기업 물류망 최적 설계 및 운영을 위한 알고리즘 설계 및 소프트웨어 구현 사례

한 재 현* · 김 장 엽** · 김 지 현* · 정 석 재*
*광운대학교 경영학부 · **연세대학교 정보산업공학과

A case study on algorithm development and software materialization for logistics optimization

Jae-Hyun Han* · Jang-Yeop Kim** · Ji-Hyun Kim* · Suk-Jae Jeong*
*School of Business, Kwangwoon University
**Dept. of Information & Industrial Engineering, Yonsei University

Abstract

It has been recognized as an important issue to design optimally a firm's logistics network for minimizing logistics cost and maximizing customer service. It is, however, not easy to get an optimal solution by analyzing trade-off of cost factors, dynamic and interdependent characteristics in the logistics network decision making. Although there has been some developments in a system which helps decision making for logistics analysis, it is true that there is no system for enterprise-wise's on-site support and methodical logistics decision. Specially, E-biz process along with information technology has been made dramatic advance in a various industries, there has been much need for practical education closely resembles on-site work. The software developed by this study materializes efficient algorithm suggested by recent studies in key topics of logistics such as location and allocation problem, traveling salesman problem, and vehicle routing problem and transportation and distribution problem. It also supports executing a variety of experimental design and analysis in a way of the most user friendly based on Java. In the near future, we expect that it can be extended to integrated supply chain solution by adding decision making in production in addition to a decision in logistics.

Keywords : Logistics Decision-making, Location & Allocation Problem, Traveling Salesman Problem, Transportation-Distribution Problem, Logistic Software Development

1. 서 론

물류 네트워크(Logistics network)는 물품 또는 서비스의 공급(창출) 지점으로부터 수요지점까지의 입지선정, 수·배송 및 차량 배달과 관련된 모든 기능과 활동의 집합으로서, 흐름의 단계와 특성에 따라 다양한 의사결정 문제들을 정의할 수 있다. 따라서 물류 네트워크의 형태는 기업 유형에 따라 다르며, 한 기업에서 여러 형태의 물류네트워크가 존재할 수 있다. 물류 네트워크의 운영은 조달, 생산, 수송, 저장, 판매, 재무 및 A/S 등 기업의 여러 기능들과 관계를 가지면서 비용과 고객서비스에 큰 영향을 미친다.

미국의 식품판매협회가 출판하는 연구보고서에 따르면, 기존의 전형적인 식품 물류네트워크의 설계와 운영을 개선함으로써 약 42일의 네트워크 소요기일을 단축할 수 있고, 연간 100억 불의 물류비용을 절감하여

† 교신저자: 정석재, 서울시 노원구 광운로 20 광운대학교 경영대학 경영학부
M · P: 010-7714-7811, E-mail: sjjeong@kw.ac.kr
2012년 9월 28일 접수; 2012년 12월 12일 수정본 접수; 2012년 12월 15일 게재확정

약 41%의 재고를 감축할 수 있는 것으로 예측된 바 있다.

물류 네트워크의 설계 및 계획은 창고의 수와 위치, 고객 및 제품 할당, 수송수단, 차량경로 등 다양한 의사결정문제를 포함하며, 때로는 얘기치 않은 돌발 상황을 대비한 비상계획까지도 필요로 한다. 물류네트워크 설계 및 계획을 위한 의사결정 문제는 상호 밀접한 관계를 유지하고 있을 뿐만 아니라, 많은 확률적 요소가 존재하며, 관련 비용들 사이에 복잡한 trade-off 분석과 여러 계량적 요소들의 평가를 필요로 하기 때문에 통합적 차원에서 최적 해를 구하는 것이 쉽지 않으므로, 현장에서 매우 어려운 컨설팅 영역 중 하나로 인식되고 있다.

이에, 최근 많은 기업들은 물류 및 수·배송 계획(수·배송 : Transportation - Distribution)에 대한 관심이 증대되고 있으며, 물류 계획을 효과적으로 지원해 줄 수 있는 솔루션에 대한 관심이 증대하고 있다. 물류계획은 지금까지 기업 운영의 주된 영역을 차지하고 있던 자재소요계획(MRP : Material Handling Planning), 전사적 자원 관리(ERP : Enterprise Resource Planning) 등의 개념을 뛰어 넘어 원자재의 공급부터 이를 이용하여 생산된 제품을 소비하는 최종 고객까지를 포함하는 방대한 기업 활동 영역을 모두 포함하고 있다. 기업 환경이 국경을 초월하여 E-biz 환경을 이루어 갈수록 효율적인 물류 계획에 대한 중요성은 커져 가고 있으며, 국·내외의 유수한 기업에서 앞 다투어 물류 계획을 위한 다양한 솔루션들을 도입하여 조금씩 그 효과를 보고 있다. 그리고 최근 IT(Information Technology)의 급속한 발전과 컴퓨터의 강력한 파워를 배경으로 한 다양한 물류 솔루션들은 기업들에게 강한 인상을 주고 있다.

그러나 물류 관련 솔루션 개발에 대한 기업의 관심이 증대되고 있지만, 실제 이를 충분히 활용하고 분석할 수 있는 전문가의 부재가 심각한 상황이라고 하겠다. 실제 E-biz 전문가 양성에 있어서, 이처럼 물류와 관련된 문제를 모델링하고, 실제 문제를 해결할 수 있는 능력을 갖추는 것은 무엇보다 중요하다고 본다. 따라서 본 연구는 실제 현장에서 물류망 설계 및 분석을 의사 결정하는 데 지원할 수 있는 솔루션을 개발하는 데 그 목적이 있다.

2. 물류 의사결정 문제 정의

본 연구에서는 물류망 운영 및 설계에 있어 핵심적 의사결정 문제로 인식되고 있는 위치 할당문제, 외판원 문제 및 이를 확장한 차량 경로문제, 수·배송 문제에 대한 알고리즘 및 소프트웨어로 구현하였다. 총 4개의 문제에 대해서 사용자가 직접 모델링하고, 그 결과를 분석해 볼 수 있도록 Java 언어로 모델링 되었으며, 각 문제에 대한 정의는 다음과 같다.

2.1 위치할당문제 (Location-Allocation Problem)

입지할당의 기본목적은 다수의 입지후보지로부터 최적의 입지를 선정하고 운송량을 할당하는 것이며, 제약조건은 통상 설비에 관련되는 투입, 공정 및 산출요인과 운송 구조를 고려하여 설정된다. 예를 들면, 원자재 투입요소는 석유화학, 제철 등의 입지선정에 주요한 제약조건이며, 지가, 설비가격 또는 운송비용은 기업의 재무조건에 의해 제약된다. 또한 산출에 있어서도 시장의 규모, 수요, 소비자의 기호 및 리드타임 등은 입지할당의 중요한 제약요인이 된다. 일반적으로 입지이론은 1909년 베버(Weber, Alfred)의 최소운송비용 이론으로부터 시작되었다.(Friedrich, 1929). 여기서 베버는 자원분산지와 한 소비시장만을 고려한 단순 상황하의 최적입지의 탐색을 시도하였다. 복잡한 현실을 단순화한 입지 의사결정이론에서 입지의사결정요인은 운송비, 노무비, 응집력(Agglomeration force)의 세 가지로 규정하고 각 요인이 입지에 미치는 영향을 분석하였다. 베버는 이 세 가지 요인 중 가장 기초가 되는 요인을 운송비로 보고 운송비만을 고려한 입지를 우선적으로 선정한 후에 노무비와 응집력을 고려하여 이를 조정하는 방식을 취하였다. 하지만 다른 중요한 요인에 따라 시장 영역, 대체의 원리에 의한 입지, 기하학적 방법을 이용한 입지, 수송 선형계획 모형을 이용한 입지할당 의사결정, 목적계획법을 이용한 입지할당 모형 등이 있다.

2.2 외판원 문제 (Traveling Salesman Problem)

순회 판매원 문제는 N개의 도시가 주어질 때, 한 도시로부터 출발하여 모든 도시를 한 번씩 방문한 다음 출발했던 도시로 되돌아오는 가장 짧은 경로를 찾는 문제로 상당이 많은 계산을 필요로 하는 NP-Hard 문제이다. 그러므로 순회 외판원 문제는 유명도와 고난이도로 인해 어떤 새로운 공간 탐색 기법이 고안되면 문제에 대한 실험이 이루어지지 않고는 제대로 평가받을 수 없을 정도로 큰 영향력을 가진 문제이다.

또한 순회 외판원 문제는 물류, 택배, 스쿨버스 등의 다양한 차량 라우팅 문제들의 원형이고 VLSI 칩의 공정, 컴퓨터 보드의 생산공정 등에서도 작업의 효율을

결정하는 데 중요한 역할을 한다. 또한 X선 결정학 문제 등 다양한 분야에서 순회 외판원 문제와 연관되어 있다. 많은 연구자들이 몇 십 개에서 많게는 수백만 개의 도시를 가진 순회 외판원 문제를 다양한 방법으로 근사 최적해를 찾으려고 노력하고 있다.

2.3 차량경로문제 (Vehicle Routing Problem)

차량경로문제란 차고지에서 출발하여 여러 곳에 흩어져 있는 알려진 수요량을 가진 고객들의 수요(배달량)들을 만족시키면서 출발지로 복귀하는데 소요되는 최단이동경로(최소이동비용)를 구하는 문제이다. 일반적으로 목적함수는 최단이동거리(최소이동비용)로 설정되며 제약조건은 차량별 용량, 차량별 이동거리, 가용차량 대수, 고객방문시간의 제한 등이 된다. 전통적인 차량경로문제는 NP-Hard 문제로 알려져 있는데 문제에서 주어진 지점 수가 증가할수록 계산시간이 기하급수적으로 증가하게 되므로 최적화 기법을 이용하여 해를 구하는 방법 대신에 발견적 기법(Heuristic)을 적용하여 근사해를 찾는 연구가 활발히 진행되어 왔다. Danzig와 Ramser가 최초에 차량경로문제를 제안한 이래로 Clarke와 Wright(1964)는 Saving기법을 제안하였는데 Saving기법이란 두 지점을 단독으로 배차하였을 때 보다 두 지점을 하나의 경로로 묶었을 때 발생하는 거리절약이 높은 지점 쌍 순서부터 차량의 경로에 포함시키는 방법을 말한다.

