韓-中 國際 學術會議
發表論文

대륙연결 최적운송경로의 탐색 : GIS의 활용

# Searching for Rail Routes from Korea to Europe Using GIS

1995. 5. 25

서울대학교 도시공학과 객원교수
ILLINOIS 대학교 교수

김 창 호

# 1. 서론

우리나라의 물품을 유럽으로 수출할 때 지금까지는 수출품의 대부분을 인도양과 수에즈 운하를 통과하여 운송했다. 한편 일부화물은 1973년부터 시베리아 횡단철도(TSR : Trans Siberia Rail)를 이용하여 운송되고 있다. 그러나 TSR은 지정학적으로 해상운송보다 7500km나 짧은 이점이 있음에도 불구하고 국제복합운송체제가 다른 자본주의 국가의 운송체계와 같이 신속성과 신뢰성을 가지고 있지 못할 뿐만 아니라 운송의 과학화가 이루어지지 않아 운송일수가 기대보다 오래 걸리며, 더욱이 운송일수가 항상 불안정하여 신뢰성을 상실하고 있는 것이 최대의 약점으로 지적되고 있다. 이러한 운송서비스의 불안정으로 화주들은 TSR의 이용을 기피했고 1985년 이후 컨테이너 물동량은 계속 감소했다. 그러나 러시아는 시장경제 도입과 자율경영을 확대하면서 운송관련조직의 신설 및 개편을 서두르고 있으며 외국기업과의 합작을 통해 선진외국의 자본 및 경영기술을 활발하게 도입하고 있다. 그리하여 TSR의 최대 약점인 운송일수의 불안정성과 화물추적의 미비 점을 개선하려 노력하고 있다.

한편 TSR와 상호 경쟁 및 보완노선인 중국횡단철도(TCR : Trans China Rail)는 TSR보다 더 짧은 수송거리와 좋은 기후조건을 지녔다. 즉 동북아 지역에서 유럽지역까지의 운송에 있어서 해상운송보다 상당히 짧은 운송거리 때문에 운송시간과 운임측면에서 다른 수송로보다 운송경쟁력을 가질 것으로 전망된다. (<표 1> 참조)

<표 1> 기존 운송로와 중국횡단철도(TCR)와의 거리, 시간 비교

| | 거리(km) | 시간(일) |
|---|---|---|
| 해상운송로 | 20610~23,540 | 27~46(W/B)<br>21~45(E/B) |
| 시베리아횡단철도(TSR) | 12,230 | 25~35 |
| 중국횡단철도(TCR) | 10,370 | 21~31 |

자료 : 해운산업연구원, 대륙횡단 철도운송과 정기선사의 대응전략, 1991

주 : 1. 여기 제시된 거리는 부산에서 로테르담까지를 기준으로 한 것임.
2. W/B는 우리 나라에서 유럽으로 가는 방향(west bound)을 나타내고 E/B는 유럽에서 우리 나라로 오는 방향(east bound)을 나타냄.
3. 해상운송은 지정학적 최단거리가 19,790km이지만, 운항선사의 취항경로에 따라 최단경로와 최장경로를 나타내었음.

중국은 TCR이 아시아대륙과 유럽을 연결하는 새로운 비단길의 역할 즉 '황금의

허리띠' 역할을 수행하게 하기 위하여 TCR이 가지고 있는 문제점을 개선하도록 보완책을 강구하고 있다. 첫째, TCR이 갖는 최대약점인 TSR과의 철도궤도폭의 차이를 극복하기 위하여 중국과 구소련 국경지역에 차량교환, 대차교환 등의 환적시설을 설치하고있다. 둘째, TCR은 화물추적을 위한 정보제공을 위해 컴퓨터시스템의 도입과 컴퓨터에 의한 화차의 속도조절, 화물의 안전유지 등 운송서비스의 개선을 위해 다양한 방법을 강구하고 있다.

본 논문은 우리 나라의 물류비를 저감하는 여러 대안을 분석하는 과정의 일환으로 GIS의 활용을 검토하기 위한 pilot study를 소개하고 있다. 다시 말해 TCR의 철도망에 관련된 시설, 운영, 법규 등의 모든 자료를 GIS(Geographical Information System : 지리정보체계)의 컴퓨터 소프트웨어에 저장하여 필요한 내용을 원하는 형태로 대단히 짧은 시간에 가시화 시킨다면, 평상시는 물론이고 여러 가지 운송로의 요인변화로 인한 경로변경이 요구될 때, 화물운송체계에서 운송경로를 결정하는 결정자에게 매우 손쉬우며 확실한 의사결정을 내릴 수 있는 도구가 제공되는 것이다.

다음 장에는 최단경로 탐색을 위한 컴퓨터 프로그램의 알고리즘을 상세히 소개하고자 한다.

# 2. 최단경로 탐색 알고리즘

본 연구의 대상은 철도이지만 최단경로를 탐색하는 기본과정은 일반 도로망에서와 같으므로 여기에서는 도로망에서의 최단경로 탐색 알고리즘을 소개하고자 한다. 특별히 실제 도로망에서 빈번하게 나타나는 회전금지 등을 고려하기 위해 vine 구축 알고리즘을 사용하고 있는데 이는 중국의 철도망에서 일어날 여러 제약조건들(사고를 포함하여)이 발생했을 경우 다른 경로를 찾아가야 하는 상황을 생각할 때 필요한 알고리즘이라고 사료된다. 한편 이 vine 구축 알고리즘은 tree 구축 알고리즘에 비해 더욱 현실적이고 실제적인 경로를 제시하고 있다. 그러나 vine 구축 알고리즘은 tree 구축 알고리즘의 기본 개념을 사용하고 있고 일부분이 공통되므로 두 가지 알고리즘의 정의 및 특성을 서술하고자 한다. 또한 vine을 구축하는 데에 Dijkstra 알고리즘을 사용한 이유를 다른 알고리즘과의 비교를 통해 설명하고자 한다.

