법. (이후 이 방법을 응답 시간과 가용도를 중심으로 한 초기해 방법이라고 부른다)

위 방법을 사용하여 제4장에서는 초기해 방법의 효과를 알아본다.

### 3.2.3 Tabu Tenure 및 열망 수준 선정

Tabu Search 기법에서 Tabu 목록의 크기를 결정하는 Tabu Tenure는 문제의 크기와 데이터에 많이 의존하게 된다. Tabu Tenure가 작으면 사이클이 발생하기 쉬우며, 크면 이웃해를 탐색할 때에 해를 탐색할 수 있는 이웃해의 공간이 줄어들게 된다. 따라서, 제4장에서는 예제를 통하여 각 문제에 대한 효과적인 Tabu Tenure를 결정 하고자 한다.

열망 수준은 Tabu 목록에 있다고 할지라도 지금까지 나온 해 중에서 가장 좋은 해(Best Solution), 즉 최선해 보다 좋은 해가 나오면 이 해를 새로운 현재해로 받아들이는 것으로 결정한다.

### 3.2.4 앨고리듬의 단계

본 앨고리듬의 구체적인 절차는 다음과 같다.

단계 1. ( 초기화 )

1) Tabu Tenure, 열망 수준, 종료 조건등을 결정한다.
2) 초기해 방법에 따라 파일을 여러 노드에 할당한 후, 가능 한 조회경로를 비교하여 가장 목적식 값을 작게 하는 조회 경로를 조회 경로로 정한다. 이것을 현재해와 최선해로 둔다.
3) Tabu 목록을 초기화 시킨다.

단계 2. ( 이웃해 생성 )

현재해로 부터 Tabu 목록에 없거나 Tabu 목록에 있지만 열망 수준을 만족하는 이웃해를 생성한다.

단계3. ( 현재해와 최선해 수정)

1) 생성된 이웃해 중에서 가장 좋은 해를 현재해로 한다.
2) 현재해가 최선해 보다 좋으면 현재해를 최선해로 둔다.

단계 4. ( Tabu 목록 수정 )

새로운 현재해를 Tabu 목록에 기록한다. Tabu 목록이 다 차게 되면 가장 먼저 기록된 현재해를 삭제하게 된다.

단계 5. ( 반복 및 해의 출력 )

주어진 횟수 동안 단계 2,3,4를 반복하여 해를 개선 시키고 가장 좋은해를 출력하고 끝내게 된다.

다음 제4장에서는 예제들을 통하여 본 장에서 제시한 앨고리듬의 효과를 알아보도록 한다

## 4. 수치 예제

본 4장에서는 3장에서 제안한 Tabu Search 기법에 대한 효과를 예제를 통하여 알아본다. 예제에서 사용되는 비용요소 및 제반요소는 아래의 방법에 의해 발생 시킨다.

질의 통신비용( $t_{ij}$ ; \$/Kilobyte) - Uniform (0.01-0.06)

파일 보관비용( $C_i^d$ ; $/Kilobyte) - Uniform (20-40)

통신링크 용량( $C^l$ ; Kilobyte/sec) - Uniform (15-40)

조회량(Volume) ( $Q_i^d$ ; Kilobyte ) - Uniform(10,000-50,000)

갱신량(Volume) ( $U_i^d$; Kilobyte ) - 질의량 의 30-50%

최고부하시간의 질의량( $\bar{Q}_i^d$ ; Kilobyte ) - Uniform(0-15)

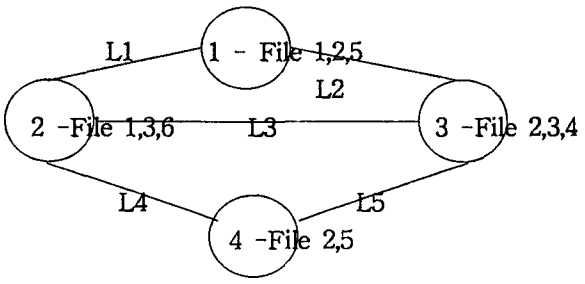
최고부하시간의 갱신량( $\bar{U}_i^d$; Kilobyte ) - Uniform(0-8)