Gillet와 Miller(1974)에 의해서 제안된 Sweeping기법은 극좌표 상에서 한 지점을 무작위로 선정하여 그 지점을 씨앗지점(Seed node)으로 선정하고 출발지인 본점(Depot)을 축으로 하여 반 시계방향 혹은 시계방향으로 돌면서 차량의 용량이 허락하는 한도까지 지점을 경로에 포함시키는 방법이다. Saving기법과 Sweeping기법은 계산과정이 단순하고 계산속도가 빠른 장점이 있는 반면에 해의 질적인 측면에서는 그리 큰 장점을 지니고 있지 못하다. 기존의 발견적 기법 이외에 최근에는 메타휴리스틱(Meta Heuristic)을 이용한 차량경로문제에 대한 접근방법이 많이 시도되고 있다. 특히 타부서치(Tabu Search)를 이용한 방법과 유전알고리즘(Genetic Algorithm)을 적용한 방법에 관해서 활발히 연구가 진행되고 있는데, 해의 질적인 측면에서는 유전알고리즘에 비해서 타부서치가 우수하다고 알려져 있다. 차량경로문제의 해법에 관한 연구 이외에 차량경로문제의 모델링에 관한 연구도 많이 진행되었는데 전통적인 차량경로문제에서 변형된 출발지가 여러 곳인 Multi Depot 차량경로문제(Laporte et al., 1988), 한 수요지점에 다회방문을 허락하는 VRPMT (Multi-Trip Vehicle Routing Problem)(Brandão, 1998), 고객의 희망방문시각의 상한과 하한이 존재하는 VRPTW(Vehicle Routing Problem with Time Window)(Cano Sevilla와 Simo´n de Blas, 2003), 한 지점에서 하역과 적재가 모두 발생하는 PDP(Pick up Delivery Problem)(Kalantari et al., 1985), 그리고 각기 용량이 다른 차량의 최적 조합으로 해를 찾는 HVRP(Heterogeneous Fleet Vehicle Routing Problem)(Taillard, 1999) 등이 있다.

2.4 수송문제(Transportation Problem)

m개의 공급지(supply center)와 n개의 수요지(demand point)가 있다. 공급지 i에서 수요지 j로 단위량을 수송하는데 드는 비용은 c_{ij}이다. 공급지 i에서 공급할 수 있는 양(supply)은 a_{ij}이고, 수요지 j에서 필요로 하는 양(demand)은 b_{ij}이다.

x_{ij}를 공급지 i에서 수요지 j로의 수송량이라고 하면 다음의 선형계획모형이 가능하다.

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} \le a_i \quad (i = 1, 2, ..., m)$$

$$\sum_{i=1}^{m} x_{ij} \ge b_j \quad (j = 1, 2, ..., n)$$

$$x_{ij} \ge 0 \quad (i = 1, 2, ..., m, j = 1, 2, ..., n)$$

3. 각 문제별 적용을 위한 알고리즘 구현

3.1 위치할당문제 (Location-Allocation Problem)

공장 위치문제와 관련한 최근의 방법론은 다음과 같다.

응용수학과 컴퓨터를 통한 접근방법은 개념적인 접근이라기보다는 수학적 모델 그 자체에 가깝다. 여기서는 단일 공장, terminal, warehouse 또는 소매·서비스 지점의 위치선정 하는데 많이 사용되는 모델을 이용하기로 한다.

위치 할당 문제를 푸는 방법에는 무게중심 접근법(exact center of gravity), 격자 방법(grid method), 중심 방법(centroid method) 등이 있다(Sule, 2001).

만들어진 제품의 수요지역으로의 운반량과 모르는

곳에 위치한 공장에서의 자원 공급량 및 이에 따른 운송율을 고려했을 때 '공장의 위치는 어디가 적당할 것인가?'라는 문제를 풀기 위해 운송량과 운송율, 그리고 운반할 두 지점사이의 거리를 곱한 값을 모든 지역에 대해 구한 다음 총합을 구한다. 이 총합을 최소로 하는 값을 구해보고자 한다.

$$\min TC = \sum_i V_i R_i d_i \quad (1)$$

TC = 총 운송비용
V_i = 지역 i에서의 운송량
R_i = 지역 i로의 운송률
d_i = 입지할 공장의 위치와 지역 i 사이의 거리

공장 위치는 그 지점의 좌표에 대한 두 개의 방정식을 통해 결정하는데, 무게 중심 좌표는 다음과 같다.

$$\overline{X} = \frac{\sum_i V_i R_i X_i d_i}{\sum_i V_i R_i d_i} \quad (2)$$

$$\overline{Y} = \frac{\sum_i V_i R_i Y_i d_i}{\sum_i V_i R_i d_i} \quad (3)$$

$\overline{X}, \overline{Y}$ = 공장의 좌표
X_i, Y_i = 자원과 수요 지역의 좌표

거리 d_i를 구하는 공식은 다음과 같다.

$$d_i = K||X_i - \overline{X}| + |Y_i - \overline{Y}|| \quad (4)$$

*K = 한 단위의 좌표 인덱스를 일반적인 거리 측정값으로 변환시키기 위한 계수. 예) 킬로미터, 마일

해법 과정은 다음의 몇 가지 단계를 거친다.

[단계 1] 각 자원과 수요 지점에 대한 X, Y 좌표를 운송량과 선형적으로 변하는 운송율에 따라 결정한다.
[단계 2] 초기 위치를 무게중심 공식으로 대략적으로 구한다. 이때 거리 d_i는 생략한다.

$$\overline{X} = \frac{\sum_i V_i R_i X_i}{\sum_i V_i R_i} \quad (5)$$

$$\overline{Y} = \frac{\sum_i V_i R_i Y_i}{\sum_i V_i R_i} \quad (6)$$

[단계 3] [단계 2]에서 얻은 $\overline{X}, \overline{Y}$와 식(4)를 이용해 d_i를 구한다.(이 때 K는 사용할 필요가 없다)
[단계 4] d_i를 식(2)와 식(3)에 대입해서 변경한 $\overline{X}, \overline{Y}$좌표를 푼다.
[단계 5] 변경한 $\overline{X}, \overline{Y}$좌표를 이용해 d_i를 다시 계산한다.
[단계 6] [단계 4]와 [단계 5]를 $\overline{X}, \overline{Y}$좌표가 더 이상 변하지 않거나 변경된 값이 너무 작을 때까지 반복한다.
[단계 7] 마지막으로, 식(1)을 사용해 최적의 입지에 대한 총 비용을 계산한다.

이 밖에도 다중의 새로운 설비를 배치하기 위하여 기존의 고객 위치가 $x-y$로 나타날 수 있을 때, 좌표를 설정하는 것은 쉬운 일이 아니다. 다시 말해 문제는 새로운 설비를 어디에 위치시킬 것인가와 어떤 고객이 어떤 설비에서 서비스를 받게 할 것인가 하는 두 가지 부분이다.
이러한 두 가지 문제를 동시에 풀기보다는 두 개의 일차원 문제로 나누어 풀려고 한다.
먼저 우리는 기존고객을 클러스터로 그룹화 하고 각 클러스터에 대하여 각 설비에 대한 최적 위치선정을 결정할 것이다. 최종적으로 모든 고객이 각 클러스터에서 새로운 설비에 할당되게 할 것이다.
하나의 설비를 적절하게 위치시키는 과정은 앞서 언급한 위치할당 해법에서 기술하였다. 따라서 여기에서 우리는 클러스터를 구성하는 과정을 배울 것이다. 우리가 구성하고자 하는 클러스터의 수는 우리가 위치시키고자 하는 새로운 설비의 수와 동일하다. 그 과정은 다음 단계를 따른다.

[단계 1] 가장 큰 가중치를 갖는 고객부터 시작한다. 첫 번째 설비는 이 고객의 좌표에 위치하도록 가정한다.
[단계 2] 다른 고객이 이 설비에 할당되는 비용을 구한다. 고객에 대한 이 비용은 고객과 새로운 설비 위치 간의 거리(직선이거나 유클라디안이거나)에 고객의 가중치와의 곱과 같다. 가장 큰 비용으로 고객의 좌표를 두 번째 설비에 대한 위치로 선정한다. 두 번째 설비에 대한 위치로써 최대의 비용이 드는 고객 좌표를 선택한다. 위치시켜야할 새로운 설비가 두 개 이상이면 [단계 3]으로 가고 그렇지 않으면 [단계 5]로 간다.
[단계 3] 고객을 각각의 지정된 새 설비 위치에 할당하는 것과 관련된 비용을 계산한다. 이 값들 중에 고객 관련된 비용의 최솟값을 선택한다.
[단계 4] 위치시키고자하는 새로운 설비의 숫자가 새로운 설비 위치만큼 얻을 때까지 [단계 3]을 반복한다.
[단계 5] 일단 모든 새로운 설비들에 대한 위치가 결

정되면, 거리를 측정하기 위해 사용된 기준에 근거하여 각 고객들을 가장 가까운 위치에 할당한다.

[단계 6] 각 고객의 클러스터를 서비스하기 위해 앞서 언급한 위치할당 해법 과정을 통해 최적의 설비위치를 찾는다.

3.2 외판원 문제 (Traveling Salesman Problem)

<표 1>의 도시 간 거리 매트릭스를 가질 때, A도시를 시작으로 하나의 도시를 한번 씩만 거치는 외판원 문제를 해결하기 위한 알고리즘 절차는 다음과 같다.

<표 1> 거리(비용) 매트릭스

도시	A	B	C	D	E
A	0	1	7	4	3
B	2	0	1	5	4
C	5	2	0	6	1
D	3	1	4	0	2
E	4	3	1	5	0

[단계 1] <표 2>와 같이 되돌아가는 것을 막기 위해 비용을 변환하여 제자리로 돌아오는 값을 무한대로 한다.