## 2.1 경로 탐색 알고리즘에서 도로망의 정의 및 특성

일반적으로 경로 탐색 알고리즘에서 도로망은 교차로를 의미하는 노드 i, j, k .... 의 유한 집합과 도로구간을 의미하는 링크의 유한집합의 조합으로 구성된다. 링크는 여기서 i와 j가 각각 링크의 시작노드와 끝노드를 나타내는 링크 (i,j), ... 로 표현된다.

개별 링크 (i,j) 상에서의 비용(통행시간, 금전적 비용 등을 포함한 일반화 통행비용)은 여기서 $c_{ij}$로 표기한다. 노드들은 1부터 $N_{nodes}$까지 연속적으로 번호가 매겨져 있다고 생각하는 것이 편하다. 따라서 $N_{nodes}$는 도로망 상의 노드의 수이다. 또한 통행이 발생하고 종결하는 통행 중심지(centroid)는 1부터 $N_{cent}$까지 번호가 매겨진다. 링크들은 시작노드의 순서대로 정렬되고 같은 시작노드 내에서는 끝노드의 순서 대로 정렬된다. 이 각각의 링크에는 도로구간의 거리, 속도, 용량, 통행시간 등의 수량적 속성이 부과된다.

경로 탐색 과정에서 사용되는 도로망은 다음과 같은 특징적 성격을 가지고 있다.

① 한 링크에 의해 제공되는 통행 저항의 크기를 비용이라고 한다면, 모든 링크비용은 비음(non-negative)이다. 예로 $c_{ij}$가 링크 (i,j)의 비용이면, $c_{ij} \geq 0$.

② 링크 비용은 주로 통행시간으로 표현된다.

③ 양 방향 링크(two way link)는 일반적으로 두 방향에 같은비용을 갖지 않는다. $c_{ij} \neq c_{ji}$

④ 일방도로(one way)는 링크 (i,j)로 표현되고, 링크 (j,i)는 없다.

⑤ 특정 회전이 금지되거나 제약된 경우가 있다. 링크 (i,j)가 주어져 있고, i -> j -> k로의 회전이 금지되거나 또는 $c_{ij} + c_{jk} < c(i\text{-}j\text{-}k)$일 수 있다. 여기서 $c_{ij} + c_{jk}$는 개별 링크의 통행비용 합이고 c(i-j-k)는 노드 i에서 노드 j를 경유해서 노드 k로 회전할 때의 통행비용이다.

⑥ 도로망의 크기는 상당히 다양하다. 소수의 링크와 노드로 구성된 도로망도 있고 15000개의 링크와 5000개의 노드로 이루어진 도로망도 있다.

⑦ 도로망에서 상대적으로 적은 노드들이 단일 링크에 의해 연결된다. 따라서 링크의 수는 노드의 수에 약 3배정도 되는 경우가 많다.

## 2.2 최단경로 구축 방법 - Tree와 Vine의 차이

통행배정에서 기초적 과정은 통행 출발지에서 통행 도착지까지의 최단경로를 결정하는 것과 최단경로에 통행량을 부과하는 것이다. 이 장은 통행배정의 첫 번째 과정과 관계가 있다. 대부분의 통행배정 과정에서 '비용최소화'가 운전자들이 경로

를 선택하는 기초가 된다고 가정한다. 따라서 통행은 출발지에서 도착지까지의 최소비용경로를 따라 배분된다고 가정한다.

단일 출발지에서 도착노드를 포함한 도로망 상의 다른 노드까지의 최소비용경로는 배정 과정에서 어떤 노드나 링크를 단 1번만 통과(one pass)한다. 한 노드나 링크를 반복해서 순환하는 불필요한 반복은 없어야 한다. 만일 최단경로 결정 과정 중, 각각의 노드에서 경로의 통과가 1번만 고려되고 경로가 중심지(centroid node)를 통과하는 것이 허용되지 않는 다면 그 알고리즘을 단일 통과 알고리즘(once through algorithm)이라 한다.

만일 출발노드 h에 대하여 알고리즘에 따라 경로가 완료되었을 때 도로망 상에서 다른 노드들이 단지 1개의 링크에 의해서만 도착된다면 h에서 다른 노드까지의 경로의 집합은 최소비용 tree(minimal cost tree)또는 단순히 tree라고 한다. 즉 통행배정이 한 tree 상에서만 이루어진다. 최소비용경로의 결정 과정은 tree 구축(tree building)이라 불린다.

만일 링크 (i,j)가 있는 데, tree에서 노드 j에 도착하는 링크가 (i,j)일 경우 (i,j)가 j에 대한 전링크(predecessor link)라고 한다. 그리고 노드 i를 j에 대한 전노드(predecessor node)라 한다.

어떤 경로 구축 알고리즘은 출발 노드에서부터 다른 노드들로 통행을 전개할 때 하나 이상의 경로를 사용한다. 그래서 어떤 노드에 대해서는 1개 이상의 전링크가 존재한다. 그러한 알고리즘에 의한 경로의 집합을 vine 이라 한다.; 그 알고리즘을 vine 구축알고리즘이라 한다.

