□ 硏 究 論 文 □

韓國國防經營分析學會誌

제 30 권, 제 1 호, 2004. 6 30.

네트워크 전환문제에 대한 타부 탐색 해법 (A Tabu Search Algorithm for the Network Diversion Problem)

양 희 원, 박 성 수*

Abstract

This research considers a Network Diversion Problem (NDP) in the directed graph, which is to identify a minimum cost set of links to cut so that any communication paths from a designated source node to a destination node must include at least one link from a specified set of arcs which is called the diversion arcs. We identify a redundant constraint from an earlier formulation. The problem is known to be NP-hard, however a detailed proof has not been given. We provide the proof of the NP-hardness of this problem. We develop a tabu search algorithm that includes a preprocessing procedure with two steps for removing diversion arcs as well as reducing the problem size. Computational results of the algorithm on instances of general graphs and grid graphs are reported.

(**Keyword** : network diversion, NP-hard, tabu search)

* 한국과학기술원 산업공학과

1. 서 론

1.1 연구 배경

네트워크 전환문제(Network Diversion Problem, 이하 NDP)란 출발점에서 도착점으로의 흐름이 반드시 특정 구간들 중 최소 한 곳 이상을 지나도록 하기 위해 최소 비용으로 다른 구간들을 차단하는 문제이다. 여기서 특정 구간에 대응되는 호들을 전환호(diversion arc)라 한다.

통신 네트워크의 중요함이 대두되고 있는 현재 상황에서 우리가 원하는 대로 네트워크 흐름의 통제를 가능하게 만드는 기술은 네트워크 운용에 있어 핵심적인 기술이 될 수 있다. 여러 목적들에 의해 흐름이 반드시 특정 구간을 거치도록 해야 하는 경우가 발생할 수 있다. 예를 들어 네트워크 공격자의 입장에서 통신 정보를 획득하고자 할 때 특정 구간으로 모든 흐름이 통과하도록 유도하려는 문제가 야기될 수 있다. 반대로 네트워크 방어자의 입장에서는 그러한 공격에 대한 취약한 구간을 찾아내서 보완하고 해결책을 구하고자 할 때 야기될 수 있는 문제이다. 또한 군사 작전시 이러한 문제가 야기될 수 있다. 예를 들어 게릴라들의 불법적인 무기류, 화학 물질, 마약류 등의 흐름을 차단하는 작전을 수행하고자 할 때 광활한 지역의 모든 곳에 진지를 구축하고 적이 오기만을 기다리고 있는 것은 현실적으로 불가능한 문제이다. 이에 특정한 몇몇 구간을 선택해서 목진지를 구축하고, 이 중에서 최소 한 구간을 적들이 통과하도록 만들 필요성이 제기된다. 이에 따라 출발점에서 도착점까지의 여러 경로들 중에서 구축해 놓은 목진지를 지나지 않는 경로들에 대해서는 어느 한 구간씩을 차단해서 적이 반드시 목진지 중 한 구간을 통과하도록 유도해야 한다. 그러나 지리적 여건에서 볼 때 차단시켜야 할 구간은 교량, 도로, 계곡 등 매우 다양한 구간이 될 수 있으며 많은 어려움을 수반하고 있고, 그에 따른 비용은 막대한 것이 될 수 있기 때문에 비용을 최소화할 필요성이 제기되는 것이다.

1.2 관련 연구

본 연구에서 다루는 네트워크 전환문제는 Curet[9]에 의해 처음 제시되었다. Curet은 이 문제의 복잡성이 NP-hard임을 언급하였으나 구체적인 증명 내용을 기술하지 않았다. Curet은 이 문제를 제약식이 추가된 단절(cut) 문제로 분석하였고, 이에 따른 정수 계획 모형을 제시하였다. Curet은 이 문제를 해결하기 위해 그가 제시한 정수 계획 모형에서 일부 제약식을 완화시킨 라그랑지 완화 기법을 이용하였다.

Erken[10]는 단순히 전환호를 포함시킨 최저 비용의 $s-t$ 단절을 구하는 것으로부터 문제를 해결해 나간다. 그러나 전환호가 포함된 $s-t$ 단절이 바로 가능해가 됨은 보장될 수 없다. 왜냐하면 전환호가 포함된 $s-t$ 단절을 전환호는 제외하고 절단하였을 때 $s-t$ 경로가 존재하지 않는 경우가 발생하기 때문이다. 따라서 단순히 전환호가 포함된 최저 비용의 $s-t$ 단절을 찾은 후 가능해 여부를 확인해서 가능해가 된다면 이는 네트워크 전환문제의 최적해가 되므로 알고리즘을 종료한다. 그러나 비가능해인 경우 $s-t$ 단절을 구성하는 호에 대해 분지 한계법(Branch and Bound) 알고리즘을 적용하여 가능해를 찾는다.

위에서 언급된 연구 외에 네크워크 전환문제에

대한 연구는 이루어지지 않은 상태이며, 이에 따라 메타 휴리스틱으로 문제를 접근한 연구 역시 없는 실정이다.

본 연구에서는 보다 좋은 가능해(feasible solution)를 찾기 위한 방안으로 네트워크 전환문제에 대한 타부 탐색 알고리즘을 제안하려고 한다. 제안된 알고리즘은 사전 처리 단계와 타부 탐색 단계로 구분되며, Curet에 의해 제시된 정수 계획 모형을 토대로 IP Solver를 이용해서 얻은 최적해와의 비교를 통해 제안된 알고리즘의 성능을 평가하고자 한다.

1.3 내용의 구성

본 연구 내용은 다음과 같이 구성된다. 2장에서는 Curet에 의해 제시된 정수 계획 모형에서 불필요한 제약식이 있음을 살펴본다. 또한 Curet이 문제의 복잡성이 NP-hard임을 언급하였으나 구체적 증명 내용이 생략되어 본 연구에서 네트워크 전환문제가 NP-hard임을 증명한다. 3장에서는 이 문제의 가능해의 형상에 대해 살펴보고, 알고리즘의 수행 노력을 줄이기 위한 2 단계의 사전 처리 단계를 제시한 후 타부 탐색 알고리즘을 제시한다. 4장에서는 제안된 알고리즘에 대한 실험 수행 결과에 대해 살펴보고, 마지막으로 5장에서는 결론을 제시한다

2. 문제 정의 및 모형의 분석

2.1 기호와 가정

네트워크 전환문제를 정의하기 위해서는 네트워크와 관련된 정보, 호를 제거하는 데 드는 비용 등의 정보들을 필요로 하며, 보다 구체적인 것은 다음과 같다.