<표 2> 열 환산 결과

도시	A	B	C	D	E
A	∞	1	7	4	3
B	2	∞	1	5	4
C	5	2	∞	6	1
D	3	1	4	∞	2
E	4	3	1	5	∞

[단계 2] <표 3>과 같이 최초 각 열의 0의 값을 제외한 최소값을 찾아 그 열의 각 원소에서 그 수를 뺀다.

<표 3> 행 환산결과

도시	A	B	C	D	E
A	∞	0	6	0	2
B	0	∞	0	1	3
C	3	1	∞	2	0
D	1	0	3	∞	1
E	2	2	0	1	∞

[단계 3] 비용이 0이 아닌 각 셀의 페널티(penalty)를 찾는다. 만약 우리가 열 i와 행 j에서 비용이 0이 아닌 것을 경로 내의 연결을 형성할 경우 그 경로를 발전시키기 위해 열 i와 행 j에서 또 다른 요소를 사용해야 한다.

<표 4> 페널티 평가 : 주기 1

도시	A	B	C	D	E
A	∞	0 0	6	1 0	2
B	0 0	∞	0 0	1	3
C	3	1	∞	2	1 0
D	1	1 0	3	∞	1
E	2	2	1 0	1	∞

우리가 할 수 있는 최선은 이런 목적을 위해 열 i와 행 j로부터 셀의 다음 최소 비용을 사용하는 것이다. 그래서 0인 셀을 사용하지 않기 위한 최소 페널티는 열 i와 행 j에서 최소 요소의 합이다.

모든 0을 가지는 셀에 대한 페널티를 계산하고 관련된 셀의 왼쪽 모서리에 그것을 기록한다. 예제에서 결과들은 <표 4>에 나타내었다.

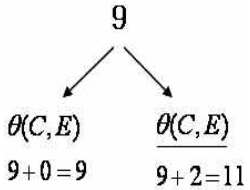
예를 들어, 첫 번째 0을 가지는 셀인 A-B 셀은 열 A에서 다음 최소비용은 B을 연결할 때 0이고, 행 B에서 열 C을 연결할 때 최소비용이 0이므로, 페널티는 0이 된다. 한편, B-A셀은 열 B에서 행 A를 연결하고, 행 A에서 다시 열 B를 연결하면 페널티는 0이 된다. 나머지 0을 가지는 모든 셀에 대해서도 동일한 과정을 거치게 된다.

[단계 4] 가장 큰 페널티를 가지는 0의 셀을 선택하며 우리는 그것을 열 h와 행 k에 있다고 가정한다. 따라서 그것은 셀 (h, k)로서 지정된다. 우리는 지금 그 문제를 다음과 같이 두개의 부분으로 나눈다.

$\theta(h, k)$: link(h, k)를 포함
$\theta(h, k)$: link(h, k)를 포함하지 않음

link(h, k)를 사용하지 않는 비용은 최소한 link (h, k) 내에서 관련된 페널티와 같다. 그래서 그 지점에서 더 멀리 진행하는 전체 비용은 비용에 cell(h, k)에서의 페널티를 합한 것이 된다.

우리의 예제에서 최대 페널티는 셀 (C, E)와 관련된다. 그래서 연결이 9인 것과 같이 (C, E)를 선택하는 것과 관련된 비용은 선택하지 않는 것에 반해 9+2=11이 된다. 이것은 관련 지점을 가지는 트리(tree)로 표현된다.



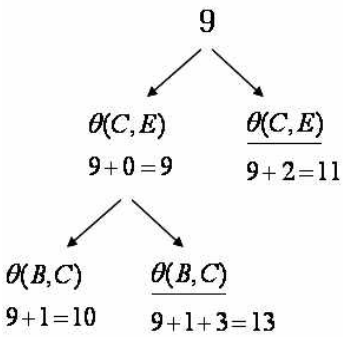
[단계 5] 최소 비용 지점을 가지고 멀리 계속한다. (C, E)가 연결로서 선택되기 때문에 고리를 만들어 감에 따라(다른 지점을 최소한 한 번 방문하지 않고 하나의 지점을 두 번 방문하는) 우리는 C에서 시작하거나 E에서 끝나는 또 다른 연결(link)을 가질 수 없다. 이것은 열 C와 행 E를 차후 고려사항에서 제외하기 위해 C와 E를 삭제함에 따라서 달성된다.

또한, 차후 경로에서 연결이 형성되는 것을 막기 위해 비용을 C와 E에서 ∞로 변경한다. 그 결과 <표 5>에 표현된다.

[단계 6] [단계 3]과 [단계 4]를 해법을 얻을 수 있을 때까지 진행한다.

이러한 과정을 계속하며 각 열과 행에서 하나의 0값을 가지지 않기 때문에 0들을 만들어야 한다. 셀들 안에서 0이 아닌 값을 가지는 경우 그러한 각 열과 행과 관련된 최소 요소를 그 열과 행에서의 모든 요소로부터 뺀다. 우리의 문제에서 해법 값을 1증가시키는 1은 열 E로부터 추출한다.

결과를 나타내는 <표 5>에서 각 0셀의 보유에 따라 페널티를 계산한다. 최대 페널티는 셀 (B, C)과 관련된다. 이와 같은 트리는 다음과 같다.



<표 5> 셀 (C, E)을 선택한 후에 남아있는 셀들

도시	A	B	C	D
A	∞	0	6	0
B	0	∞	0	1
C	1	0	3	∞
D	2	2	∞	1

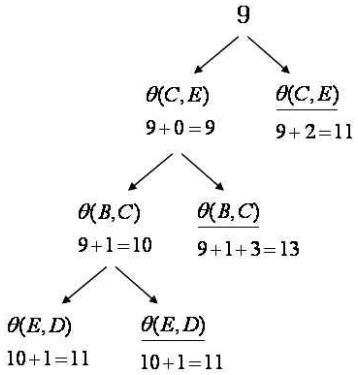
그 트리를 가지고 열린 모든 지점들 사이에서 계속 진행하면, $\theta(B, C)$는 최소 비용을 갖는다.

그래서 $\theta(B, C)$는 진행하는 것을 택한다. 차후 고려사항에서 열 B와 행 C를 제거한다. 지금까지의 결과는 다음 <표 6>과 같다.

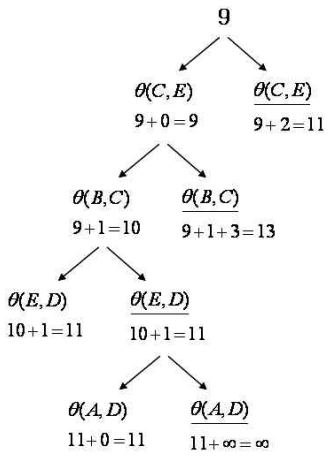
<표 6> 페널티 테이블 : 주기 2

도시	A		B		C		D	
A		∞	0	0		6	0	0
B	1	0		∞	3	0		1
C		1	1	0		3		∞
D		1		1		∞	1	0

고리를 제거하기 위해 다시 E에서 B로의 비용을 <표 7>에서 ∞로 변경한다. 0을 얻기 위해 행 A로부터 1을 빼고 <표 7>에서 페널티 결과를 계산한다. 전체 경로의 전체비용은 1 증가하게 된다.



우리가 셀(E, D)을 택하는 것은 두 가지에서 모두 페널티 11을 갖기 때문에 중요하지 않다. 셀(E, D)을 택하지 않음으로써 해법의 절차를 계속 진행한다. 셀 (E, D)에서 페널티 ∞를 위치시킴으로써 이와 같은 것이 해결될 수 있다. 결과적은 표는 <표 8>과 같다.



<표 7> 페널티 테이블 : 주기 3

도시	A		B		C	
A		∞	0	0	0	0
B	0	0	0	0		∞
C	0	0		∞	0	0

<표 8> 셀(D, E)을 선택하지 않은 후의 페널티 테이블

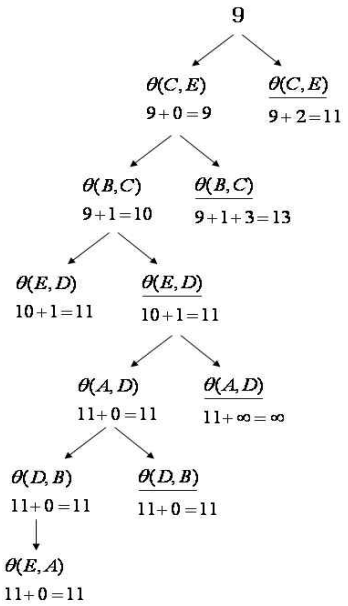
도시	A		B		C	
A		∞	0	0	∞	0
B	0	0	0	0		∞
C	0	0		∞		∞

A, D는 다음의 <표 9>에서의 진행되는 바와 같이 다음에 선택된다.

<표 9> 셀(C, E)을 선택하지 않은 후의 페널티 테이블

도시	A		B		D	
A		∞	0	0	∞	0
B	0	0	0	0		∞
D	0	0		∞		∞

다음 페널티 결과는 <표 10>와 같다.



<표 10> 셀 (A, D)을 선택한 후의 페널티 테이블

도시	A		B	
A		∞	0	0
B	0	0		∞

<표 10>로부터 두 경로 모두 D-B와 다음 E-A가 선택이 될 것이다. 관련된 트리 도표는 보는 바와 같다. 모든 노드들은 A-D-B-C-E-A인 최적 경로로 할당이 된다. (할당된 각 가지를 추적하고 연결한다. 이동 거리는 11이다. 이것은 <표 1>의 초기 비용을 사용하여 비용을 계산한 결과와 같은 값을 갖는다.)

3.3 차량 경로 문제 (Vehicle Routing Problem)

차량경로문제(Vehicle Routing Problem)는 재화 및 서비스를 수집 또는 배분하기 위하여 차량을 할당하고 그 운행경로를 결정하는 문제이다. 일반적으로 차량 경로문제의 최적화 목표는 운행비용의 최소화이며, 제약조건은 가용 차량의 대수와 적재 요구량, 수요지점의 수요량 및 차량에 대한 각종 운행 조건 등이다. 차량경로문제는 처음 Dantzing와 Ramser에 의해 제기된 이후 관련된 제약 조건을 다양화하여 광범위한 연구가 꾸준히 진행되고 있으며, 최근에 물류와 관련된 수송비 절감의 필요성에 대한 인식으로 인하여 그 중요성을 더해 감에 따라 산업공학 및 경영과학의 중요한 하나의 문제로 자리 잡고 있다.