## 2.3 Tree 구축 알고리즘 - Moore와 Dijkstra와 D'Esopo 방법

이 알고리즘들은 통행배분모형의 최소경로결정을 위해 흔히 사용된다. 이 알고리즘들은 Moore(1957), Dijkstra(1959), D'Esopo(1978)에 의해 만들어 졌다.

전통적인 tree 구축 알고리즘에서는 출발노드에서 도착노드로 가는 경로는 다른 중심지(centroid)를 통과하는 것이 허용되지 않는다. 중심지(Centroid)는 도로망에는 포함되나 말단링크(dead end link), 그 링크의 끝 노드가 다른 링크의 시작 노드로 기여하지 못하는 링크는 제외된다.

## Tree 구축 알고리즘의 일반적 형태

참조를 쉽게 하기 위해 알고리즘은 다음과 같은 세 가지 표에 의해 표현된다.

- ▶ **도로망 표 A (시작노드와 끝노드 그리고 링크통행시간 표현)**
- ▶ **tree표 R : 도로망상의 모든 노드의 현재 등록 상황과 전노드 그리고 그 노드까지의 경로비용이 노드번호의 오름차순으로 정리된 표.**
- ▶ **작업표 L : 'loose-ends' table이라고도 하며 tree 구축 과정에서 어떤 노드에 경로가 도착되었을 때 그 노드가 입력되는 표. 이 표에서 NEXT를 선택한다.**

일부 알고리즘에서는 tree 구축 과정 중에 한 노드에 한 개의 경로 이상이 지나갈 수 있기 때문에 L에 입력된 수는 일정하지 않다. 초기 L은 전부 '0'으로 채워져 있거나 비어 있다고 생각한다.

각 노드는 전노드 p와 h로부터 그 노드까지의 비용을 포함하여 R에 등록된다. 그러나 어떤 단계에서는 비용은 최종 경로비용이 아니라 그 상태까지의 임시비용이므로 어떤 노드 i까지의 비용은 $m_i$로 표현된다. 그러나 $m_i$의 최종 값은 $c_i^*$와 같다.

출발노드 h에서 다른 모든 노드로의 최소비용 tree를 구축하는 과정은 다음과 같다.

## [알고리즘]

**1단계 : (초기화)**

(a) R을 초기화한다.; 모든 노드에 대해 $m_i = \infty$(h에서 다른 노드까지 가능한 실제 비용보다 훨씬 큰 수) 그리고 $p_i = 0$으로 초기화한다. $m_h = 0$으로 설정한다.

(b) L을 초기화한다. 그리고 출발노드 h를 등록한다.

(c) NEXT를 출발노드 h로 설정한다.(NEXT는 2단계에서 진행될 tree구축과정에서 고려될 다음 노드를 말한다.)

**2단계 : (NEXT의 고려, R을 갱신하고 L에 등록을 하는 과정을 포함한다.)**

(a) L로부터 NEXT를 제거한다.

(b) 도로망표 표(A)에서 각 링크 (NEXT,k)에 대해(즉, 시작노드를 NEXT로 갖는 링크) R을 참조하여,

만일 $m_k \leq m_{NEXT} + c_{next\ k}$이면 그대로 두고
그렇지 않으면,
(i) R에 $m_k = m_{NEXT} + c_{next\ k}$ 를 설정하고 $p_k$ = NEXT로 둔다.
(ii) k를 L에 등록한다. (여기에서 논의되는 알고리즘에서 경로는 다른 노드로 진행하기 위해 통행죤을 통과할 수 없다. 따라서 k > Ncent이면 k는 L에 등록된다. 어떤 알고리즘에서는 어떤 노드가 동시에 L에 여러 번 등록되는 것을 허용하지 않는다. 따라서 이러한 알고리즘에서는 k가 L에 존재하지 않을 때, 즉 $m_k = \infty$ 일 경우, 등록된다.)

**3단계 : (L에서 고려되어질 다음 노드 NEXT를 선택한다.)**

(a) L이 비었으면 4단계로 간다.
(b) 알고리즘의 규칙에 따라 L로부터 다음 노드 NEXT를 선택한다.
(C) 2단계로 간다.

**4단계 : (최소비용경로와 비용의 도출)**

임의의 도착노드 d에 대해, 도출된 결과는 출발지 h로부터 비용이 감소하는 순서로 출발지 h에서 도착노드 d까지의 최소경로비용, 저장된 $m_d$, $P_d$, 그리고 도착노드 d까지 오는 경로 상의 전노드들의 집합 등으로 이루어져 있다.

(a) 출발지 h에서부터의 최소경로를 구해야 하는 다음 목표노드를 d로 설정한다. 그리고 r = d로 설정한다. 만일 더 이상의 노드가 존재하지 않는다면 중지.
(b) R에서 $m_d$를 읽는다.
(c) R에서 $p_r$을 읽는다. 만일 $p_r$ = 0, (e)로 가고 그렇지 않으면 (d)로 가라. (만일 $p_r$ = 0이면, r은 출발노드이거나 아직 tree구축 과정에서 목표노드에 도착하지 않았다.)
(d) $P_d$에 $p_r$을 넣는다.; r = $p_r$로 설정하고 (c)로 간다.
(e) 결과를 도출한다.; $m_d$, d 그리고,$P_d$에 있는 모든 요소들
(f) $P_d$에 있는 모든 내용을 지우고 (a)로 간다.

## Moore와 Dijkstra와 D'Esopo Algorithm의 차이

Moore와 D'Esopo 알고리즘의 기본적인 차이는 작업표(L)로의 등록 방법에 있다.