$G=(V,A)$: 네트워크를 나타내는 그래프, $V=\{1,2,...,n\}$는 마디의 집합, A는 유방향호의 집합

D : 모든 $s-t$ 경로가 반드시 최소 하나는 통과해야 하는 전환호의 집합, $\overline{D}=A\backslash D$

c_{ij} : 호 (i,j)를 제거하는 데 드는 비용

s : 출발점

t : 도착점

위와 더불어 $G=(V,A)$는 연결된 그래프이고, 출발점에서 도착점까지의 경로가 존재해야 한다. 또한 D에 속하는 각각의 호를 전환호(diversion arc)라 부른다. 이러한 전환호가 다수일 때 경로상에 어떠한 전환호가 포함되어야 하는지 모르기 때문에 문제는 좀더 어려워진다. 모든 c_{ij}는 호 (i,j)를 공격하거나 파괴하는데 필요한 노력의 양이나 비용을 나타내는 것으로 비음인 정수로 가정한다. 단, 우리는 전환호를 파괴하지 않을 것이기 때문에 전환호에 대한 제거 비용은 없는 것으로 설정한다. 네트워크 전환문제와 관련된 문제는 최저 비용 단절 (minimum cost cut)문제이다. 어떤 마디의 부분집합 X에 대해 한 끝은 X에 있고 다른 한 끝은 $\overline{X}=V\backslash X$ 에 있는 순방향호들의 집합을 $(X:\overline{X})$라고 하고 이를 X에 대한 단절(cut)이라 정의한다. 즉,

$(X:\overline{X})=\{(i,j)\in A \mid i\in X, j\in \overline{X}\}$이 다. 간단히 $s-t$ 단절 $(X:\overline{X})$은 $s\in X$, $t\in \overline{X}$인

단절을 의미한다. 이때 $s-t$ 단절에 속하는 호들을 제거하므로써 출발점에서 도착점으로의 유방향호들의 경로를 차단할 수 있다. 이후부터 언급되는 모든 단절은 $s-t$ 단절로 간주한다. 한편 단절의 비용은 호들의 비용의 합으로 나타낸다. 즉 $C(X:\overline{X}) = \sum_{(i,j)\in(X:\overline{X})} c_{ij}$. 또한 기호 P_{uv}는 마디 u에서 마디 v로의 경로를 의미하고, 어떤 경로 P에 대해 $A(P)$는 P를 구성하는 호들의 집합으로, $V(P)$는 P를 구성하는 마디들의 집합으로 정의한다.

2.2 기존 모형 분석

네트워크 전환문제의 정수 계획 모형은 Curet [9] 에 의해 제시되었다.

모형 설정을 위한 기호의 정의는 다음과 같다.

b_i : 마디 i에서의 흐름 보존 상수, $b_s=1$, $b_t=-1$, $b_i=0$ for $i\neq s,t$

ϵ : 목적식에서의 최단 경로 구성에 대한 가중치

이 모형에서 사용하는 결정 변수는 다음과 같다.

z_{ij} : 호 (i,j) 가 제거되면 1, 그렇지 않으면 0 값을 갖는 이진 변수

y_i : 마디 i에 대해 $i\in\overline{X}$이면 1, 그렇지 않으면 0 값을 갖는 이진 변수

x_{ij} : 호 (i,j)가 s에서 t로의 경로를 따라 흐름을 보내는 데 사용되면 1, 그렇지 않으면 0 값을 갖는 이진 변수

Curet이 제시한 정수 계획 모형은 다음과 같다.

$$\min \sum_{(i,j)\in\overline{D}} c_{ij}z_{ij} + \epsilon \sum_{(i,j)\in\overline{D}} x_{ij}$$

$$s.t \quad y_i - y_j + z_{ij} \geq 0 \qquad \forall (i,j)\in A \tag{1}$$

$$y_t = 1, y_s = 0 \tag{2}$$

$$y_i \in \{0,1\} \forall i \in V \tag{3}$$

$$z_{ij} \in \{0,1\} \forall (i,j) \in A \tag{4}$$

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{k:(k,i)\in A} x_{ij} = b_i \forall i \in V \tag{5}$$

$$x_{ij} \in \{0,1\} \forall (i,j) \in A \tag{6}$$

$$\sum_{(i,j)\in D} z_{ij} \geq 1 \tag{7}$$

$$x_{ij} + z_{ij} \leq 1 \forall (i,j) \in \overline{D} \tag{8}$$

단, $\overline{D} = A \backslash D$

제약식 (1)-(4)는 $s-t$ 단절을 찾아야 함을 의미하며, 제약식 (5)-(6)은 s에서 t로의 경로가 존재하는 것을 보장하게 만든다. 여기서 (1)-(4)의 제약식과 (5)-(6)의 제약식은 서로 상충된다. 따라서 제약식 (8)을 추가하였다. 제약식 (8)은 s에서 t로의 흐름이 호 (i,j)를 사용한다면 호 (i,j)는 단절을 구성하는 호가 될 수 없음을 의미한다. 반대로 호 (i,j)가 단절에 포함된다면 s에서 t로의 흐름이 호 (i,j)를 이용할 수 없게 된다. 위의 제약을 만족시키면서 호의 절단 비용을 최소화시키는 것이 문제의 목적이다. 주의할 점은 $s-t$ 경로를 보장하기 위해 $s-t$ 단절에 포함된 전환호는 절단하지 않는다는 것이다. 따라서 목적식에서 단절 비용을 계산

시 전환호에 대한 비용은 고려하지 않는다. 또한 경우에 따라 길이가 길거나 짧은 $s-t$ 경로의 필요성이 존재한다. 이러한 $s-t$ 경로 길이의 중요성은 가중치 ϵ를 설정하므로써 반영되었다. 그러나 위의 모형에서 (7)번 제약식은 나머지 제약식을 만족시키면 자동적으로 만족되는 제약식으로써 중복된 제약식이 된다. 따라서 이 제약식을 생략할 수 있다.

정리 1. 제약식 (7)은 중복 제약식(redundant constraint)이다.

증명: 유방향 그래프 $G=(V,A)$가 주어졌을 때, 그래프는 단절 제약식에 의해서 $s\in X$, $t\in \overline{X}$인 $(X,E(X))$와 $(\overline{X},E(\overline{X}))$로 양분된다. 여기서, $E(X)=\{(i,j)\mid (i,j)\in A, i\in X, j\in X\}$이다.

이때 단절 $(X:\overline{X})=\{e_1,e_2,...,e_m\}$이라고 하자. 출발점 s에서 도착점 t로의 흐름을 보낸다고 할 때, $s-t$ 단절의 정의상 모든 s에서 t로의 한 단위 흐름은 $(X:\overline{X})$중 하나의 호를 이용해야만 한다. 즉 $x_{e_1}+x_{e_2}+...+x_{e_m}=1$ 성립. 한편 제약식 (8)에 의해 $x_{e_i}+z_{e_i}\le 1$ for $A\backslash D$가 성립되어야 하므로 단절 $(X:\overline{X})$에는 최소한 하나 이상의 D에 속하는 호가 포함되어야 한다. 즉 $\sum_{e\in (X:\overline{X})\cap D} Z_e \ge 1$이 성립되어야 함을 의미한다. 따라서 제약식 (7)은 나머지 제약식을 만족시키면 자동적으로 만족되어 지므로 중복 제약식이다.

2.3 문제의 복잡성

Curet은 The Directed Subgraph Homeomorphism Problem(이하 DSHP)이 일반적인 유방향 그래프에서 NPC에 속한다는 사실에서 본 연구에서 다루고 있는 네트워크 전환문제가 NP-hard임을 주장하였다. 그러나 Curet의 논문에서 구체적인 증명은 다루지 않았으므로 본 연구에서 이에 대한 증명을 제시하고자 한다.

DSHP의 특수한 경우로서 두 개의 서로 만나지 않는 호(disjoint arcs)와 네 개의 인접한 마디를 가지는 고정된 형태 그래프(fixed pattern graph)P에 대한 DSHP는 NP-Complete[11]임이 알려져 있다. 한편 이 문제는 다음과 같은 문제로 재정의 할 수 있다.

Instance : 유방향 그래프 $G_1=(V_1,A_1)$, 특정 마디들의 쌍 $\{(a,b),(c,d)\}$

Question : 그래프 G_1에 마디들을 공유하지 않는 a에서 b로의 경로와, c에서 d로의 경로가 존재하는가?

이후부터는 위의 문제를 2-DSHP라고 언급하겠다. 또한 네트워크 전환문제(NDP)의 decision problem을 정의하면 다음과 같다.