차량경로문제는 문제의 특성에 따라 여러 형태의 모델을 구축하고, 각기 다른 발견적 해법을 적용할 수 있겠지만, 본 연구는 하나의 창고와 적재 용량이 동일한 차량, 수요량이 알려진 수요지점, 그리고 단일의 수요지점에 대해서는 오직 한대의 차량만이 일회 방문한다는 일회방문원칙의 제약 조건하에서, 모든 수요지점의 수요량을 만족시키면서 총 운행비용을 최소화하는 차량의 운행 경로를 구하는 문제를 해결하는 데 그 목적이 있다.

차량 경로문제는 수리적으로 NP-hard문제에 속하는 계산량이 아주 많은 문제이다. 또한 문제의 크기가 커질수록 계산량이 급격히 증가하는 등 규모가 큰 문제에 최적화 해법을 적용하기에는 어려움이 많으므로, 발견적 해법이 상대적으로 활발히 연구되고 있다. 차량경로문제의 가장 근간이 되는 방법이 Clake와 Wright에 의해 1964년도에 개발된 saving 방법이다. 이 때, saving이란 두 대의 차량이 각각 서로 다른 수요지점을 방문하고 돌아오는 것에 비해 한 대의 차량이 두 소요지점을 한 번에 경유하여 돌아올 때 얻게 되는 거리(또는 비용)의 절약을 의미한다.

차량경로문제의 수리 모델을 살펴보면 다음과 같다.

차량경로문제의 전통적 모형은 단일의 본점과 N개의 수요지점, 그리고 차량 용량이 U인 M대의 차량으로 구성되며, 수요지점의 수요량은 미리 알려져 있고, 차량의 운행 거리 에는 제한이 있을 수 있다.

기본적인 기호는 다음과 같이 정의할 수 있다.

N : 수요지점의 수
D_i : 수요지점의 i의 수요량 $(i=1, 2, \cdots, N)$
M : 차량의 수

U : 차량의 용량

C_{ijk} : 차량 k가 수요지점 i와 수요지점 j를 운행하는 경로 거리(또는 시간)

($i=0, 1, \cdots, N$; $k=1, 2, \cdots, M$; i, j가 0일 대 본점을 의미)

W : 차량의 최대 운행 거리(또는 시간)

수리 모형은 다음과 같다.

$$\min \sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ijk}X_{ijk} \quad (7)$$

$$\text{s.t.} \sum_{k=1}^{M}\sum_{i=0}^{N} X_{ijk} = 1 \qquad \forall j \quad (8)$$

$$\sum_{k=1}^{M}\sum_{j=0}^{N} X_{ijk} = 1 \qquad \forall j \quad (9)$$

$$\sum_{j=0}^{N} X_{ijk} - \sum_{j=0}^{N} X_{jik} = 0 \qquad \forall i, k \quad (10)$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} C_{ijk}X_{ijk} \le W \qquad \forall k \quad (11)$$

$$\sum_{i=0}^{N} D_i\left(\sum_{j=0}^{N} X_{ijk}\right) \le U \qquad \forall k \quad (12)$$

$$\sum_{j=1}^{N} X_{0jk} \le 1 \qquad \forall k \quad (13)$$

$$\sum_{i=1}^{N} X_{i0k} \le 1 \qquad \forall k \quad (14)$$

$$y_i - y_j + N\cdot\sum_{k=1}^{M} X_{ijk} \le N-1 \qquad \forall i, j(i \ne j) \quad (15)$$

$$y_i = \text{ real number} \qquad \forall i \quad (16)$$

$$X_{ijk} = \{0, 1\} \qquad \forall i, j, k \quad (17)$$

수리 모형의 식(7)은 총 운행거리(또는 시간)를 최소화하는 목적식이다. 이 때 거리(또는 시간)는 비용과 일대일로 대응하므로 식(1)은 곧 총 운행비용을 최소화하는 목적식이기도 하다. 식(8)와 식(9)은 각 수요지점은 한 대의 차량에 의해 한 번만 방문되어야 한다는 것을 의미한다. 식(10)는 차량이 수요지점을 방문한 후 반드시 떠나야 한다는 것을 의미하며, 식(11)는 경로에 포함된 지점들을 방문하는데 걸리는 총 운행거리(또는 시간)는 차량의 최대 운행거리(또는 시간)를 초과할 수 없음을 의미한다. 식(12)은 각 경로에 포함된 수요지점의 총 수요량은 차량의 적재용량을 초과할 수 없다는 것을 의미한다. 식(13)과 식(14)은 모든 수요지점을 방문하는데 필요한 차량 수는 주어진 차량 대수 M대를 넘을 수 없다는 것을 의미하며, 식(15)와 식(16)은 모든 차량들이 본점을 통과한다는 것을 의미한다. 식(17)은 0, 1 정수 조건이다.

본 문제를 풀기 위한 알고리즘은 다음과 같다.

① Clarke 와 Wright의 saving 개념을 이용한다.
② 초기해는 Clarke 와 Wright의 saving기법으로 구한다.
③ 본 해법은 이웃해 집합을 구성하고 그 과정에서 보다 개선된 해를 구하는 집합적인 접근 방식을 취한다. 따라서 집합을 구성하는 조건과 집합의 크기를 결정하는 것을 중요시 하며 집합을 구성하는 조건과 집합의 크기에 따른 해의 질적 개선 및 수행 시간에 대한 영향을 함께 분석하고자 한다.
④ 초기해의 각 차량 경로는 수요지점간의 아크(arc)를 연결한 형태로 구성된다. 이때 연결되어 있는 임의의 한 아크를 경로에서 제거하고 savings 기법을 적용하여 경로를 재구성하므로써 새로운 경로를 얻게 되며, 이것을 초기해의 이웃해 생성 절차로 한다. 이때 아크 수만큼의 이웃해들을 얻는다.
⑤ 위 절차를 다시 이웃해에 적용하여 해당 이웃해의 이웃해들을 구하며, 이러한 이웃해 생성절차를 k번 반복함으로써 이웃해의 범위를 계속 넓혀 갈 수 있다. 이때 k를 이웃해 생성 제한 횟수라 하여 k에 따라 이웃해의 범위가 정해진다. 이웃해 생성 제한 횟수 k는 이웃해 집합의 범위를 정하는데 매우 중요한 역할을 한다.
⑥ 이웃해 생성 단계에서 추가로 아크를 제거할 때는 이미 앞서 제거된 아크들에 비해 saving 값이 작은 아크를 제거한다. 또한 saving기법을 적용할 때도 제거된 아크보다 saving값이 작은 아크들만을 대상으로 하며 saving 값이 큰 아크들은 그대로 기존 해에서 가져온다.
⑦ 모든 해들은 자신의 이웃해 생성 절차를 수행하기 전에 이웃해를 생성해도 좋은지에 대한 여부를 가리고, 이웃해를 생성해도 좋은 경우에만 이웃해를 생성하게 되며 그 기준은 다음과 같다. 즉 자신이 임의의 해로부터 이웃해로서 유도되어 나왔을 때 그 해보다 값이 좋은 해만이 자신의 이웃해를 생성할 수 있다.

알고리즘 설명을 위한 기호는 다음과 같다.

K : 이웃해 생성 제한 횟수

k : 이웃해 생성 횟수, $k=1, \cdots, K$

v_k : 이웃해 생성 절차를 k번 수행하여 얻은 이웃해의 값, $k=1, \cdots, K$

A_k : 이웃해 생성 절차를 k번 수행하여 얻은 이웃해의 아크의 집합 $k=1, \cdots, K$

S_k : 이웃해 생성 절차를 k번 수행하여 얻은 이웃해의 아크 수, $k=1, \cdots, K$

i_k : A_k에 있는 아크 번호, $i_k=1, \cdots, S_k$

해법의 절차는 다음과 같다.

[단계 0] 초기화

모든 수요지점 간의 saving 기법으로 초기해를 구한다. 이때 구한 초기해와 경로를 최선해, 최선경로로 저장한다.

$k=1$

초기해, 경로, 아크 수를 각각 v_k, A_k, S_k에 저장한다.

$i_k = 1$

[단계 1] 이웃해 생성

i_k번째 아크를 경로에서 제거한다.

i_k번재 아크보다 saving 값이 작은 아크들에 대해 saving 기법을 다시 적용하여 새로운 이웃해를 구한다. 이 때 구한 해, 경로, 아크 수를 V_{k+1}, A_{k+1}, S_{k+1}에 저장한다.

[단계 2] 최선해, 최선경로 수정

[단계 1]에서 구한 해가 최선해보다 좋은 경우 최선해, 최선경로를 수정한다.

[단계 3] 이웃해를 생성할 것인지의 결정

k가 미리 내정된 이웃해 생성 제한 횟수 k를 넘지 않고, V_{k+1}이 V_k보다 좋은 경우에만 이웃해를 생성한다. 이웃해를 생성하기로 한 경우, [단계 4]로 가고, 아니면 [단계 5]로 간다.

[단계 4] k의 증가

$k = k+1$

$i_k = i_{k+1}$,

[단계 1]로 간다.

[단계 5] i_k의 증가

$i_k = i_{k+1}$

i_k가 S_k보다 크면 [단계 6]으로 아니면, [단계 1]로 간다.

[단계 6] k의 감소

이때 k가 1이상이면 [단계 5]로, k가 0일 때는 [단계 7]로 간다.