Dijkstra 알고리즘은 다른 알고리즘과 주로 작업표(L)에서 NEXT 노드를 선택하는 방법에서 차이가 나고, 2b(ii)단계에서 L에 노드를 중복하여 등록하는 것이 허용되지 않는다.

Moore 알고리즘에서는 2(b)(ii)단계에서 어떤 노드 k가 L에 등록될 때, 그 L의 바닥에 등록된다. 3(b)단계에서 L로부터 선택된 NEXT 노드는 작업표 L의 맨 위에서 선택된다.

D'Esopo 알고리즘에서는 L은 2개의 부분(상위와 하위)으로 구분된다. 2(b)(ii)단계에서 만일 노드 k가 이전에 L에 등록된 적이 없다면 하위부분의 맨 아래에 등록된다.: 만일 노드 k가 이미 L에 등록되어 있다면 등록은 이루어지지 않는다. 그리고 k가 NEXT 노드로 설정되었었다면 k는 상위부분의 맨 아래에 등록된다. 3(b)단계에서 L로부터 선택된 NEXT 노드는 작업표 L의 맨 위에 있는 노드이고 상위의 맨 위에서부터 시작하여 상위부분이 비면 하위부분으로 점차 내려가며 진행된다. 따라서 이미 고려되었던 등록되어 있는 임의의 노드는 다시 고려될 것이다. D'Esopo 알고리즘에서 사용되는 배열 S는 노드들의 상태를 나타내며 작업표 L로의 등록에 관여한다. 만일 노드 k가 tree구축과정에서 아직 도착되지 않았다면 즉, L에 등록되지 않았다면, $s_k$ = -1이다. 만일 k 노드가 L에 현재 있다면, $s_k$ = 0. 그러나 등록된 적이 있다면 $s_k$ = 1이다. $s_k$ = 1이고 k가 2(b)단계에 따라 다시 L에 등록될 필요가 있을 경우, 다음의 상위부분에 재등록된다. 즉, $s_k$ = 0으로 설정된다.

Dijkstra 알고리즘의 경우에는 2(b)(ii)단계에서 노드 k는 Moore 알고리즘에서처럼 L의 맨 아래에 등록된다. 그러나 3(b)단계에서 선택된 NEXT 노드는 출발지 h에 가장 가까운 노드가 된다. 즉, L에 존재하는 모든 노드 중에서 가장 작은 값을 갖는 $m_k$를 만족시키는 노드이다.

다른 두 알고리즘에 비해 Dijkstra 알고리즘의 가장 큰 장점은 단일통과(once through) 방법이다. 출발노드 h에서 각 노드로의 tree를 구축할 때 각 노드 또는 각 링크를 연속적으로 지나갈 때 단 한번 고려된다. 그러나 이 장점은 고려될 다음 노드, NEXT를 찾을 때 작업표 L에서 시간소비가 많은 탐색법을 사용하여야만 얻을 수 있다.

Moore와 D'Esopo 알고리즘에서는 NEXT 노드(다른 노드들에 대한 전노드)가 고려된 후에 그 NEXT에 대한 더욱 비용이 적은 경로가 발견될 수 있을 것이다. D'Esopo알고리즘의 경우, 배열 S를 다루기 위해 요구되는 메모리의 증가를 처리하기 위해, 적은 비용의 경로가 계속 발견되는 모든 NEXT로 처리되는 노드는, tree의 확장이 더 진행되기 전에 재고되어진다. 그러나 도로망이 커질 수록 Dijkstra 보다 Moore나 D'Esopo 알고리즘이 많은 반복으로 인해 더 많은 시간이 소모되는 단점이 있다.

## Dijkstra 알고리즘의 개선

각 알고리즘에서 속도 향상과 메모리 요구 량의 최소화를 위해 많은 새로운 생각과 방법들이 적용되어 왔다.

① 도로망 입력에서의 개선

도로망은 이상적으로 링크와 노드의 크기가 가능한 작아야 하고 그 도로망의 목적에 적합해야 한다. 군더더기 링크와 노드는 미리 제거되어야 한다. 말단링크(끝노드를 가지고 있지만 끝노드가 다른 링크의 시작노드로 작용하지 않는 링크; dead end link)는 제거되어야 한다. 만일 꼭 필요하지 않다면, 한 교차로(Intersection)에 2개의 노드번호를(그 중 하나는 길이가 0인 연결링크와 연결되어 있는 더미노드)부여하는 것은 피해야 한다. 또한 링크 중간의 더미노드와 일방도로(one way)의 존재하지 않는 방향을 표현하는 것은 자제해야 한다.

통행존을 포함하여 도로망 노드의 번호를 매기는 것은 중간에 공백 없이 이루어져야 한다. 그렇게 함으로써 초기화 과정에서 시간을 절약할 수 있고 적은 컴퓨터 메모리를 사용할 수 있다. 또한 도로망에서 통행존 그리고 노드간의 구별을 쉽게 한다. 이때 실재 노드 번호는 $N_{cent}$ + 1에서 시작한다. 따라서 2(b)(ii)단계에서 노드 k는 $k > N_{cent}$이면 L에 등록된다. 이렇게 하는 것이 L에 등록되는 수를 줄일 수 있고 또한 경로가 통행존을 통과하는 것을 방지할 수 있다.