Instance : 유방향 그래프 $G_2=(V_2,A_2)$, 전환호의 집합 $D\subset A_2$, $K\in Z_+^1$

Question : 그래프 G_2에 호의 부분집합 $A'\subset A_2$을 절단했을 때 모든 $s-t$ 경로들이 최소한 하나의 전환호를 통과하도록 만들면서

$|A'| \leq K$를 만족시키는 A'이 존재하는가?

정리2. NDP는 NP-hard이다.

증명 : NDP가 NP-hard임을 증명하기 위해서 NDP의 decision problem이 NP-Complete임을 보인다. 이를 위해서 먼저 NDP의 decision problem이 NP에 속함을 보이고, NP-Complete인 특정 문제에서 NDP의 decision problem으로 다항 변환(polynomial transformation)이 가능함을 보인다.

NDP의 decision problem이 NP에 속함을 보이는 것은 다음과 같다. NDP의 decision problem의 참 값을 갖는 예제(instance) $G_2=(V_2, A_2)$, $D \subset A_2, K \in Z_+^1$가 있다고 하자. 주어진 조건을 만족시키는 호의 집합 $A'(\subset A_2)$이 주어졌을 때 $|A'| \leq K$이 성립하는지 확인하는 과정과 G_2에서 A'을 절단했을 때 모든 $s-t$ 경로가 반드시 최소 한 하나 이상의 D를 구성하는 호를 통과하는지 확인하는 과정이 필요하다. 이를 위해서는 $\tilde{G}_2=(V_2, A_2 \backslash A')$에서 $s-t$ 경로가 존재하는지 확인하고, $\tilde{G}_2=(V_2, A_2 \backslash \{A' \cup D\})$에서 $s-t$ 경로가 존재하지 않는지 확인하는 과정이 필요하다. 위의 과정은 모두 다항 시간 안에 가능하다.

2-DSHP에서 NDP로의 변환(transformation)은 다음과 같다. 먼저 $V_2=V_1$으로 설정하고, G_1에서 호 (b, c)가 존재하면 $A_2=A_1$, 호 (b, c)가 존재하지 않으면 $A_2=A_1 \cup \{(b, c)\}$로 설정한다. G_1의 입력 요소 $\{(a, b), (c, d)\}$에 대해 G_2에서 $s=a, t=d, u=b, v=c$, $D=\{(u, v)\}$로 설정한다. 또한 $K=|A_1|$으로 설정한다. 이 변환은 2-DSHP 예제의 크기(size)에 대해 다항 시간 안에 이루어 질 수 있음은 쉽게 알 수 있다.

이제 2-DSHP의 예제가 참값을 가지면 NDP의 decision problem의 예제에서도 참값을 가지고, 2-DSHP의 예제가 거짓값을 가지면 NDP의 decision problem의 예제도 거짓값을 가짐을 보이고자 한다. 먼저 2-DSHP의 예제가 참값을 가지면 NDP의 예제에서도 A'을 단절했을 때 모든 $s-t$ 경로가 호 (u, v)를 통과하도록 만드는 A'이 존재한다. 왜냐하면 G_2에서 마디를 공유하지 않는 $s-u$ 경로 P_{su}와 $v-t$ 경로 P_{vt}가 존재할 때 $V(P_{su}) \subset X$, $V(P_{vt}) \subset \overline{X}$(여기서, $V(P_{su})$는 s에서 u로의 경로를 구성하는 마디의 집합, $\overline{X}=V_2 \backslash X$)인 임의의 X와 $\overline{X}$에 대해서 생성되는 단절 $(X : \overline{X})$을 볼 때, $A'=(X : \overline{X}) \backslash \{(u, v)\}$로 설정한다면 $\tilde{G}_2=(V_2, A_2 \backslash A')$의 모든 $s-t$ 경로들은 호 (u, v)를 통과하기 때문이다. 이때 $|A'| \leq K$이 만족되므로 NDP의 예제도 참값을 가진다. 반면 2-DSHP의 예제가 거짓값을 가지면 NDP의 예제도 거짓값을 갖는다. 2-DSHP의 예제가 거짓인 경우는 두 가지가 있다. 첫째 마디 a에서 마디 b까지의 경로나 마디 c에서 마디 d까지의 경로 자체가 없는 경우이다. 이 경우에는 NDP의 예제가 거짓인 것은 자명하다. 둘째, 마디 a에서 마디 b까지의 경로와 마디 c에서 마디 d까지의 경로가 서로 마디를 공유하지 않고서는 두 개의 경로를 구성할 수 없는 경우이다. 그러한 경우 G_2에서는 호 (u, v)를 통과하는 모든 $s-t$ 경로상에는 호 (u, v)를 포함하는 순환(cycle)이 포함된다. 이때 $\{s, u\} \subset X, \{v, t\} \subset \overline{X}$인 임의의 X와 $\overline{X}$에

대해서 생성되는 단절 $(X:\overline{X})$을 고려할 때, $A'=(X:\overline{X})\backslash\{(u,v)\}$로 설정한다면 A'을 절단했을 때 어떠한 $s-t$ 경로도 존재하지 않는다. 그 이유는 다음과 같다. 먼저 A'을 절단하면 호 (u,v)를 통과하지 않는 모든 $s-t$ 경로는 차단된다. 반면 호 (u,v)를 통과하는 모든 $s-t$ 경로는 호 (u,v)를 절단하지 않기 때문에 $s-t$ 경로가 그대로 유지될 수 있을 가능성이 있다. 그러나 그러한 $s-t$ 경로에서 호 (u,v)를 포함하는 순환을 제외시킨 경로는 A'을 절단하면서 차단되기 때문에 결국 호 (u,v)를 통과하는 모든 $s-t$ 경로 또한 A'을 절단하면서 차단된다. 그러므로 $\tilde{G}_2=(V_2,A_2\backslash A')$의 모든 $s-t$ 경로들은 존재할 수 없다. 따라서 NDP의 decision problem에서 호 (u,v)를 고려한 가능해는 존재할 수 없으므로 거짓값을 가지게 된다. 그러므로 $|D|=1$을 가지는 NDP의 decision problem은 NP-Complete이다.

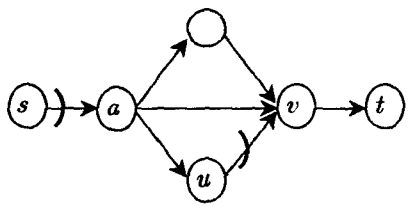
여기서 $|D|=1$을 가지는 NDP의 decision problem은 $|D|\geq 1$을 가지는 NDP의 decision problem의 특수한 경우이므로 NDP의 decision problem은 NP-Complete이다. 따라서 NDP는 NP-hard이다.

3. 타부 탐색 알고리즘

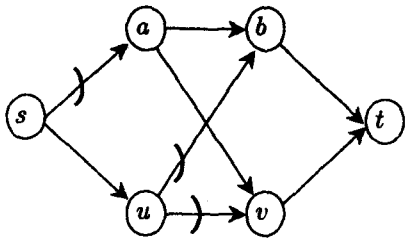
네트워크 전환문제를 풀기 위해서 제안한 알고리즘은 먼저 2단계의 사전 처리를 수행한 후 사전 처리에 의해서 제거된 전환호를 전환호의 집합에서 제외시킨다. 갱신된 전환호의 집합을 가지고 좋은 가능해를 찾기 위한 알고리즘을 적용한다. 알고리즘은 하나의 전환호에 대해서 타부 탐색 알고리즘을 적용하여 좋은 가능해를 찾고, 여러 개의 전환호를 가지는 문제를 풀기 위해 타부 탐색 알고리즘을 하부루틴으로 사용한다. 각각의 전환호에 대해서 찾은 가능해 중 가장 좋은 해를 최종적으로 알고리즘에서 구한 가장 좋은 해로 선택한다. 따라서 3장에서는 하나의 전환호를 가지는 문제를 풀기 위한 타부 탐색 알고리즘에 초점을 맞춘다.