[단계 7] 종료

해의 질적 개선과 수행시간은 이웃해 생성 제한 횟수 k에 크게 영향 받으며, 이 때 이웃해 생성 제한 횟수 k의 역할은 다음과 같다. 즉, 새로운 이웃해들을 생성하는 절차를 반복하여 이웃해 집합을 계속 넓혀 가는 과정에서 이웃해를 생성하는 절차의 횟수에 제한을 둠으로써 이웃해 집합의 범위를 결정하고 이로 인하여 해의 개선정도나 수행 시간은 달라진다. 이웃해 생성 절차가 많을수록 더욱 개선된 해를 얻을 수 있으며, 이 때 수행 시간은 더욱 늘어난다.

3.4 수송문제 (Transportation Problem)

수송문제를 풀기 위한 심플렉스법은 다음 단계를 따른다.

[단계 1] 초기 기저가능해 하나를 찾는다. 수요를 만족시키는 어떠한 해도 초기 기저가능해가 될 수 있다.

[단계 2] u_i와 v_j는 쌍대변수이다. 기저변수에 대한 할인가는 0이므로 기저셀 (i, j)에 대해 $u_i + v_i - c_{ij} = 0$을 이용하여 모든 u_i와 v_j를 구한다. 이때 $u_1 = 0$으로 시작한다.

[단계 3] 모든 (i, j)에 대해 $u_i + v_i - c_{ij}$를 계산한다. $\{u_i + v_i - c_{ij}\} \leq 0$이면 현재의 해가 최적해이다. 아니면 [단계 4]로 간다.

[단계 4] 모든 비기저 셀 중에서 할인가 $u_i + v_i - c_{ij}$가 제일 큰 셀 (i^*, j^*)을 고른다. 그 셀의 변수 x_{i^*,j^*}가 새로운 기저변수가 되면 목적함수 값을 제일 많이 줄일 수 있다. 그 셀에 해당하는 변수의 현재 값 0을 θ만큼 증가시킨다. 그러면 x_{i^*,j^*}를 포함하는 열의 어떤 기저변수 x_{i^1,j^1}의 값이 반드시 θ만큼 줄어들어야 하고 따라서 x_{i^1,j^1}이 포함된 행의 어떤 기저변수 x_{i^2,j^2}의 값은 θ만큼 늘어나야 한다. 이와 같이 θ만큼 늘이고 줄이는 것을 되풀이 하여 결국 θ-사이클이라는 폐쇄경로(closed loop)를 형성하면서 출발지 (i^*, j^*)셀로 돌아와야 한다. 이 때 사이클이 형성되지 못하면 공급과 수요의 균형을 맞출 수 없다.

[단계 5] θ의 증가량을 결정한다. θ는 어떤 기저변수의 값이 최초로 0이 될 때까지 증가할 수 있다. 이 변수가 비기저변수로 빠져나간다.

[단계 6] θ-사이클에 따라서 θ를 더하고 빼줌으로 새로운 기저가능해를 구한 후 [단계 2]로 간다.

[단계 1]에서 초기 기저가능해를 찾는 방법에 대해 알아보자. 대표적인 방법으로는 북서코너법(North-West Corner Rule)을 들 수 있다. 북서코너법은 표의 북서쪽에서부터 차례로 수요를 채워나가는 손쉬운 방법이다. 북서코너법으로 할당을 해 나갈 때는 반드시 직각으로 꺾이도록 할당이 이루어져야 한다. 그래야만 $(n+m-1)$개의 셀에 할당이 가능하다. 직각을 이루면서 할당을 할 수 없는 경우에는 0을 할당하면서까지 직각으로 꺾이도록 해야 한다. 이 경우에 0을 수송하는 것은 퇴화해(degenerate solution)가 발생함을 의미한다.

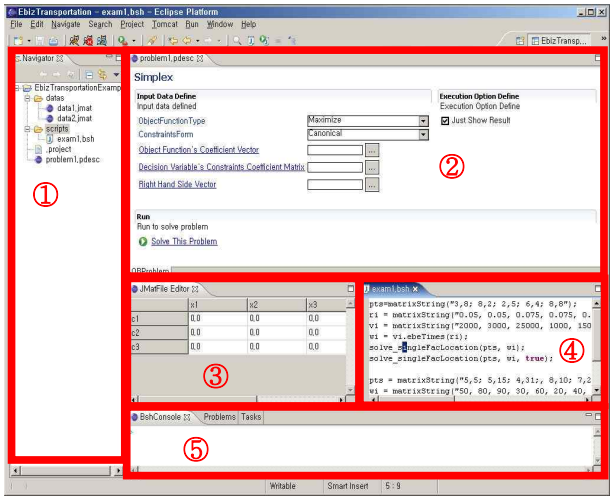
4. 소프트웨어 구현 사례

본 연구에서는 앞서 소개된 다양한 물류 의사결정 문제를 해결하기 위해 객체지향 언어인 Java를 사용하여 컴포넌트 형태의 재사용성이 높은 엔진을 개발하고, 역시 Java 기반의 통합 개발 환경 플랫폼인 Eclipse(www.eclipse.org : Eclipse 프로젝트를 주관하는 조직의 사이트, Eclipse 관련 각종소식과 활동내용들을 살펴볼 수 있다.)기반에서 운영체제 및 장비에 독립적으로 동작할 수 있는 S/W를 개발하였다.

4.1 소프트웨어 개요

4.1.1 소프트웨어 개요

E-Biz 물류 Solver Application은 아래와 같은 구성으로 이루어진다. 물류 문제관련 자료들의 묶음인 "Project"를 탐색하는 탐색기, 그리고 문제를 정의하고, 설명하는 Problem Description Editor, 그리고 입력 data를 Matrix 형태로 입력할 수 있도록 도와주는 JMatrix File Editor, 기타 간단한 Scripting을 지원하는 Script 에디터와 Script console로 이루어져 있다.



<그림 1> E-Biz 물류 소프트웨어 화면 구성

1) File Navigator

파일 네비게이터는 Project를 관리하고 탐색할 수 있도록 도와준다. 여기에서 필요한 파일을 생성 또는 삭제 할 수 있으며 파일을 편집할 수 있는 에디터를 실행시킬 수 있다.

Project는 특정 문제와 관련하여, 입력 데이터와 문제 정의파일 등을 모아놓은 폴더이다.

2) Problem Description Editor

에디터는 pdesc 파일을 사용자가 쉽게 편집할 수 있도록 직관적인 UI를 제공한다.

pdesc 파일은 확장자가 "*.pdesc"인 파일로 문제의 유형에 따라 입력 data와 실행 옵션 등의 설정을 저장하고 있는 파일이다.

3) JMatrix File Editor

이 에디터는 Matrix 혹은 Vector 형태의 Data를 쉽게 편집할 수 있도록 한다. 스프레드시트와 같은 UI로 사용자가 쉽게 작업할 수 있다.

jmat 파일은 확장자가 "*.jmat"인 파일로 Matrix 혹은 Vector 형태의 Data를 저장하고 있는 파일이다.

4) Bean Shell Script Editor

(www.beanshell.org : Bean Shell Script를 제작한 제작자의 사이트로써 매뉴얼과 구조 등에 대한 자세한 정보를 얻을 수 있다.)

Bean Shell Script는 프로그램 언어인 Java의 스크립트 언어 버전이다. 이 BSH Script 언어를 편집할 수 있는 에디터를 제공하여 사용자가 복잡한 문제를 유연하게 풀 수 있도록 지원한다.

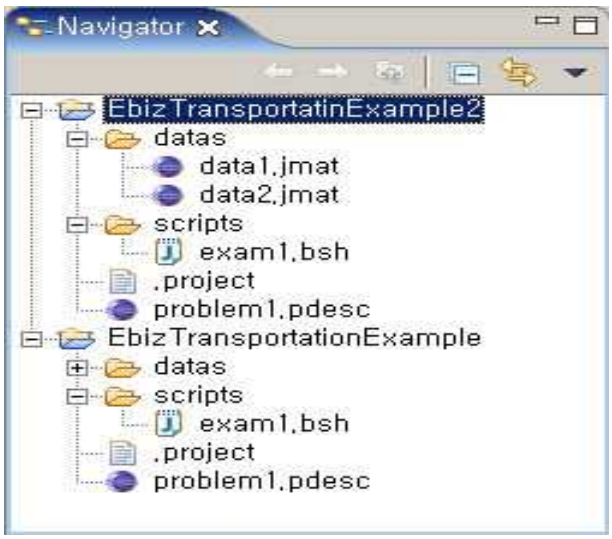
bsh 파일은 확장자가 "*.bsh"인 파일로 java의 스크립트 언어 버전인 Bean Shell Script의 코드를 저장하고 있는 파일이다.

5) Bean Shell Console

Bean Shell Script를 상호 대화모드로 실행시키는 Console이다. 이 콘솔에서는 사용자가 입력한 내용을 즉시 실행하여 그 결과를 화면에 출력한다.

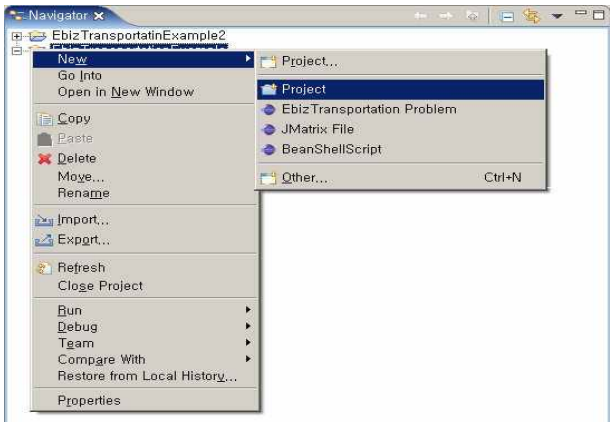
4.2 소프트웨어 구성

4.2.1 File Navigator



<그림 2> E-Biz 물류 소프트웨어 File Navigator

파일 네비게이터는 Project를 관리하고 탐색할 수 있도록 도와준다. 여기에서 필요한 파일을 생성 또는 삭제 할 수 있으며 파일을 편집할 수 있는 에디터를 실행 시킬 수 있다.



<그림 3> E-biz 물류 소프트웨어 프로젝트 생성

1) Project

프로젝트는 문제와 관련된 여러 파일들의 묶음을 표시하는 폴더이다. 위의 화면처럼 여러 형태의 파일들을 하위 폴더를 생성하여 폴더단위로 관리할 수 있다.