② NEXT의 선택

초기화 단계의 정확한 특징은 2단계, 2(a)단계, 2(b)(ii)단계처럼 고려되는 특정 알고리즘에 따라 다르다. 그러나 2(b)단계와 2(b)(ii)에서 NEXT 노드를 시작노드로 가지는 링크를 선택하는 것은 대부분의 알고리즘에서 동일하다. Dijkstra 알고리즘에서는 시작노드로 NEXT를 가진 링크의 식별은 포인터 배열 F(포인터 배열의 i번째 요소 $f_i$는 시작 노드로써 i를 가진 첫 번째 링크를 나타낸다. 단, 링크가 시작노드 순서로 정렬되어 있는 경우)를 설정함으로써 쉬워진다. 배열 F는 도로망표에서 링크의 시작노드의 배열로써 사용될 수 있다. 그렇게 함으로써 메모리 요구량을 줄일 수 있다. 시작노드 NEXT를 가진 링크는 도로망 table에서 $f_{NEXT}$에서 $f_{NEXT+1}$ - 1까지 번호가 매겨진 링크이다.

③ 작업표에서의 개선

Dijkstra 알고리즘에서 메모리 요구량을 줄이거나 컴퓨터 처리 시간을 줄임으로써 개선하려는 노력은 주로 작업표 L의 크기와 등록 방법의 수정에 집중되었다.

Moore와 D'Esopo 알고리즘의 경우에 처리 속도가 L의 크기에 의존하지 않는다. 단 예외는 초기화에서 0의 값들이 들어가거나 다른 단계들이 다른 알고리즘에 의해 결정되는 경우이다.

앞에서 나온 알고리즘의 2(b)(ii)단계에서 등록은 table에서 다음에 나오는 적당한 빈칸에 되고, 3(b)단계에서는 노드들이 작업표의 맨 위에서 선택된다. 기본적으로 이러한 알고리즘에서 필요한 것들은 L이 충분한 크기가 되어야 한다는 것이다. Dijkstra 알고리즘에서 비록 L에 등록을 고려하는 기본적인 상황이 Moore 알고리즘과 같지만 컴퓨터 처리 속도는 L의 크기에 많이 의존한다. 왜냐하면 3(b)단계에서 NEXT를 찾고 고려될 NEXT 노드로서 선택될 가장 h에 가까운 노드를 결정할 필요가 있기 때문이다. 따라서 L을 관리하는 여러 가지 방법을 사용함으로써 Dijkstra 알고리즘을 수정하려는 많은 노력이 있어 왔다.

Moore와 Dijkstra 알고리즘의 일반적인 모형에서는 개개의 노드들이 L에 몇 번이나 등록될지 알 수 없기 때문에, L의 크기는 반드시 미리 결정되어야 한다.; 간단한 방법은 단순히 빈 공간을 없애기 위해 L에서 지워진 노드에 등록을 대체시키는 것이다. 다른 방법은 미리 최대 L크기를 결정하고 이 최대 크기에 도달할 때마다 빈 공간이 남지 않도록 모든 현재 등록들을 빈 공간에 대체시키는 것이다. 그러나 Moore와 Dijkstra 알고리즘간에는 2가지 기본적 차이가 있다. 그 차이는 Dijkstra 알고리즘은 사용되는 L의 형태의 차이에 의하여 더욱 융통성 있게 된다.

- Moore알고리즘에서는 L에 등록되는 노드의 순서가 중요하다. 왜냐하면 그 순서가 NEXT 노드로 사용될 노드 순서를 결정하기 때문이다. 그러나 Dijkstra 알고리즘의 경우 그렇지 않다. 왜냐하면 L에 등록된 모든 노드들은 NEXT 노드로 사용될 h에 가장 가까운 노드를 결정하기 위해 한번 처리되기 때문이다.
- Moore 알고리즘에서는 각 노드들이 NEXT노드로 사용되고 그 횟수가 L에 등록된다. 그러나 Dijkstra 알고리즘에서는 각 노드가 단 1번 고려된다.

첫 번째 차이점에서 중요한 것은 다음과 같다. Moore알고리즘에서는 노드의 L 등록 순서가 유지되어야 한다. 그러나 Dijkstra 알고리즘에서는 노드가 L의 빈 위치에 순서에 상관없이 등록되고 특히 고려되고 있는 노드가 NEXT로서 바로 전에 제거된 위치에 등록된다. 그래서 빈 공간을 채우기 위해 따로 L에 등록들을 대체시킬 필요가 없다.

Dijkstra 알고리즘에서 L에 이미 등록되어 있는 노드가 첫 번째 등록이 고려되기 전에 다시 등록될 수 있다. 그러한 경우에 특별한 단계가 이루어지지 않는다면, 그 노드는 똑같은 비용을 가지고 1번 이상 NEXT로 고려될 수도 있다. 임의의 노드 k는 알고리즘 2(b)(ii)단계에서 만약 $m_k = \infty$이면 등록시킴으로써 1번 이상 L에 등록

되는 것을 막을 수 있다.

노드가 단 1번만 NEXT로 고려되는 것이 필요하다는 사실이 갖는 2가지 장점은 다음과 같다. NTREE(NEXT로 고려된 노드의 수)를 기록한다.

(i) 만일 필요할 경우 다음 사항을 가정한다. h가 고려되고 있으면 NTREE는 '0'이고 NTREE = $N_{nodes}$ - $N_{cent}$일 때 tree구축 과정이 중지된다. 이 것은 모든 실제 노드들이 선택되었다는 것을 의미하고, 따라서 L을 비울 필요가 없다.

(ii) h로부터 비용의 증가 순서로 배열된 노드에 대한 기록들이 유지된다.