3.1 해의 형상

네트워크 전환문제는 제약식이 추가된 최저 비용의 $s-t$ 단절(minimum cut)문제로 볼 수 있다. 우리는 전환호를 포함시키는 $s-t$ 단절을 탐색해 나갈 것이다. 그러나 단순히 전환호가 포함된 $s-t$ 단절이 아니라 $s-t$ 단절에서 전환호가 제거되었을 때는 $s-t$ 경로가 존재하는 단절을 추구한다. 이때 우리는 전환호를 고려한 가능해를 찾게 된다. 단순히 전환호가 포함된 $s-t$ 단절을 구한 경우 단절에 포함된 전환호가 제외되더라도 $s-t$ 경로가 존재하지 않아 문제에서 요구하는 가능해가 될 수 없는 경우가 발생할 수 있다.



<그림 1> $Cut=\{(s,a),(u,v)\}$



<그림 2> $Cut = \{(s, a), (u, b), (u, v)\}$

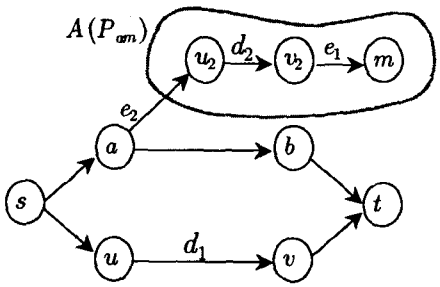
예를 들어 위 <그림 1>과 <그림 2>는 단순히 전환호 (u, v)를 포함시키는 최저 비용의 $s-t$ 단절(minimum cut)을 구한 경우이다. <그림 1>에서는 $s-t$ 단절 $(X:\overline{X}) = \{(s, a), (u, v)\}$에 전환호 (u, v)가 포함되어 있다. 그러나 $s-t$ 단절에서 전환호 (u, v)를 제외시켰을 때 s에서 t로의 경로가 존재하지 않으므로 비가능해가 된다. 그러나 <그림 2>를 보면 $s-t$ 단절 $(X:\overline{X}) = \{(s, a), (u, b), (u, v)\}$에 전환호 (u, v)가 포함되어 있는 가능해를 보여준다. 왜냐하면 $s-t$ 단절 $(X:\overline{X})$을 전환호 (u, v)를 제외하고 절단하였을 때 출발점 s에서 도착점 t까지의 경로가 존재하고, 이때 모든 $s-t$ 경로가 전환호 (u, v)를 반드시 통과하기 때문이다.

3.2 사전 처리

본 절에서는 네트워크 전환문제를 보다 효율적으로 풀기 위한 사전 처리에 대해 논한다. 여기서 제시하는 사전 처리는 단순히 문제의 크기를 줄이는 것을 통한 계산 노력 절감을 추구하는 사전 단계의 기능뿐 아니라, 네트워크 전환 문제의 초기해를 설정하는데 이용할 수 있는 특징을 지니고 있다. 또한 본 연구에서 제시될 알고리즘이 각각의 전환호에 대해서 타부 탐색 알고리즘을 적용하여 각 전환호에 대한 가능해를 구하기 때문에 가능해가 존재할 수 없는 전환호를 미리 제거하므로써 제거되는 전환호에 따른 타부 탐색 알고리즘에 대한 반복 수행을 생략하는 계산 노력의 절감을 꾀하고자 하였다.

(사전 처리 1)

네트워크 전환문제의 구조상 s에서 t로의 경로를 구성하는 데 있어 아무런 관련이 없는 마디나 호를 제거하더라도 문제의 가능해를 구하는 것에는 전혀 영향을 끼치지 않는다. 따라서 시작점과 도착점을 제외한 모든 마디에 대해서 인접한 호의 개수가 1인 마디를 제거시킨다. 또한 이 마디에 인접한 모든 호들을 제거시킨다. 제거후 갱신된 그래프에 대해서 해당되는 마디와 호를 제거시키는 과정을 반복한다. 이렇게 해서 제거된 호 중에서 전환호 d_i가 있다면 전환호의 집합을 $D \leftarrow D \backslash \{d_i\}$로 갱신한다.



<그림 3> $A(P_{am})$ 제거, $D = \{d_1, d_2\}$

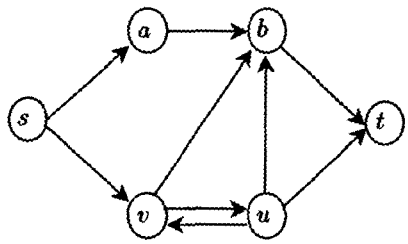
예를 들어, 위 <그림 3>의 경우에는 마디 m과 대응되는 호 e_1을 제거하고, 다시 마디 v_2와 대응되는 호 d_2를 제거한다. 계속해서 마디 u_2와 호 e_2를 제거시키면 결과적으로 $A(P_{am})$이 제거된다.

이때 $D \leftarrow D \backslash \{d_2\}$로 갱신한다.

(사전 처리 2)

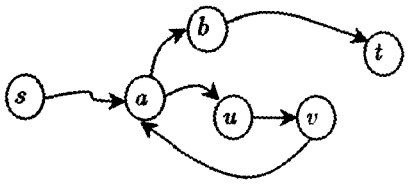
그래프 $G=(V,A)$와 전환호 $(u,v) \in D$가 주어졌을 때, 문제에서 요구하는 가능해의 경우에 임의의 $s \in X, t \in \overline{X}$인 $X, \overline{X}$에 대해서 $(X, E(X))$에서는 s에서 u로의 경로 P_{su}와 $(\overline{X}, E(\overline{X}))$에서는 v에서 t로의 경로 P_{vt}가 존재해야 한다는 점에 착안하여, $G=(V,A)$에서 $v \notin V(P_{su}), u \notin V(P_{vt})$를 만족하는 s에서 u로의 경로 P_{su}와 v에서 t로의 경로 P_{vt}가 존재하는지를 확인한다. 만약 이러한 조건을 만족하는 두 경로 P_{su}, P_{vt}가 존재하지 않는다면 전환호 (u,v)를 고려한 가능해는 존재하지 않는다. 왜냐하면 $s \in X, t \in \overline{X}$인 어떤 $X, \overline{X}$를 설정하더라도 가능해를 주는 $s-t$ 단절 $(X:\overline{X})$은 존재하지 않기 때문이다. 따라서 전환호 (u,v)에 대해 위에서 제시한 조건을 만족시키는 두 경로 P_{su}, P_{vt}가 존재하지 않는다면 전환호 (u,v)를 전환호의 집합 D에서 제외시킨다.

앞의 <그림 2>의 경우에는 s에서 u로의 경로에서 v를 지나지 않고, v에서 t로의 경로에서는 u를 지나지 않는 경우로 전환호 (u,v)를 고려한 가능해가 존재함을 알 수 있다. 즉 $V(P_{su}) \subset X$, $V(P_{vt}) \subset \overline{X}$인 임의의 $X, \overline{X}$에 대한 $s-t$ 단절 $(X:\overline{X})$는 모두 가능해를 제공한다.