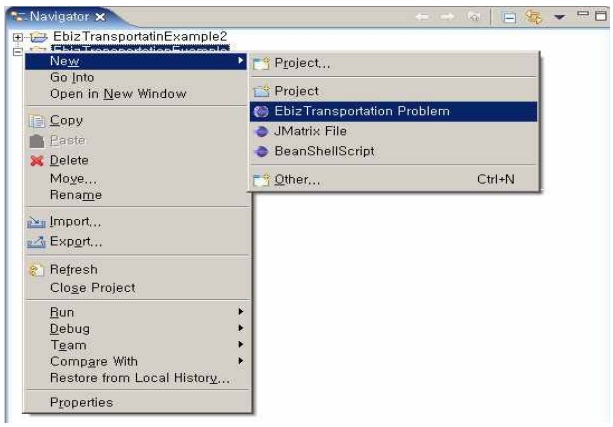
프로젝트는 아래 화면과 같이 Navigator에서 마우스 오른쪽 버튼을 클릭 하여 생성할 수 있다.

Project에는 Data와 Script 파일 외에도 여러 가지 파일을 담을 수 있다. 이중에서 E-Biz 물류 Solver와 관련한 파일들은 아래와 같다.

① pdesc 파일

확장자가 "*.pdesc"인 파일로 문제의 유형에 따라 입력 data와 실행 옵션 등의 설정을 저장하고 있는 파일이다. 이 파일을 더블클릭하면 Problem Description Editor가 나타난다.

pdesc 파일은 파일을 생성할 Project 혹은 Project의 하위 폴더를 선택하고, 마우스 오른쪽 버튼을 누른 후 아래 화면과 같이 "Ebiz Transportation Problem"을 선택하여 생성할 수 있다.

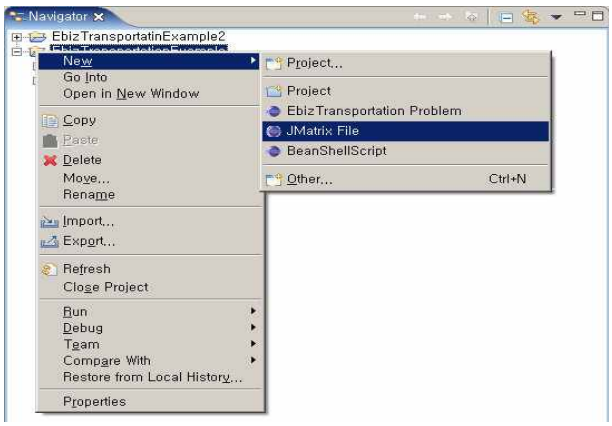


<그림 4> E-Biz 물류 소프트웨어 Problem Description 파일 생성

② jmat 파일

확장자가 "*.jmat"인 파일로 Matrix 혹은 Vector 형태의 Data를 저장하고 있는 파일이다. 이 파일을 더블클릭하면 JMatrix File Editor가 나타난다.

jmat 파일은 파일을 생성할 Project 혹은 Project의 하위 폴더를 선택하고, 마우스 오른쪽 버튼을 누른 후 아래 화면과 같이 "JMatrix File"을 선택하여 생성할 수 있다.

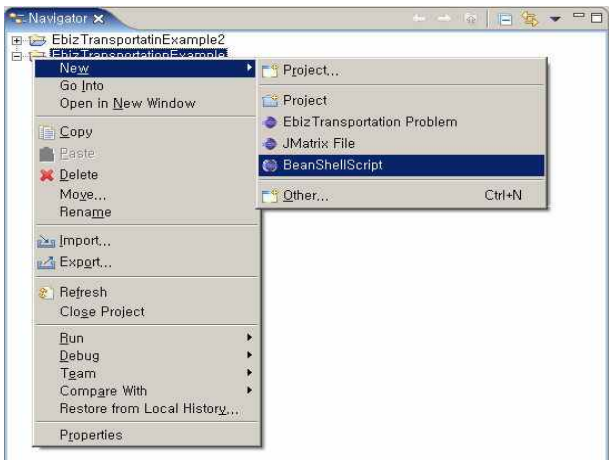


<그림 5> E-Biz 물류 소프트웨어 jmat 파일 생성

③ bsh 파일

확장자가 "*.bsh"인 파일로 java의 스크립트 언어 버전인 Bean Shell Script의 코드를 저장하고 있는 파일이다. 이 파일을 더블 클릭하면 Bean Shell Script Editor가 나타난다.

bsh 파일은 파일을 생성할 Project 혹은 Project의 하위 폴더를 선택하고, 마우스 오른쪽 버튼을 누른 후 아래 화면과 같이 "Bean Shell Script"를 선택하여 생성할 수 있다.



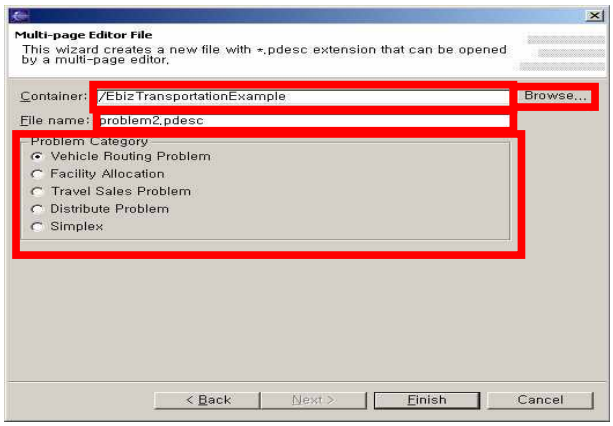
<그림 6> E-Biz 물류 소프트웨어 bsh 파일 생성

4.2.2 Problem Description Editor

Problem Description Editor는 E-biz 물류 문제의 유형과 입력 Data, 그리고 실행 옵션을 저장하고 있는 pdesc file을 편집하고 실행시킬 수 있는 편집기이다.

1) pdesc file의 생성

Navigator에서 파일을 생성할 Project 혹은 Project의 하위 폴더를 선택하고, 마우스 오른쪽 버튼을 누른 후 "New>Ebiz Transportation Problem"를 선택하면 아래와 같은 화면이 나타난다.



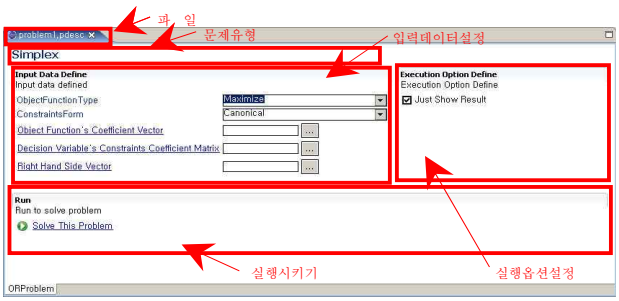
<그림 7> E-Biz 물류 소프트웨어 pdesc 파일 생성 option 설정

<표 11> E-Biz 물류 소프트웨어 pdesc 파일 생성 option 항목 설명

항 목	내 용
Container	파일을 생성할 프로젝트 혹은 프로젝트의 하위 폴더를 입력한다. 파일 생성대화상자 선택 시 선택되어 있던 프로젝트 혹은 프로젝트의 하위폴더가 기본적으로 입력이 된다. 만약 이를 바꾸고자 할 때는 화면 왼쪽의 "Browse" 버튼을 클릭하여 폴더 선택 대화상자를 실행시킨 후 입력 할 수 있다.
Filename	파일의 이름을 지정한다.
Problem category	문제의 유형을 선택한다. 현재 E-Biz 물류 solver는 5가지의 문제 유형을 지원한다. 이 문제 유형중 하나를 선택한다. · Vehicle Routing Problem : 차량 물류 문제를 푼다. · Facility Allocation : 물류 기지의 위치 선정 문제를 푼다. · Travel Sales Problem · Distribute Problem · Simplex : LP유형의 문제를 Simplex Method로 푼다. "Finish"버튼을 누르면 파일이 생성되면서 "Problem Description Editor"가 화면에 나타나게 된다.

2) 화면구성

"Problem Description Editor"는 문제의 유형을 표시하고, 문제 유형에 맞는 Data를 입력받은 후, 문제 유형에 따른 실행 Option을 선택하고, 이를 실행시킬 수 있다.



<그림 8> E-Biz 물류 소프트웨어 Problem Description Editor화면구성

<표 12> E-Biz 물류 소프트웨어 Problem Description Editor 항목 설명

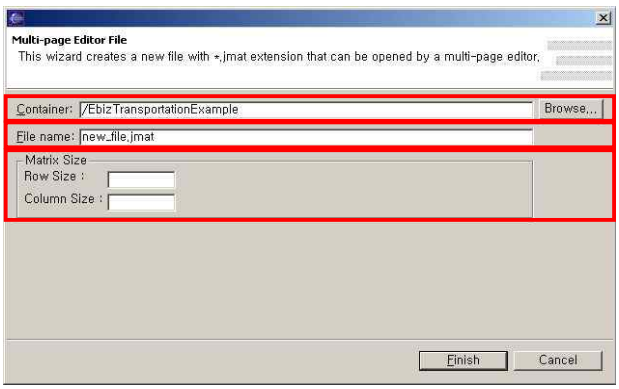
항목	내 용
제목	파일명이 표시된다.
문제 유형	문제 유형이 표시된다.
입력Data설정	문제 유형에 다른 필요 입력 Data를 설정한다. · 입력데이터 밑줄을 클릭하면 선택한 파일을 연다. 만약 선택한 파일이 없다면 새로운 파일을 생성을 도와주는 마법사 대화상자가 나타난다. · 입력데이터 항목 왼쪽의 버튼을 클릭하면 파일 선택 대화상자가 나타난다. 이 대화상자를 통해서 적절한 파일을 선택할 수 있다.
실행옵션설정	문제 유형에 따른 실행 Option을 설정한다.
실행	"Solve This Problem" 링크를 누르면 문제 풀기를 시작한다.