## 2.4 모형들의 비교

Dijkstra와 Moore와 D'Esopo 알고리즘을 비교할 때, 그 방법들이 비교될 수 있는 기초가 다를 뿐만 아니라 각각에 대해 다양한 변형들이 나와 있다. 배정문제를 다룬 초창기에는 메모리 요구량 최소화를 매우 중요시했고, 메모리 제한 때문에 Dijkstra 알고리즘에서 순서화 링크 리스트법(ordered link lists)과 박스 정렬법(box sort method)의 사용이 어려웠다.

Dijkstra 알고리즘은 Moore와 D'Esopo 알고리즘에 비해 많은 장점을 가지고 있다. 즉, 각 노드는 단 한번 NEXT로 고려되기 때문에 h로부터 비용의 증가의 순서로 된 노드의 목록이 tree가 구축되자마자 나올 수 있다. 이 목록은 h로부터 다른 노드까지의 통행이 단일 통과(single once through pass)로 가로망에 통행을 부과할 수 있게 된다. 이 방법은 각 목적지까지의 통행을 따로따로 배분하는 것보다 효율적이다. Moore와 D'Esopo 알고리즘에서 그러한 목록을 만들려면 각 노드들이 마지막으로 고려되는 순서에서 목록의 유지를 가능하게 하는 추가적인 프로그램이 필요하다.

가장 일반적으로 tree 구축 절차는 CPU시간으로 비교된다. 그러나 이러한 CPU시간은 컴퓨터에 따라, 프로그램 입력양식에 따라, 그 도로망의 특징뿐만 아니라 컴파일러와 언어에 따라 다르므로 확실히 정의하기 힘들다. 프로그램 코딩에서의 차이만 본다면 Moore 알고리즘이 가장 간단하다. 세 가지 알고리즘 모두에서 전 링크가 작업표 L에 등록되는 것보다 전노드가 등록되는 것이 더욱 빠르다. Dijkstra 알고리즘은 순서화 링크 리스트와 박스 정렬법이 사용되어지면 더욱 복잡해진다. 그러나 이 기법들은 가장 효율적인 Dijkstra 알고리즘을 만들게 한다. 알고리즘의 효율성에 많은 영향을 주는 도로망의 특징은 도로망의 크기이다. 또한 링크와 노드의 수와 링크 비용의 공분산과 평균에 의해서 도로망의 형태와 노드당 링크의 수가 영향을 받는다.

CPU시간에 있어서 Dijkstra알고리즘과 Moore, D'Esopo 알고리즘의 기본적 차이는 Dijkstra 알고리즘에서의 작업표 L의 상대적으로 더욱 복잡한 처리와 Moore,

D'Esopo 알고리즘 고유의 반복처리의 정도에 의해 소모된 시간에 있다. Moore, D'Esopo 알고리즘의 경우, 어떤 노드까지 발견된 첫 번째 경로는 링크의 최소수로 이루어진 효율적인 경로이다. 따라서 노드가 NEXT로 단 한번 고려될 확률은 최소 비용경로가 가장 적은 링크 수를 가진 경로와 같을 확률이다. 이것은 링크 비용의 공분산과 관계되고 도로망의 크기에 따라 줄어들 것이다.

Van Vliet(1978)은 5개의 실제 도로망에 기초하여 CPU시간을 기준으로 세 가지 tree 구축알고리즘을 상세히 비교하였다. 그는 (박스 정렬)Dijkstra 알고리즘이 적은 (정수) 링크비용을 가지고 높은 공분산을 가지며 큰 도로망에 가장 적합하다고 결론 지었다. 이 알고리즘은 평균 링크 수의 노드 수에 대한 비율이 0.02보다 적을 때 Moore, D'Esopo 알고리즘보다 적은 CPU시간이 필요하다.

## 2.5 회전 제약과 금지를 고려하는 Vine 구축 알고리즘

회전금지나 회전제약 및 지체를 고려하는 것은 교차로를 노드만으로 표현한 일반적인 도로망에서는 적합하지 않다. Caldwell(1961)은 기본 도로망을 매우 세밀한 도로망으로 전환시킴으로써 회전금지와 양보회전을 적용할 수 있는 방법을 설명하였다. 전환된 도로망에서는 각 노드는 원래 도로망의 링크를 구성하고 또 각 링크는 허용된 회전경로가 된다. TRRL의 도로망의 정의 방법에서처럼, 이 접근 방법은 정상적 노드/링크 법에 의해 정의된 기존의 대규모 도로망에는 적합하지 않다. 대규모 도로망에서는 특별할 처리가 요구되는 교차로만 선택하여 이러한 교차로에 개개의 회전경로를 보장하는 링크를 사용한 보조 도로망이 포함된 도로망으로 확장하는 것이 좋다.

도로망을 명확히 확장시키지 않고 회전금지와 양보회전을 처리할 수도 있다. 회전금지와 회전에 따르는 제약 및 지체를 나타내기 위해서는 완전히 분리된 표를 사용하거나 아니면 보통 도로망표를 확장하여 사용할 수 있다. 일반 도로망표를 확장하는 경우, 링크(i,j)에서 링크(j,k)로 회전이, 즉 i -> j -> k로 회전금지가 되어 있다면, 링크(j, k)에 대한 도로망 자료는 노드 i와 회전 제약 $P_{ijk}$까지 포함되게 확장되어야 한다. 만일 회전제약이 통행배정 다음에 수정되어야 한다면 수정을 수행하기 위해 사용될 형태를 지시할 지시자(indicator)는 반드시 저장되어야 한다. 회전제약을 적용하기 위해 분리표를 사용하더라도 같은 자료가 저장되어야 한다. 회전제약의 수에 대한 자료는 저장되어야 한다. 그러나 그러한 회전을 찾고 각각의 회전제약을 계산하는 데 소요되는 시간의 증가를 감수해야 한다. 회전제약이 명확히 확장된 도로망 없이 고려할 때 tree구축 알고리즘은 적절하지 않다. tree 구축알고리즘에서는 만일 h에서 j까지 최소비용경로에서 링크 (i,j)가 노드 j의 전링크라면 j를 시작 노드