<그림 4> s에서 u로의 경로는 반드시 v를 경유

그러나 <그림 4>의 경우에는 s에서 u로의 경로가 v를 방문할 수 밖에 없는 경우이다. 이 경우 전환호 (u,v)를 고려한 가능해가 존재하지 않는다. 왜냐하면 이런 경우 $\{s,u\} \subset X, \{v,t\} \subset \overline{X}$인 어떤 $X, \overline{X}$를 설정하더라도 $s-t$ 단절 $(X:\overline{X})$에서 전환호 (u,v)가 제외되었을 때 s에서 t로의 경로가 존재할 수 없기 때문이다. 따라서 이러한 경우 전환호 (u,v)는 전환호의 집합 D에서 제외시킨다. 즉 $D \leftarrow D \backslash \{(u,v)\}$. 사전 처리 1과 2의 결과 $D=\varnothing$이면 비가능 문제(infeasible problem)가 되어 종료한다. 그러나 사전 처리 1과 2로 비가능 문제를 모두 판별할 수는 없다.



<그림 5> P_{vt}는 $V(P_{su})$의 한 마디인 a를 경유

예를 들어 <그림 5>를 보면 사전 처리 1과 2에 의해 제거되는 전환호가 없지만 v에서 t로의 경로가 $V(P_{su})$에 속하는 마디인 a를 경유해야만 하는 경우로 전환호 (u,v)를 고려한 가능해는 존재

하지 않게 된다. 왜냐하면 이런 경우 $\{s,u\} \subset X$, $\{v,t\} \subset \overline{X}$인 임의의 $X, \overline{X}$에 의해 생성되는 어떤 $s-t$ 단절 $(X:\overline{X})$도 가능해를 제공하지 못하기 때문이다. 그러나 이러한 모든 경우를 고려하는 것은 전환호 (u,v)에 관한 가능해 존재 여부에 대한 문제가 NP-Complete 임을 감안하다면 매우 비효율적임을 예상할 수 있다. 따라서 사전 처리의 범위를 위에 제시한 범위로 한정한다. 또한 본 연구에서 언급되는 모든 경로는 단위 길이를 가지는 호에 대한 최단 경로로 구현하고, $V(P_{su})$를 저장하므로써 나중에 전환호 (u,v)에 대한 초기해로 이용한다

3.3 타부 탐색 알고리즘의 특성

본 절에서는 각각의 전환호에 대해서 좋은 가능해를 찾기 위한 타부 탐색 알고리즘을 제시한다.

타부 탐색의 가장 큰 특징은 메모리 기반 탐색 전략(memory-based search strategy)이라는 점이다. 타부 탐색은 매 반복 수행(iteration)마다 이웃해 집합 $N(S)$을 생성하고 이 중 하나의 해로 이동(move)하면서 보다 좋은 해를 얻어 나가는 알고리즘이다. 그러나 이러한 과정에서 국소해(local optimal solution)에 머물게 되는 상황이 발생하게 된다. 이를 방지하기 위해 메모리 이용을 통해서 알고리즘의 진행 과정 중에 최근에 검색했던 해에 대한 정보를 저장하므로써 최근에 검색했던 해로 돌아가지 못하도록 하여 이전의 해로 돌아감으로 인해 생기는 순환(cycle)을 막는다. 이러한 역할을 하는 정보를 타부(TABU)라 한다. 메모리는 타부를 저장하는 타부 목록(TABL, tabu list)을 도입하므로써 필요하게 된다. 타부 목록에는 크기(TLS, tabu list size)가 있어서 일정 반복 회수동안 타부가 되고, 일정 회수가 지나면 타부 목록에서 제외된다. 이러한 타부 목록의 크기가 너무 작으면 순환을 방지한다는 역할을 수행할 수 없게 되며, 크기가 너무 클때는 이웃해로의 이동에 대한 너무 많은 제약을 주게 된다.

반면 타부 목록의 사용은 탐색 과정에서 방문하지 않은 좋은 해로의 이동을 금지하게 되는 단점을 초래하기도 한다. 다른 해로의 이동이 타부라 하더라도 지금까지 발견한 가장 좋은 해보다 더 좋은 해일 경우, 타부 상태를 무시하고 새로운 해를 받아들여 이동을 허용한다. 이것이 열망 기준(aspiration criteria)이다.

또한 원문제(original problem)의 가능해만을 방문하는 것이 아니라 비가능해로의 이동도 허용하므로써 가능해와 비가능해 근접에 위치해서 방문되지 못하고 놓치는 좋은 해를 얻을 수 있는 가능성을 허용하는 전략적 진동(strategic oscillation)등의 방법을 포함하고 있다.

본 연구에서는 사전 처리 결과 갱신된 전환호의 집합을 구성하는 각각의 전환호 d_i에 대해서 타부 탐색 알고리즘을 실행하므로 기본적으로 사전 처리 후 $|D| \neq 0$ 이라고 가정한다. 먼저 알고리즘에서의 가능해를 정의한 후, 본 연구에서 제시하는 타부 탐색 알고리즘의 주요 특성을 살펴보기로 한다.

가능해 : 그래프 $G=(V,A)$와 $D=\{(u,v)\}$가 주어졌을 때, $s \in X, t \in \overline{X}$이면서 그래프를 양분한 $(X, E(X))$와 $(\overline{X}, E(\overline{X}))$에 대해서,

$(X, E(X))$에서는 s에서 u로의 경로가 존재하며, $(\overline{X}, E(\overline{X}))$에서는 v에서 t로의 경로가 존재하는 $X \subset V$를 "가능한 마디의 집합" 이라 정의할 때, 가능한 마디의 집합 X에 의해 형성되는 단절 $(X : \overline{X})$를 전환호 (u, v)에 대한 가능해라 정의한다.

초기해(initial solution) : 전환호 $d = (u, v)$에 대한 초기해는 사전 처리 작업에서 구한 s에서 u로의 경로상에 있는 마디들을 초기해로 설정한다. 즉, $X \leftarrow V(P_{su})$

이웃해(neighborhood) : 현재의 해를 나타내는 마디의 집합 X를 기준으로 X에 포함되어 있지 않는 마디를 추가하거나(add), X에 포함되어 있는 마디를 $\overline{X}$로 옮김으로써(drop) 얻어지는 모든 해를 이웃해라 정의한다.

이동(move) : 우선적으로 타부 목록에 없는 가능해 중 가장 좋은 것을 선택한다. 이러한 조건을 만족하는 해가 없다면 타부 목록에 있는 가능해중 열망 기준을 만족시키면 이동한다. 그러나 이동 가능한 가능해가 존재하지 않는다면, 타부 목록에 없는 가장 좋은 비가능해로 이동한다. 이러한 비가능해조차 없는 경우 타부 목록에 있는 가장 좋은 비가능해를 선택해서 열망 조건이 만족되면 이동시킨다.

타부 속성(tabu attribute) : 타부에 유지하는 방식을 나타내는 것으로써, 현재해 $(X : \overline{X})$를 결정짓는 마디의 부분 집합 X에 추가되거나, X에서 삭제된 마디를 타부로 정의하고, 이를 타부 목록에 저장한다. 단, 저장시 이동에 있어 추가 또는 삭제에 대한 구분은 없는 것으로 한다.

열망 기준(aspiration criteria) : 현재까지 구한 가장 좋은 목적식의 값보다 작은 값을 가지는 해의 경우에는 타부 상태라 하더라도 타부해로의 이동을 허용한다.

장기메모리(long term memory) : 기존에 찾은 가장 좋은 해보다 더 좋은 가능해를 찾은 경우에 그 가능해 $(X : \overline{X})$를 결정짓는 마디의 부분 집합 X를 장기 메모리에 저장한다.