예제는 총 5문제로 이루어져 있다.
노드 4개, 파일6개, 링크5개인 경우 문제 A
노드 5개, 파일5개, 링크8개인 경우 문제 B
노드 6개, 파일4개, 링크13개인 경우 문제 C
노드 10개, 파일12개, 링크 44개인 경우 문제 D
노드 20개, 파일18개,링크 89개인 경우 문제 E

문제 A,B,C는 비교적 크기가 작은 문제이며,문제D는 중간크기,문제E는 큰 문제로 선택하였으며, 작은 문제에서는 Lindo 프로그램을 통하여 Optimal값을 구하여 본 앨고리듬과 비교를 하였다.

다음은 문제 A의 네트워크 모습이다. 먼저 문제 A를 가지고 데이터 할당 및 조회 경로의 예를 들어 보았다.

예) ( File = 6, Node = 4, Link 5 )




< 그림 4.1 > 문제 A에 파일이 할당된 예(I)

| | | 노드 | | | |
|---|---|---|---|---|---|
| | | 1 | 1 | 3 | 4 |
| 파일 | 1 | 1 | 1 | 0 | 0 |
| | 2 | 1 | 0 | 1 | 1 |
| | 3 | 0 | 1 | 1 | 0 |
| | 4 | 0 | 0 | 1 | 0 |
| | 5 | 1 | 0 | 0 | 1 |
| | 6 | 0 | 1 | 0 | 0 |

<그림 4.2>문제A의 파일 할당 예(II)

| | | 노드 | | | |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 |
| 노드 | 1 | 1 | 0 | 0 | 0 |
| | 2 | 0 | 1 | 0 | 0 |
| | 3 | 1 | 0 | 0 | 0 |
| | 4 | 0 | 1 | 0 | 0 |

<그림 4.3>파일1의 조회경로의 예

4.1 해법 효율 분석

본 연구의 앨고리듬은 C언어로 코드화 되어 IBM-PC 호환기종 Pentium-120, Ram 32M상에서 수행되었다.

< Table 4.6 > Optimal 값과 Heuristic값과의 비교 분석

| | Optimal 값 | | Heuristic 값 | | |
|---|---|---|---|---|---|
| | 운영 비용($) | 시간(sec) | 운영 비용($) | 시간(sec) | 오차(%) |
| 문제 A | 17047.61 | 13 | 17047.61 | 0.76 | 0 |
| 문제 B | 22875.02 | 23 | 22924.67 | 1.76 | 0.2 |
| 문제 C | 23636.67 | 111 | 23727.61 | 2.3 | 0.38 |
| 문제 D | | | 65236.34 | 874.69 | |
| 문제 E | | | 143256.34 | 1978.84 | |

본 결과에서 알 수 있듯이 문제A에서는 Tabu Search 기법을 이용한 해법이 최적해와 같은 값을 구하였으며, 문제B와 C에서도 적은 오차만을 가진, 비교적 좋은 해를 구하였다. 따라서 본 연구에서 제시한 앨고리듬은 문제 크기가 큰 경우에도 원하는 시간에 최적해에 가까운 값을 구할 수 있을 것이다. 여기서 제시한 앨고리듬의 값들은 다음 절에서 비교 분석을 통해 얻어진 가장 좋은 조건을 기준으로 하여 값을 구하였다 (예를 들어 문제A의 경우, 초기해는 조회를 중심으로 한 초기해로, Tabu Tenure는 3으로 하여 문제를 풀었다.)

4.2초기해의 변화

* 초기해 변화에 따른 비교 분석 *

초기해 방법 중, Random하게 한 초기해는 여러 번 반복을 수행하여 비용과 시간을 평균을 내서 값을 구하였다.( 문제 A,B,C : 50번 반복 수행, 문제 D,E : 10번 반복 수행 ) 본 연구 실험들에서의 종료 방법은 100번의 반복 횟수로 정하였다.