4.2.3 JMatrix File Editor

JMatrix File Editor는 Matrix 형태의 Data를 입력하고 편집할 수 있도록 스프레드 시트의 UI를 제공하는 편집기이다.

1) jmat file 생성

Navigator에서 파일을 생성할 Project 혹은 Project의 하위 폴더를 선택하고, 마우스 오른쪽 버튼을 누른 후 "New> JMatrix File"를 선택하면 아래와 같은 화면이 나타난다.



<그림 9> E-Biz 물류 소프트웨어 jmat 파일 생성 option

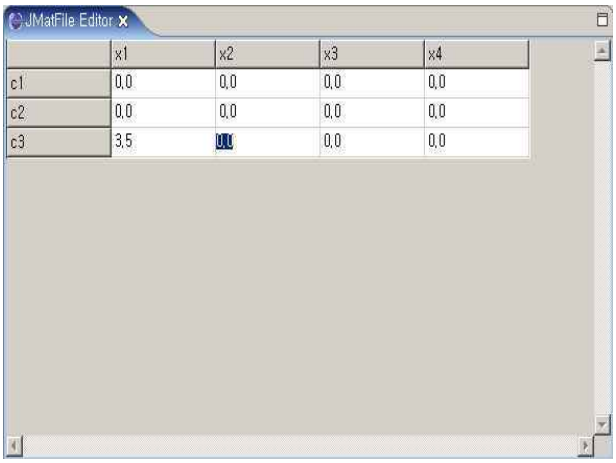
<표 13> E-Biz 물류 소프트웨어 jmat 파일 생성 option 항목 설명

항 목	내 용
Container	파일을 생성할 프로젝트 혹은 프로젝트의 하위 폴더를 입력한다. 파일 생성대화상자 선택 시 선택되어 있던 프로젝트 혹은 프로젝트의 하위폴더가 기본적으로 입력이 된다. 만약 이를 바꾸고자 할 때는 화면 왼쪽의 "Browse" 버튼을 클릭 하여 폴더 선택 대화상자를 실행시킨 후 입력 할 수 있다.
File name	생성할 파일의 이름을 입력한다.
MatrixSize	생성할 Matrix Data의 Row와 Column의 크기를 입력한다.

내용을 입력한 후 "Finish"버튼을 누르면 파일이 생성되고, JMatrix File Editor가 실행된다.

2) 편집하기

아래는 새로 만들어진 jmat 파일을 에디터로 실행시킨 모습이다. 이 화면에서 입력하고자 하는 셀을 선택하여 더블클릭한 후 Data를 입력하면 된다.



<그림 10> E-Biz 물류 소프트웨어 JMatrix Editor

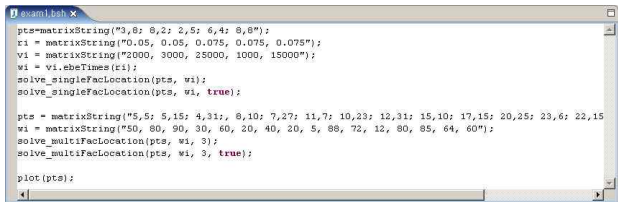
4.2.4 Bean Shell Script Editor

Bean Shell은 Java로 구현된 객체 스크립팅 언어의 특성을 가지는 Java 소스 인터프리터이다. Bean Shell Script 에디터의 주요기능으로는 문법 키워드를 컬러로 표시하고, 간단한 코드 완성 기능을 지원하며, 작성한 코드를 실행시킨다.

1) Bean Shell Script Example

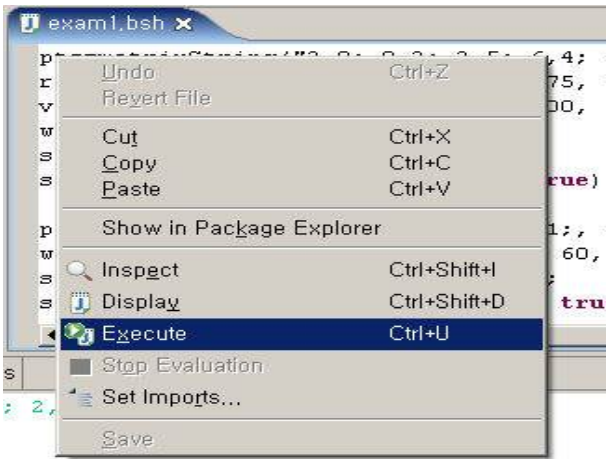
Bean Shell Script 파일을 생성한 후 아래의 내용을 에디터에 입력한다.

```
pts=matrixString("3,8; 8,2; 2,5; 6,4; 8,8");
ri = matrixString("0.05, 0.05, 0.075, 0.075,
0.075");
vi = matrixString("2000, 3000, 25000, 1000,
15000");
wi = vi.ebeTimes(ri);
solve_singleFacLocation(pts, wi);
solve_singleFacLocation(pts, wi, true);
pts = matrixString("5,5; 5,15; 4,31;, 8,10;
7,27; 11,7; 10,23; 12,31; 15,10; 17,15; 20,25;
23,6; 22,15; 25,18; 30,15; 31,30");
wi = matrixString("50, 80, 90, 30, 60, 20, 40,
20, 5, 88, 72, 12, 80, 85, 64, 60");
solve_multiFacLocation(pts, wi, 3);
solve_multiFacLocation(pts, wi, 3, true);
```



<그림 11> E-Biz 물류 소프트웨어 Bean Shell Script Editor

에디터에 입력 후 에디터에서 마우스 오른쪽 버튼을 눌러 아래 화면과 같이 "Execute"를 선택하면 코드가 실행이 된다.

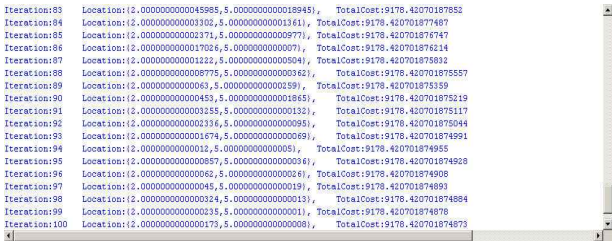


<그림 12> E-Biz 물류 소프트웨어 Bean Shell Script 파일 실행시키기(1)

또는 툴바 에서 Script 실행버튼을 눌러도 실행이 된다.

<그림 13> E-Biz 물류 소프트웨어 Bean Shell Script 파일 실행시키기(2)

실행을 시키면 아래와 같은 내용이 화면 하단의 콘솔창에 출력이 된다.



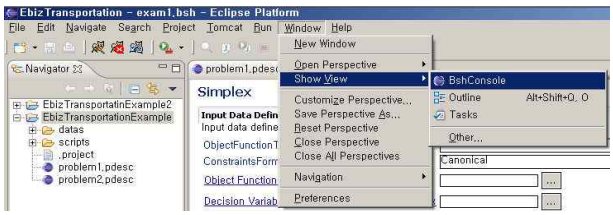
<그림 14> E-Biz 물류 소프트웨어 Bean Shell Script 실행결과 화면

4.2.5 Bean Shell Console

Bean Shell Script를 상호 대화모드로 실행시키는 Console이다. 이 콘솔에서는 사용자가 입력한 내용을 즉시 실행하여 그 결과를 화면에 출력한다.

1) Bean Shell Console 실행시키기

Bean Shell Console은 아래와 같이 화면 메뉴 "Window>Show View>Bsh Console"을 선택하면 윈도우 하단에 나타나게 된다.

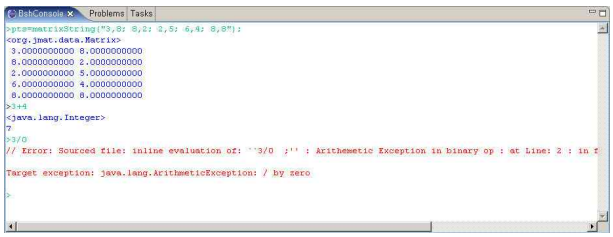


<그림 15> E-Biz 물류 소프트웨어 Bean Shell Console 실행시키기

2) Bean Shell Console 사용 예

Bean Shell Console이 실행되면 화면에 ">"표시가 나타난다.

이 상태에서 Bean Shell Script Code를 입력하고 "<enter>" 키를 입력하면 코드를 실행하여 그 결과를 바로 화면에 출력하게 된다.



<그림 16> E-Biz 물류 소프트웨어 Bean Shell Console 실행 결과화면

간단한 예제를 살펴보기 위해 Matrix 에디터를 생성한다. 화면에 "pts=matrixString("3,8; 8,2; 2,5; 6,4; 8,8");" 라고 입력 후 "<enter>"키를 입력하면 Matrix Data의 생성결과를 화면에 표시하게 된다. 또한 간단한 계산 결과를 살펴보면, 화면에 "3+4"라고 입력 후 "<enter>"키를 입력하면 계산 결과인 "7"을 화면에 표시하는 것을 알 수 있다. 에러표시를 정확히 표현하고 있는지를 보기 위해 화면에 "3/0"라고 입력 후 "<enter>"키를 입력하면 화면에 숫자 "0"으로는 "/"연산을 할 수 없다는 내용의 에러 메시지가 출력된다.

4.3 소프트웨어 엔진

4.3.1 E-biz 물류 Solver 엔진

E-biz 물류 Solver는 문제 해결을 위해 자체적인 엔진을 사용한다. 이 엔진은 확장성 있는 구조로 작성되어있기 때문에 문제가 변형되고 추가되는 상황에서 유연성을 발휘 할 수 있다.

E-biz 물류 Solver의 엔진은 객체지향언어인 Java로 제작 되었다. 따라서 객체 지향의 장점을 수용하고, 또한 Java 언어의 특성상 플랫폼에 종속되지 않고 다양한 플랫폼에서 동작한다. 또한 Component 형태로 제작되어 다른 application에서도 쉽게 재사용할 수 있다.