장기메모리를 이용한 새로운 시작해 설정 : 장기 메모리에 저장된 마디의 부분 집합 X에 속하는 각각의 마디들로 들어오는 호에 대해 호 길이를 증가시킨 후(즉, $w_e = \lambda \times w_e$)v를 거치지 않는 s에서 u로의 최단 경로를 구한다. 이때 경로에 속하는 마디를 X로 설정하여 생성되는 단절 $(X : \overline{X})$를 새로운 시작해로 설정한다. 단, 전환호 $d = (u, v)$, $w_e = 1$ for $e \in A$, λ는 미리 정해진 상수

임의 추출 방식을 통한 새로운 시작해 설정 : 우선 $s, u \in X$와 $v, t \in \overline{X}$로 설정한 후 나머지 마디에 대해서는 0과 1 사이의 임의의 수를 생성해서 CTR보다 작으면 X에, 그렇지 않으면 $\overline{X}$에 속하도록 한다. 단, 전환호 $d = (u, v)$, CTR는 미리 정해진 상수

종료 조건(stop criterion) : 각 전환호 $d_i \in D$에 대해 반복 회수가 $MAXITR$을 넘으면 다음 전환호 $d_{i+1} \in D$에 대해 반복 수행을 한다.

모든 전환호에 대해 반복 수행을 한 후 전체 알고리즘을 종료한다. 단, $MAXITR$는 미리 주어지는 상수

초기해를 구하는 알고리즘을 따로 설정하지 않고 사전 처리 작업에서 구해진 정보를 이용해서 초기해를 설정한다. 이렇게 구한 초기해는 X에 마디를 추가하거나, X에서 마디를 제거함으로써 여러 이웃해를 생성할 수 있다. 여기서 X에 마디를 추가하는 것은 $\overline{X}$에 속하는 마디를 X로 옮기는 것을 의미하고, X에서 마디를 제거하는 것은 X에 속하는 마디 하나를 $\overline{X}$로 옮기는 것을 의미한다. 다만 전환호가 (u,v)일 때 마디 s,u,v,t는 이동을 금지시켜 항상 $s,u \in X, v,t \in \overline{X}$가 성립되도록 한다. 이는 구한 해가 가능해 또는 비가능해 인지를 구분하기 전에 $s-t$ 단절에 전환호를 포함하도록 보장하기 위함이다. 타부의 설정으로 인해 놓칠 수 있는 좋은 해들을 얻기 위해 열망 기준을 설정하여 이를 만족시키는 해는 설사 타부라도 이동을 허용하게 만든다. 또한 이동 가능한 가능해가 없는 경우 비록 비가능해라 하더라도 이동을 허용한다. 타부와 같은 단기 메모리와는 별도로 장기 메모리를 이용해서 알고리즘 진행 과정 중에 좋은 해들을 저장하여 다른 해 영역을 탐색하기 위한 해를 생성할 때 무작위로 해를 생성하는 것이 아니라 기존의 좋은 해의 특성을 고려한 해를 생성토록 한다. 다만 새로운 시작해를 설정할 때 장기메모리에 저장된 가용한 정보가 없는 경우에는 임의 추출 방식을 이용한다.

반면 알고리즘의 진행 과정에서 생성되는 하나의 이웃해마다 이웃해의 가능해/비가능해 상태를 구분하기 위해서는 $(X,E(X))$, $(\overline{X},E(\overline{X}))$에서 $V(P_{su}) \cap V(P_{vt}) = \varnothing$ 인 P_{su}와 P_{vt}의 존재를 확인해야 한다. 이때 전환호 $d=(u,v)$이며 $s \in X, t \in \overline{X}$.

그러나 상황에 따라 이러한 확인 과정을 생략할 수 있다. 예를 들어 $(X,E(X))$에서 P_{su}가 존재하고 $(\overline{X},E(\overline{X}))$에서 P_{vt}가 존재하면서 $\overline{X}$에 속하는 마디 m을 X로 옮기는 이동(add move)인 경우, 이웃해의 가능해 여부 검사는 $(\overline{X},E(\overline{X}))$상에 P_{vt}가 존재하는지만 검사하면 된다. 단, $m \notin V(P_{vt})$인 경우 $(\overline{X},E(\overline{X}))$상에서도 P_{vt}의 존재 여부에 대해 검사를 생략할 수 있다. 따라서 실제 구현시 현재의 해에서 이웃해를 생성할 때의 이동에 대한 구분과 $(X,E(X))$,$(\overline{X},E(\overline{X}))$상의 어느 부분에 경로가 존재하는가에 대한 정보와 더불어 존재하는 경로에 대해 $V(P_{su})$, $V(P_{vt})$의 정보도 함께 저장하여 경우에 따라 생성된 이웃해의 가능해 여부 확인 과정을 생략하여 계산적 노력을 줄일 수 있다.

4. 실험 결과 및 분석

4.1 실험 환경

본 연구에서 제시한 타부 탐색 알고리즘의 성능을 평가하기 위해 이를 프로그램으로 구현하였다.

제시한 알고리즘의 성능을 비교하기 위한 최적해는 Curet이 제시한 정수 계획 모형을 토대로 IP

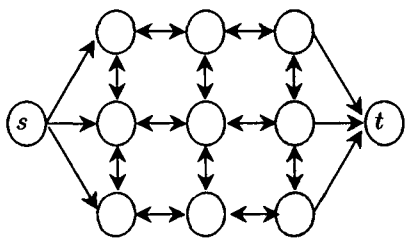
Solver를 통하여 구했다. 모든 프로그램은 C로 구현되었으며, 512MB의 RAM을 지닌 Pentium IV(2.53GHz) PC에서 실행되었다.

4.2 실험 예제

기존 연구에서 Curet[9]은 유방향 격자형 그래프에 대해서만 실험을 하였고, Erken[10]은 무방향 그래프에서만 실험을 하였다. 그러나 고려중인 네트워크 전환문제가 무방향 그래프에 대해서도 NP-hard임은 아직 증명되지 않았다.

따라서 본 연구에서는 실험을 위한 예제 문제로써 유방향의 일반형, 격자형 그래프를 생성하여 별도의 실험 결과를 제시하였다. 또한 그래프의 크기는 IP Solver를 통해 합리적인 시간안에 최적해를 구할 수 있는 크기로 제한하였다.

단, 생성된 예제 그래프(일반형, 격자형)는 출발점에서 도착점에 이르는 경로는 반드시 존재하도록 하고, 출발점으로 들어오는 호와 도착점에서 나가는 호는 존재하지 않도록 한다. 전환호는 생성된 그래프의 호 중에서 임의대로 선택한다. 출발점을 꼬리 마디로 가지는 호와 도착점을 머리 마디로 가지는 호는 전환호에서 제외시키고, 호의 절단 비용은 1과 10사이의 정수에서 무작위로 할당한다.



<그림 5> 격자형 그래프

4.3 실험 결과

위에서 생성된 예제에 대해 목적식에 최단 경로 구성에 가중치가 고려되지 않은 것과(즉, $\epsilon = 0$), 최단 경로 구성에 가중치가 고려된 것(즉, $\epsilon \neq 0$)으로 구분해서 실험하였다.

알고리즘의 성능을 기술하기 위해 Z^*는 IP Solver가 제시한 최적값을 의미하고, Z는 타부 탐색 알고리즘의 결과로 구한 최종해의 목적함수 값을 의미할 때 GAP을 다음과 같이 정의하였다.

$$\text{GAP} = \frac{Z - Z^*}{Z^*} \times 100\ (\%)$$

이때 해를 구하는데 걸린 시간은 알고리즘 수행을 완료하는데 소요된 CPU시간으로, 단위는 초(sec)이다.

<표 1>에서 <표 4>까지는 목적식의 내용과 그래프의 유형에 따라 구분하여 기술된 실험 결과를 나타낸다. 각 표에서의 행(row)은 설정된 크기를 지니는 그래프에 대해 10개의 예제를 생성 후 실험한 결과를 평균치를 구해 기술하였다.