문제 A

| 초기해 방법 | 운영 비용($) | | 시간(sec) |
|---|---|---|---|
| | 초기해 | 최선해 | |
| Random하게 한 초기해 | 22796.19 | 20917.03 | 1.51 |
| 조회를 중심으로 한 초기해 | 20910.05 | **17047.61** | 0.76 |
| 응답시간과 가용도를 중심으로 한 초기해 | 22706.12 | 17252.34 | 1.0 |

문제 B

| 초기해 방법 | 운영 비용($) | | 시간(sec) |
|---|---|---|---|
| | 초기해 | 최선해 | |
| Random하게 한 초기해 | 31031.54 | 23767.02 | 1.88 |
| 조회를 중심으로 한 초기해 | 27989.02 | **22924.67** | 1.76 |
| 응답시간과 가용도를 중심으로 한 초기해 | 27979.42 | 23048.41 | 1.77 |

문제 C

| 초기해 방법 | 운영 비용($) | | 시간(sec) |
|---|---|---|---|
| | 초기해 | 최선해 | |
| Random하게 한 초기해 | 31584.49 | 24818.27 | 2.4 |
| 조회를 중심으로 한 초기해 | 25645.69 | **23727.61** | 2.3 |
| 응답시간과 가용도를 중심으로 한 초기해 | 28782.84 | 23869.87 | 2.1 |

비교적 작은 문제 A,B,C에 대한 초기해를 달리 했을 경우의 해의 값 및 시간을 조사해 본 결과 위와 같은 결과를 얻을 수 있었다. 3문제 모두 조회를 중심으로 초기해를 구한 방법에서

가장 좋은 최선해를 얻을 수 있었으며, 시간상으로도 상대적으로 빠른 결과를 얻을 수 있었다. 다음으로 Random하게 초기해를 준 경우 보다는 응답 시간과 가용도를 중심으로 한 초기해를 사용한 방법이 더 효과적인 것으로 나타났다.

문제 D

| 초기해 방법 | 운영 비용($) | | |
|---|---|---|---|
| | 초기해 | 최선해 | 시간(sec) |
| Random하게 한 초기해 | 113659.53 | 73645.56 | 1023.70 |
| 조회를 중심으로 한 초기해 | 80256.71 | 66788.58 | 874.69 |
| 응답시간과 가용도를 중심으로 한 초기해 | 81356.47 | **65236.34** | 890.46 |

문제 E

| 초기해 방법 | 운영 비용($) | | |
|---|---|---|---|
| | 초기해 | 최선해 | 시간(sec) |
| Random하게 한 초기해 | 180342.82 | 146342.58 | 2300.65 |
| 조회를 중심으로 한 초기해 | 173256.41 | **143256.34** | 1978.84 |
| 응답시간과 가용도를 중심으로 한 초기해 | 175367.47 | 144238.23 | 1883.23 |

비교적 큰 문제에 있어서도(문제 D, E) 조회를 중심으로 한 초기해 및 응답 시간과 가용도를 중심으로 한 초기해가 Random하게 초기해를 준 경우보다는 좋은해를 구할 수 있었다.

결국, 본 실험을 통하여 초기값을 Random하게 주고 시작하는 방법보다는 보다 최적해에 근접하도록 초기값을 구한 다음에( 본 실험에서는 조회를 중심으로 한 초기값에서 가장 좋을 값을 얻을 수 있었다. ), 값을 구하는 방법이 시간 및 해의 질에 있어서 우수함을 알 수 있었다.

### 4.3 Tabu Tenure의 변화

* Tabu Tenure 변화에 따른 비교 분석 *

| Tabu Tenure | 문제 A | 문제 B | 문제 C |
|---|---|---|---|
| | 운영 비용($) | 운영 비용($) | 운영 비용($) |
| 2 | **17047.61** | **22924.67** | **23727.61** |
| 3 | **17047.61** | **22924.67** | 24013.24 |
| 4 | 18034.23 | 24253.67 | * |
| 5 | 18646.34 | * | * |

문제 A와 B에서는 Tabu Tenure가 2 와 3에서 가장 좋은 값을 구하였으며 문제 C는 Tabu Tenure를 2로 했을때 가장 좋은 값을 구할 수 있었다.