로 해서 j를 거쳐 끝노드 k까지 가는 가장 비용이 적게 드는 경로는 마찬가지로 (i,j)를 거친다. 그러나 회전제약이 발생한다면 위의 경우가 항상 성립하지는 않는다. 노드 k까지 링크 (i',j)를 거쳐서 가는 비용이 (i,j)를 거쳐서 가는 비용보다 더 적은 경우, (i,j)에서 (j,k)로의 회전과 관련된 제약이 (i',j)에서 (j,k)로의 회전의 경우보다 더욱 클 것이다. 따라서 회전제약이 발생한다면 h로부터 j를 경유하는 최소비용경로에 j에서 최소비용경로가 만나거나 가로지를 수 있는 vine 구축알고리즘이 그러한 가능성을 완전히 배제하는 tree 구축알고리즘보다 더 요구된다.

## 2.6 Vine 구축 알고리즘

각 노드에 한 개 이상의 경로를 결정하는 알고리즘들을 vine 구축 알고리즘이라 한다. 이에 비해 tree는 각 노드에 단 한 개의 경로만을 허용한다. vine 구축알고리즘의 가장 간단한 형태는 tree 구축알고리즘의 확장된 형태이다. 예를 들어, 알고리즘이 다음 설명처럼 각 노드에 대해 두개의 전노드를 가질 수 있도록 수정되면 가장 간단한 vine 구축을 수행할 수 있다.

tree표 R을 각 노드에 대해 두 번째 전노드 $p'_i$와 이에 따른 비용 $m'_i$를 갖도록 확장한다.

1(a)단계에 추가 : $p'_i = 0$, 그리고 $m'_i = 0$으로 초기화.

2단계를 다음 내용으로 대체 :

2단계 : (NEXT의 고려; 이 과정은 R을 갱신하는 것과 L에 등록하는 과정을 포함한다.

(a) L로부터 NEXT를 제거

(b) 각 링크(NEXT, k)에 대해; R로부터,

$m_{NEXT} + c_{NEXT} < m_k$이면, $m'_k = m_k$, $p'_k = p_k$, $m_k = m_{NEXT} + c_{NEXTk}$, $p_k$ = NEXT로 설정, 그리고 k를 L에 등록;

만일 $m_k \le m_k + c_{NEXTk} < m'_k$ 이면, $m_k = m_k + c_{NEXTk}$, $p'_k$ = NEXT로 설정;

그렇지 않은 경우는 아무런 작업도 하지 않는다.

최소비용경로는 원래의 4단계를 사용하여 출력할 수도 있다. h로부터 노드 d까지 두 번째 전노드를 경유한 경로를 출력하기 위하여 그 과정은 변경해야 할 4(b)단계를 제외하고 최소비용경로와 같다.

(i) R에서 $m'_r$를 읽음; $m'_r = 0$이면 d에 대한 전노드는 없다. 적절한 안내문을 출

력하고 (a)로 돌아감.

(ii) R에서 $p'_r$을 읽음; $P_d$에 $p'_r$를 입력; r = $p'_r$ 그리고 (c)로 분기

그러나 이 방법에서는 $p'_d$를 경유해서 d까지 가는 경로가 h에서 d까지 가는 두 번째로 비용이 적은 경로임을 보장할 수 없다. $m'_d$보다 적은 비용으로 h에서 d까지 가는 경로가 많이 있을 것이다. 만일 두 번째로 비용이 적은 경로가 필요하다면 $m_k$나 $m'_k$가 바뀔 때마다 k가 L에 등록되어야 하고 Dijkstra 알고리즘에서는 $m'_k$가 NEXT 노드를 결정하기 위한 탐색과정에 포함되어야 한다. 이러한 문제를 해결하기 위해 다음과 같은 방법이 사용된다.

## 2.7 출발노드에서의 최소비용경로 결정에 대한 Vine 구축 알고리즘

임의의 링크 (j,k)로의 임의의 회전이 제약되었거나 회전금지가 있다고 가정하면 링크 (j,k)에 대한 도로망표는 노드 j 또는 노드 i에 관계된 전노드와 그에 따른 제약 $P_{ijk}$의 쌍의 목록을 가지고 있다.

위에서 언급된 Moore와 Dijkstra와 D'Esopo 알고리즘 모두 사용될 수 있고 단지 tree table R에 저장될 자료의 필요한 형태로의 개조와 함께 NEXT 노드 선택방법과 초기화 과정의 수정이 필요하다. 그러나 본 연구에서는 기본적으로 Dijkstra 알고리즘을 이용하여 vine 구축을 수행토록 한다. tree표는 임의의 노드 k에 대해 전노드 $p_k$와 h로부터 현재까지의 비용 $m_k$ 뿐만 아니라 $pp_k$로 표시되는 $p_k$에 대한 전노드까지 포함한다.

**1단계 : (초기화)**

(a) R에서; 모든 노드 i에 대해 $p_i$ = 0, $m_i$ = ∞로 설정한다.; $m_h$ = 0으로 설정.

(b) 선택된 알고리즘에 따라 L을 초기화한다.