먼저 목적식에 최단 경로 구성의 가중치가 고려되지 않은 문제의 결과는 <표 1>과 <표 2>에서 볼 수 있다. 일반형 그래프에서의 평균 GAP은 3.12%이고, 격자형 그래프에서의 평균 GAP은 1.64%로써 상대적으로 격자형 그래프에서 보다 좋은 실험 결과를 확인할 수 있다. 또한 전체적으로는 평균 GAP이 2.70%로써 최적해와 근사한 해를 얻을 수 있음을 보여준다.

보다 자세히 결과를 살펴보면 제안된 알고리즘

은 문제의 크기가 작다고 해서 보다 좋은 결과를 나타내지 못하고 있다. 오히려 문제의 크기가 커지더라도 마디의 개수에 대한 호의 개수가 비교적 많은 밀집한 그래프(dense graph)상에서 좋은 결과를 보여주고 있다. 마디대 호의 비가 2 이하인 것을 희소 그래프(sparse graph)라 하고 2 이상인 것을 밀집 그래프라고 구분할 때, 실험 결과를 보면 확연히 차이가 나는 것을 알 수 있다. 예를 들어 <표 1>에서 $|V|$=100, $|A|$=600, $|D|$=3인 밀집 그래프에 대한 실험 결과 GAP은 3.79%이나 <표 1>의 $|V|$=100, $|A|$=200, $|D|$=3인 크기가 보다 작으면서 희소 그래프에 대한 GAP 결과는 8.75%임을 볼 수 있다. 실제로 알고리즘을 구현하는 과정에서 희소 그래프에서 타부 목록 크기의 변화에 따라 예제별로 실험 결과가 민감하게 반응함을 관찰할 수 있었다. 반면 밀집 그래프와 격자형 그래프에서는 타부 목록 크기의 변화에 상관없이 실험 결과가 안정된 모습을 보여주었다.

또한 Re.D1과 Re.D2의 결과를 보면 사전 처리 1과 2에 의해 전환호가 제거된 예제의 수가 적은 것을 알 수 있는데, 사전 처리 2의 과정은 알고리즘 수행 시간에 미미한 영향을 끼치면서도 초기해를 설정할 수 있는 장점이 있는 반면, 사전 처리 1은 상대적으로 수행 시간을 많이 소요함에도 불구하고 문제의 크기를 줄이거나 그에 따른 전환호를 제거하는 것에 따른 전체적 알고리즘의 계산 노력 절감에 미미한 영향을 끼치는 것으로 판단된다.

따라서 사전 처리 1의 과정을 생략한다면 평균 알고리즘 수행 시간을 줄일 수 있을 것으로 예상된다.

앞의 최단 경로 구성의 가중치를 고려하지 않은 문제에서 다루었던 예제를 대상으로 목적식에 최단 경로 구성의 가중치를 고려하여 실험을 한 결과는 <표 3>과 <표 4>에 나타냈다.

일반형 그래프에 대한 평균 GAP은 1.87%이며, 격자형 그래프에 대한 평균 GAP은 0.22%이다. 전체적으로는 평균 GAP 1.40%로써 앞의 실험과 마찬가지로 최적해와 근사한 해를 구할 수 있음을 알 수 있다.

여기서도 희소 그래프에서 타부 목록 크기의 변화에 따른 실험 결과가 민감하게 반응하였다. 한편 <표 4> $|V|$=38, $|A|$=136, $|D|$=1인 그래프에 대한 평균 계산 시간은 0.493초인 반면 <표 2> $|V|$=38, $|A|$=136, $|D|$=1인 그래프에 대한 평균 계산 시간은 0.209초인 것에서 알 수 있듯 제안된 알고리즘의 수행 시간은 같은 크기의 문제에 대해서 목적식에 최단 경로 구성의 가중치가 고려된 것의 수행 시간이 2배 이상 소요되는 것을 확인할 수 있는데, 이것은 목적식에 최단 경로 구성의 가중치가 고려되지 않은 문제에 대해서 이웃해의 가능해 여부 확인 과정을 생략할 수 있었던 경우가 많은 반면 최단 경로 구성의 가중치가 고려되는 문제에서는 최단 경로를 유지해야 하는 이유로 이 과정을 생략할 수 없기 때문으로 분석된다.

사전 처리의 결과를 보면 격자형 그래프에서는 전환호가 제거된 예제가 없는 것을 볼 수 있는데 이는 격자형 그래프의 형상에서 알 수 있듯 인접한 호의 개수가 1인 마디가 없고, 마디를 공유하지 않는 P_{su}, P_{vt}가 반드시 존재하기 때문으로 격자형 그래프에 대해서는 사전 처리 1을 생략하는 것이 바람직하다.

<표 1> $\epsilon = 0$, 일반형 그래프에 대한 실험 결과

$\|V\|$	$\|A\|$	$\|D\|$	Re.D1	Re.D2	Opt.time	Tabu.time	GAP(%)	Not.Opt
30	60	1	0	0	0.061	0.120	3.33	0
		2	0	1	0.025	0.245	0.42	0
		3	1	2	0.039	0.314	11.93	0
	120	1	0	0	0.414	0.236	0.00	0
		2	0	1	0.508	0.414	0.00	0
		3	0	0	0.139	0.536	0.00	0
	180	1	0	0	3.536	0.262	0.00	0
		2	0	1	0.481	0.437	0.00	0
		3	0	0	0.459	0.756	0.00	0
50	100	1	0	0	0.091	0.286	0.00	0
		2	0	5	0.117	0.389	6.48	0
		3	0	7	0.081	0.548	2.28	0
	200	1	0	0	4.561	0.559	0.44	0
		2	0	1	1.119	0.992	0.91	0
		3	0	0	0.647	1.531	0.00	0
	300	1	0	0	268.816	0.691	0.57	0
		2	0	0	1.755	1.472	0.60	0
		3	0	0	1.464	2.206	0.00	0
100	200	1	0	0	28.246	0.975	7.06	0
		2	0	8	1.356	1.401	5.66	0
		3	2	5	1.319	2.192	8.75	0
	400	1	0	0	27.494	2.059	11.50	0
		2	0	0	12.661	4.786	6.10	0
		3	0	0	8.185	6.958	0.96	0
	600	1	0	0	30.470	2.747	3.89	1
		2	0	0	12.000	6.248	9.19	0
		3	0	0	12.905	8.588	3.79	0

* 각 그래프에 대해 10개의 예제를 대상으로 실험한 결과를 평균치로 기술한 것임

$|V|$: 마디의 개수

$|A|$: 유방향호의 개수

$|D|$: 전환호의 개수

Re.D1 : 10개의 예제 중 사전처리 1에 의해 전환호가 1개 이상 제거된 예제의 수

Re.D2 : 10개의 예제 중 사전처리 2에 의해 전환호가 1개 이상 제거된 예제의 수

Opt.time : IP Solver를 통해 10개의 예제에 대한 최적값을 구하는 데 걸린 평균 시간, 단위는 초(sec)

Tabu.time : 제안한 알고리즘을 통해 10개의 예제에 대한 가능해를 구하는 데 걸린 평균 시간, 단위는 초(sec)