| Tabu Tenure | 문제 D | 문제 E |
|---|---|---|
| | 운영 비용($) | 운영 비용($) |
| 2 | 68362.41 | * |
| 3 | * | 164254.26 |
| 4 | 66342.35 | * |
| 6 | **65236.34** | **143256.34** |
| 8 | 69243.04 | 147347.56 |
| 12 | * | 144863.03 |
| 15 | * | 145032.38 |

문제 D와 E에서는 모두 Tabu Tenure가 6에서 가장 좋은 값을 얻을 수 있었다.
위 그림에서 알 수 있듯이 문제 크기가 큰 경우에는 Tabu Tenure의 변화에 따른 값의 변화가 문제 크기가 작은 경우 보다는 큰 것으로 나타났다.

## 5. 결론

본 연구에서는 발견적 기법의 하나인 Tabu Search 기법을 사용하여 분산 컴퓨터 시스템의 할당 문제에 어떻게 적용할 지에 대해 연구하였다. 본 연구에서 제시된 모형은 비용 요소(데이터 파일 저장 비용 및 통신 비용) 뿐만 아니라 비용 요소 외의 다른 요소(응답 시간 및 가요도)를 고려한 보다 현실성 있는 모형이다.

연구의 결과, 크기가 작은 문제(문제 A,B,C)에서는 최적 값(Optimal Value)과의 비교를 통하여, 본 연구에서 제시한 앨고리듬이 최적해 혹은 최적해에 가까운 해를 구할 수 있었고,큰 문제에서도 쉽게 적용할 수 있음을 알 수 있었다. 또한 초기값의 변화에 따른 해의 값과 시간을 비교한 결과, 초기값은 Random하게 시작하는 경우 보다는 먼저 조회가 가장 많이 일어나는 곳에 파일을 배정한 후, 운영비용의 변화를 살펴가며 계속해서 파일을 삽입하는 방법이 시간과 해의 질에 있어서 우수함을 알 수 있었다. 응답 시간과 가용도를 중심으로 하여 초기값을 구하는 방법 역시 조회를 중심으로 초기값을 구하는 방법 다음으로 우수한 해를 얻을 수 있었다.

Tabu Tenure 비교를 통하여, 작은 문제에서는 비교적 Tenure의 영향을 많이 받지 않지만 문제가 커질수록 Tenure의 영향을 많이 받음을 알 수 있었다.

본 연구에서 제시한 앨고리듬은 다른 파일 배정 문제에도 쉽게 적용 가능하며, 특히 초기해의 선정 방법 및 Tenure 결정 등에 있어서는 많은 도움을 줄 수 있으리라 생각된다. 앞으로는 이번 연구에서 비교하지 않은 큰 문제에 대한 다른 발견적 기법들과의 비교 연구가 요구된다.

## 참 고 문 헌


1. 홍진원, "Tabu 탐색 기법을 이용한 분산 컴퓨터 시스템 설계", 석사 학위 논문, 한양대학교 대학원,서울, 1995.
2. Ciolin R. Reeves BSc. Mphil, *Modern Heuristic Techniques for Combinatorial Problems,* Coventry University, New York, 1993.
3. Deb Ghosh and Ishwar Murthy, "A Solution Procedure for the File Allocation Problem with File Availability and Response Time", *Computers Ops Res,* Vol.18, No. 6, pp. 557-568,1991.
4. Heeseok Lee and Olivia R. Liu Sheng, "A Multiple Criteria Model for the Allocation of Data Files in a Distributed Information System", *Computers Ops Res,* Vol. 19, No. 1, pp. 21-33, 1992.
5. Hemant K. Jain, "A Comprehensive Model for the Design of Distributed Computer Systems", *IEEE Transactions on Software Engineering,* Vol. SE-13, No. 10, pp. 1092-1104, October 1987.
6. Morgan, H. I. And K.D. Levin, "Optimal Program and Data Locations in a Computer Networks", *Communications of the ACM,* Vol. 20, No 5, pp. 315-321, 1977.
7. Ranamoothy C.V. and B.B. Wha, "The Isomorphism of Simple File Allocation," *IEEE Transactions on Computers,* Vol. C-32, No. 3, pp.221-231, 1983.