(c) 링크 (h,j)에 대한 모든 시작노드 j에 대해; R에서 $p_j$ = h, $m_j$ = $c_{hj}$로 설정한다.; L에 j를 등록한다.

(d) NEXT = h

**2단계 : (NEXT 노드의 고려. 이 단계는 R과 L의 갱신을 포함한다.)**

(a) Dijkstra알고리즘에 따라 L에서 NEXT를 제거

(b) 각 링크 (NEXT, j)에 대해, 그런 링크가 없으면 3단계로 분기하고 그렇지

않으면 ;

(1) $m'_j = m_{NEXT} + c_{NEXTj} + P_{rNEXTj}$를 결정, 단 r = $p_{NEXT}$

(2) 각 링크 (j,k)에 대해, 만일 그러한 링크가 없다면 (1)로 돌아가고 그렇지 않으면;

(i) $m''_k = m'_j + c_{jk} + p_{NEXTjk}$를 결정

(ii) $m''_k > m_k$이면 그냥 지나가고 그렇지 않으면 ;

R에서 $pp_k$ = NEXT, $p_k$ = j, $m_k = m''_k$로 설정하고;

k를 L에 등록.

(c) 3단계로 감

**3단계 : (L로부터 고려될 NEXT 노드를 선택)**

L로부터 Dijkstra 알고리즘에 따라 NEXT로 고려될 수 있는 다음 노드를 선택한다. 그리고 2단계로 간다. 만일 NEXT가 없다면 4단계로 간다.

**4단계 : (최소비용경로와 비용의 결과)**

구축 결과는 일반 tree 알고리즘의 형태와 같다.; h로부터 d까지의 경로에서 노드의 목록은 $p_d$에 저장된다.

임의의 종착지 d에 대하여;

(a) r = d로 설정, R에서 $m_r$을 읽음

(b) R에서 $p_r$과 $pp_r$을 읽음

(c) 만약 $pp_r$ = 0이면, $P_d$에 $p_r$을 입력하고 (d)로 감,

$pp_r$ = h이면, $p_r$과 $pp_r$을 $P_d$에 입력하고 (d)로 감,

그렇지 않으면, $p_r$과 $pp_r$을 $P_d$에 입력하고 r = $pp_r$로 설정후 (b)로 감.

(d) $P_d$에 있는 모든 요소와 $m_r$과 d를 출력.

어러한 알고리즘을 바탕으로 하여 본 연구에서는 한국 철도망과 중국 및 러시아 철도망을 통하여 유럽 철도망과의 최단경로를 탐색하는 프로그램 개발을 착수하였다. 다음 장에서는 기존 GIS S/W의 ROUTING 기능의 효율성을 검토하기 위한 작업의 일환으로 ARC/INFO를 이용한 최단경로 탐색을 모색하였다.

# 3. GIS를 이용한 최단경로 탐색

현재 시판되고 있는 거의 모든 GIS S/W는 Routing이란 기능이 있어서 거리나 시간을 이용한 최단경로나 Traveling Salesman's Problem의 해를 구해주고 있다. 이러한 S/W의 Source Code가 공개되어 있지 않아 정확한 최단경로 탐색의 알고리즘은 알 수는 없으나 거의 모든 S/W들이 각자 고유의 신속한 처리 기능의 최단 경로 탐색 기능을 갖고 있다고 선전하고 있다. 그러나 현재 시판되는 모든 GIS S/W는 상기 2장에서 언급한 vine알고리즘을 처리하지 않는 것으로 분석되고 있으며 기존 교통계획 S/W의 최단경로 탐색 수준에 못 미치고 있는 실정이라 판단된다.

본 논문에서는 앞으로 본 연구 결과와 비교하기 위하여 우리나라와 중국과의 철도망의 최단경로를 ARC/INFO를 이용하여 구하였다. 그 결과는 출발지와 도착지의 선정에 따라 정해짐으로 LAP TOP COMPUTER를 이용하여 DEMO하기로 하였다.

북한의 철도 공개를 거부하는 경우 인천에서 중국해안의 도시로 BARGE를 이용하여 화물열차가 수송된다는 가정을 하여 최단경로 탐색을 모색하였다.

# 4. 결론

본 논문에는 우리나라 물류비를 저감하는 여러 대안들을 분석하는 과정의 일환으로 GIS를 응용하여 우리나라와 중국 철도의 최단경로탐색을 하는 PILOT STUDY를 소개하였다. 최단경로 탐색의 가장 최근에 개발된 알고리즘을 이용한 것 같지는 않지만 ARC/INFO의 ROUTING 기능을 이용하여 그 나름대로의 최단경로 탐색의 효용성을 보여주고 있음을 DEMO하였다. 궁극적으로는 현 교통계획에서 수행되고 있는 ROUTE 선택 기법들을 이용한 알고리즘이 현존하는 GIS S/W에 도입되어야만 정확하고 효율적인 최단경로 탐색을 모색할 수 있을 것이라 판단된다.

# A Comparing Studying of thc China Citics' Outsidc Transportation Modcls

Cao Zhongyong
( Shanghai Tiedao Univercity , Shanghai , 200333 )

On the basis of deeply analysing the historical course of formation and national conditions as well as economic environment of the China various kinds and different characteristics cities in several large districts , this thesis studies the present situation of China cities' outside transportation ways and their existing problems which are given a quantitative comparing analysis . For the purpose of promoting the economic development of these cities , this thesis proposes some opinions on perfecting direction and improving strength of the China cities outside transportation ways .

Kcywords:China , City , Transportation Modcl