GAP : 본문내 설명 참조

Not.Opt : 10개의 예제 중 IP Solver를 통해 합리적인 시간안에 최적해를 구하지 못한 예제의 수

<표 2> $\epsilon = 0$, 격자형 그래프에 대한 실험 결과

$\|V\|$	$\|A\|$	$\|D\|$	Re.D2	Opt.time	Tabu.time	GAP(%)	Not.Opt
38	136	1	0	1844.803	0.209	0.00	0
		2	0	564.445	0.440	0.00	0
		3	0	0.064	0.648	0.00	0
51	182	1	0	183.384	0.663	2.89	1
		2	0	0.942	1.079	1.60	1
		3	0	0.158	2.353	3.72	2
82	300	1	0	0.728	2.252	0.49	3
		2	0	0.633	3.627	2.81	0
		3	0	2194.381	5.426	1.69	0
102	380	1	0	97.595	6.544	0.83	5
		2	0	1.357	5.273	3.11	0
		3	0	11.134	8.297	1.77	0

<표 3> $\epsilon \neq 0$, 일반형 그래프에 대한 실험 결과

$\|V\|$	$\|A\|$	$\|D\|$	Re.D1	Re.D2	Opt.time	Tabu.time	GAP(%)	Not.Opt
30	60	1	0	0	0.053	0.189	0.00	0
		2	0	1	0.031	0.382	1.50	0
		3	1	2	0.041	0.451	0.00	0
	120	1	0	0	0.437	0.356	0.00	0
		2	0	1	0.334	0.668	0.00	0
		3	0	0	0.117	0.949	0.00	0
	180	1	0	0	2.667	0.453	0.00	0
		2	0	1	0.451	0.743	0.00	0
		3	0	0	0.461	1.253	0.00	0
50	100	1	0	0	0.114	0.611	1.71	0
		2	0	5	0.131	0.817	1.84	0
		3	0	7	0.084	0.996	1.67	0
	200	1	0	0	4.319	1.151	1.29	0
		2	0	1	1.408	2.029	0.39	0
		3	0	0	0.619	3.034	0.89	0
	300	1	0	0	89.942	1.484	0.54	0
		2	0	0	2.213	2.947	0.00	0
		3	0	0	1.494	4.334	0.00	0
100	200	1	0	0	7.497	2.512	1.52	0
		2	0	8	0.852	3.183	5.54	0
		3	2	5	0.455	4.859	11.14	0
	400	1	0	0	21.346	5.844	7.21	0
		2	0	0	10.076	12.631	4.99	0
		3	0	0	6.977	19.818	6.12	0
	600	1	0	0	106.036	7.867	1.80	0
		2	0	0	10.984	17.861	2.27	0
		3	0	0	11.305	25.017	0.00	0

<표 4> $\epsilon \neq 0$, 격자형 그래프에 대한 실험 결과

$\|V\|$	$\|A\|$	$\|D\|$	Re.D2	Opt.time	Tabu.time	GAP(%)	Not.Opt
38	136	1	0	900.841	0.439	0.00	0
		2	0	169.194	0.908	0.00	1
		3	0	0.045	1.475	0.00	0
51	182	1	0	1.552	1.372	0.00	1
		2	0	0.625	2.766	0.00	1
		3	0	0.170	4.192	0.35	2
82	300	1	0	0.305	3.992	0.31	2
		2	0	0.377	8.334	0.25	0
		3	0	5.122	12.659	0.22	0
102	380	1	0	92.997	7.087	0.69	5
		2	0	0.853	13.225	0.42	0
		3	0	52.891	20.772	0.62	0

5. 결 론

5.1 연구 요약

본 연구에서는 출발점에서 도착점으로의 네트워크의 흐름이 특정호 중 최소한 하나를 경유하도록 다른 호들을 최소의 비용으로 절단하는 문제를 다루었다.

기존에 제시되었던 수리 계획 모형에서 불필요한 제약식이 존재함을 발견하였고, 네트워크 전환 문제가 유방향 그래프에서 NP-hard임을 증명하였다. 본 연구에서는 가능한 좋은 가능해를 구하기 위해 메타 휴리스틱 알고리즘의 하나인 타부 탐색 알고리즘을 적용하였다. 사전 처리는 2 단계를 통해서 실시하였다. 사전 처리 1은 소요되는 계산 수행 시간에 비해 전체 실험에 따른 계산 수행 시간을 절감하는데 크게 효용이 없는 것으로 판단된다. 그러나 사전 처리 2는 소요되는 계산 수행 시간이 적고, 사전 처리후 초기해를 설정하는데 직접 이용할 수 있다는 장점이 있다. 제안된 알고리즘의 성능은 실험 결과에서도 알 수 있듯 유방향의 일반형 그래프와 격자형 그래프 모두에 대해서 평균 GAP이 3% 이내로써 최적해와 근사한 가능해를 구하였다.

5.2 개선 방향

추후 연구로서는 제안된 알고리즘의 개선과 새로운 알고리즘의 도입을 고려할 수 있다. 첫째, 제안된 알고리즘은 회소 그래프에서 타부 목록 크기의 변화에 상관없이 안정적인 실험 결과를 보여줄 수 있도록 개선할 여지가 남아있다. 이를 개선하기 위해서 이웃해를 설정하는 방법을 변화시키고, 장기메모리를 이용한 새로운 시작해 설정 방법을 개선시킬 필요가 있는 것으로 판단된다. 또한 이러한 일정 회수 동안 다른 해 영역으로 이동한 후에도 해의 개선이 이루어지지 않는 경우에 특정 해 영역을 집중적으로 탐색하도록 알고리즘을 개선하는 것도 하나의 방법으로 판단된다. 또한 제안된 알고리즘은 각각의 전환호에 대해서 가능해를 탐색하기 때문에 전환호의 개수가 증가할수록 전체 알고리즘

의 계산 수행 시간도 선형적으로 증가하는 구조적인 단점을 개선시킬 방안이 필요하다. 둘째, 네트워크 전환문제의 최적해를 구하기 위해서 유효 부등식(valid inequality)의 도입을 통한 분지 절단법(Branch and Cut)을 생각해 볼 수 있다. 제안된 문제가 제약식을 갖는 단절(cut) 문제로 볼 수 있기 때문에 단절 문제에 대한 기존 연구에서 제시된 유효 부등식을 이용할 수 있을 것으로 판단된다.

참 고 문 헌

[1] 김여근, 윤복식, 이상복, 메타 휴리스틱, pp452, 영지문화사, 1999.

[2] A. Hertz and D.de Werra, "Using Tabu Search Techniques for Graph Coloring," Computing, Vol.45, pp.345-351, 1987.

[3] C. Friden, A. Hertz and D.de Werra, "STABULUS: A Technique for Finding Stable Sets in Large Graphs with Tabu Search," Computing, Vol.42, pp.35-44, 1989.

[4] D.R. Shier and D.E. Whited, "Algorithms for generating minimal cutsets by inversion," IEEE TRANSACTIONS ON RELIABILITY, Vol.R-34(4), pp.314-319, 1985.

[5] F. Barahona and A.R. Mahjoub, "On the Cut Polytope," Mathematical Programming, Vol.36(2), pp.157-173, 1986.

[6] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence," Computers & Operations Research, Vol.13(5), pp.533-549, 1986.

[7] F. Glover, "Tabu Search-Part I," ORSA Journal on Computing, Vol.1(3), pp.190-206, 1989.

[8] J.A. Diaz and E. Fernandez, "A Tabu search heuristic for the generalized assignment problem," European Journal of Operational Research, Vol.132(1), pp.22-38, 2001.

[9] Norman D. Curet, "The Network Diversion Problem," Military Operations Research, Vol.6, pp.35-44, 2001.

[10] Ozgrur Erken, "A branch and bound algorithm for the network diversion problem," Master's Thesis, Naval postgraduate School, Monterey, California, 2002.

[11] S. Fortune, J. Hopcroft and J. Wyllie, "The directed subgraph homeomorphism problem," Theoretical Computer Science, Vol.10(2), pp.111-121, 1980.