<표 13> E-biz 물류 Solver 엔진 적용 package

package	기 능
yonsei.sol.core	E-biz 물류 Solver 엔진의 Framework을 형성하는 주요 Interface에 대해서 정의한다.
yonsei.sol.core.lp	Linear Programing의 기본적인 Algorithm 클래스를 제공한다.
yonsei.so.core.util	package들에서 공통적으로 사용하는 기능을 class화하여 구현의 용이성을 제공한다.
yonsei.sol.core.logistics	물류와 관련한 물류 문제들과 관련된 공통된 Interface를 정의한다.
yonsei.sol.core.logistics.allocation	Facility Location등 위치 설정 문제를 풀 수 있는 Library를 제공한다.
yonsei.sol.core.logistics.tsp	Travel Sales Problem에 속하는 다양한 형태의 문제를 풀 수 있는 Library를 제공한다.
yonsei.sol.core.logistics.vrp	차량경로 문제에 속하는 다양한 형태의 문제를 풀 수 있는 Library를 제공한다.
yonsei.sol.core.logistics.distribute	transportation의 기본적인 분배 문제를 푼다.

5. Test bed

본 연구에서 개발된 소프트웨어의 검증을 위한 Test bed는 차량 경로문제와 관련한 간단한 예제의 해결을 통해 제시하고자 한다. 이와 관련한 기준 문제는 <표 15>와 같고, 기준 문제 각각에 대해서는 이웃해 생성 제한 횟수(k)에 따라 구분하여 결과치를 제시하였다.

각 지점마다 수요량이 발생하는 것으로 가정하여 실험을 수행하였으며, 차량의 용량은 처리가능한 수요 물량 크기로 하였으며, 차량 운행시간의 제한을 둔 6~10문제에 대해서는 추가적인 하역시간을 고려하여 문제의 복잡성을 높였다.

<표 14> 기준 문제

문제번호	지점 수	수요량 범위	차량용량	운행제한시간	하역시간
1	50	3~41	160	-	-
2	75	1~37	140	-	-
3	100	1~36	200	-	-
4	120	2~35	200	-	-
5	100	10~50	200	-	-
6	50	3~41	160	200	10
7	75	1~37	140	160	10
8	100	1~36	200	230	10
9	120	2~35	200	720	50
10	100	10~50	200	1040	50

우선 차량의 운행 시간에 제한이 없는 경우에 대한 결과는 다음과 같다. 기준 문제 중 문제 1~5는 차량의 운행시간에 제한이 없는 경우에 대한 문제로써, <표 16>는 이웃해 생성 제한 횟수인 k값의 변경에 따른 운행시간 결과를 보여주고 있으며, <표 17>은 해를 찾는데 소요되는 수행시간을 나타낸다. <표 16>과 <표 17>를 살펴보면 k값이 커질수록 더욱 개선된 해를 구할 수 있음을 알 수 있다. 그러나 k가 일정 값 이상으로 커지면 더 이상 해가 개선되지 않는다는 것을 알 수 있다. <표 16>과 <표 17>에 공백으로 남아 있는 부분은 해가 더 이상 개선되지 않았음을 의미한다.

<표 16> 차량의 운행 시간에 제한이 없는 문제에 대한 해

k	문제 1	문제 2	문제 3	문제 4	문제 5
1	548.49	867.22	863.94	1051.47	824.68
2	546.43	863.84	853.23	1049.82	823.25
3	944.97	859.31	851.48	1049.55	822.32
4	541.06	852.28	850.31	-	820.89
5	538.99	844.55	849.62	-	819.97
6	-	840.57	848.47	-	819.56
7	-	839.76	848.06	-	-

<표 17> 차량의 운행 시간에 제한이 없는 문제에 대한 해법의 수행시간

k	문제 1	문제 2	문제 3	문제 4	문제 5
1	2.857	19.230	54.170	75.329	45.714
2	4.725	38.956	99.615	115.659	60.484
3	8.351	99.340	244.175	244.505	96.318
4	13.516	234.065	660.989	-	150.164
5	17.250	465.549	1603.901	-	200.604
6	-	770.549	3145.164	-	232.032
7	-	1083.186	5285.549	-	-

다음으로 차량의 운행 시간에 제한이 있는 경우인 문제 6~10은 앞선 문제 1~5와 본점이나 수요지점의 위치, 수요지점의 수요량, 차량의 용량 등의 제약은 동일하나 차량의 운행 시간에 제한이 있다는 점이 다르다. 이러한 내용을 고려하였을 때, <표 18>는 k값의 변경에 따른 결과치이며 <표 19>는 해법의 수행시간을 나타낸다.

<표 18> 차량의 운행 시간에 제한이 있는 문제에 대한 해

k	문제 6	문제 7	문제 8	문제 9	문제 10
1	579.98	951.58	948.85	1568.32	868.52
2	557.64	942.04	922.33	1566.70	866.95
3	555.43	936.58	919.65	1564.72	-
4	-	936.14	917.05	1558.73	-
5	-	-	916.46	1557.29	-
6	-	-	-	-	-
7	-	-	-	-	-

<표 19> 차량의 운행 시간에 제한이 있는 문제에 대한 해법의 수행시간

k	문제 6	문제 7	문제 8	문제 9	문제 10
1	3.186	14.615	43.186	75.714	44.318
2	6.040	23.846	68.846	130.109	50.602
3	10.604	45.659	149.340	360.274	-
4	-	76.538	332.802	812.637	-
5	-	-	639.870	2368.131	-
6	-	-	-	-	-
7	-	-	-	-	-

6. 결 론

본 연구에서 개발된 소프트웨어는 물류 및 공급사슬을 전공하는 학생이나 관련된 업종에 종사하는 사용자들을 대상으로 물류 의사결정에 대한 다양한 문제를 정의하고 알고리즘을 통해 이를 해결해 나가는 데 도움을 주고자 하였다. 본 소프트웨어를 통해 사용자들은 물류와 관련된 위치 할당 문제, 외판원 문제(Travelling Salesman Problem), 차량 경로 문제(Vehicle Routing Problem), 수송 문제 (Transportation Problem) 등을 접해 볼 수 있으며, 원하는 문제를 정의하고 그 문제의 해를 찾는 데 사용되어질 수 있을 것으로 사료된다.
본 연구에서 제시된 문제는 가장 일반적인 문제 형태를 사용하였으며, 문제를 풀기 위한 알고리즘도 이에 맞는 것들로 선택하였다. 즉, 학생들이 교재를 통해 쉽게 접해 볼 수 있는 알고리즘을 사용하여 문제가 풀어지는 과정을 이해할 수 있도록 하였다.

본 연구를 통해 개발된 소프트웨어는 Java 환경 하에서 사용자들로 하여금 사용하기에 친숙하도록 Interface를 설계하였으며, 간단한 입력값만 설정하면

간단한 예제 문제에 대한 해를 쉽게 구할 수 있도록 구현되어 있다.

또한, 중급 사용자들을 위해서 실제 다른 형태의 알고리즘을 개발하여 사용할 수 있도록 하였으며, 이를 통해 일반적인 문제가 아닌, 많은 제약과 조건을 가진 현실적인 문제를 접근할 수 있도록 설계되었다.

본 소프트웨어를 통해 학생들이 E-biz환경에서 다양한 문제를 해결해 봄으로써, 실제 문제를 정의하고 실습하는 데 유용하게 이용될 수 있다는 데 그 의의가 있겠다. 뿐만 아니라, 실제 기업의 데이터를 활용하여 물류 의사결정 문제를 접근해 봄으로써의 모형의 현장 적용성을 검증하고자 한다.

7. 참 고 문 헌

[1] Brandão, J.C.S., Mercer, A.(1998), "The multi-trip vehicle routing problem," Journal of the Operational Research Society, 49, 799-805.
[2] Clarke, G., Wright, J.W.(1964), "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operations Research, 12, 568-581.
[3] Gillett, B., Miller, L.(1974), "A heuristic for the vehicle dispatching problem," Operations Research, 22, 340-349.
[4] Kalantari, B., Hill, A.V., Arora, S.R.(1985), "An algorithm for the traveling salesman problem with pickup and delivery customers". European Journal of Operational Research, 22, 377-386.
[5] Laporte, G., Nobert, Y., Taillefer, S.(1988), "Solving a family of multi-depot vehicle routing and location-routing problems," Transportation Science, 22, 161-172.
[6] Lau, H.C., Sim, M., Teo, K.M.(2003), "Vehicle routing problem with time windows and a limited number of vehicles," European Journal of Operational Research, 148, 559-569.
[7] Sule, D.R.(1998), Manufacturing Facilities Location, Planning, and Design Second Edition, PWS Publishing Company.
[8] Taillard, E.D.(1999), "A heuristic column generation method for the heterogeneous Fleet vrp," RAIRO Recherche Opérationnelle, 33(1), 1-14.
[9] www.beanshell.org
[10] www.eclipse.org

저 자 소 개

한 재 현

고려대학교 동양사학과 학사
University of Chicago 금융수학 석사 George Washington 대학 재무관리 박사, 현재 광운대학교 경영대학 경영학부 교수
관심분야: 기업재무, 재무적 공급사슬
주소: 서울시 노원구 광운로 20 광운대학교 경영대학 경영학부

김 장 엽

한국해양대학교 물류시스템공학과 학사, 연세대학교 정보산업공학과 석사, 현재 연세대학교 정보산업공학과 박사과정
관심분야: SCM, 물류시뮬레이션
주소: 서울시 서대문구 신촌동 연세대학교 정보산업공학과

정 석 재

한국해양대학교 물류시스템공학과 학사, 연세대학교에서 산업공학 석사 취득, 현재 동 대학원에서 산업공학 박사과정 중이며, 공급사슬 관리와 스케줄링이 주 전공이며 관심분야는 SCM, 친환경 공급사슬, 가치기반 공급사슬 등
주소: 서울시 노원구 광운로 20 광운대학교 경영대학 경영학부

김 지 현

고려대학교 산업공학과 학사
고려대학교 산업공학과 석사
University of Michigan 산업공학과 박사
현재: 광운대학교 경영대학 경영학부 교수
관심분야 : 생산관리, 품질관리
주소: 서울시 노원구 광운로 20 광운대학교 경영대학 경영학부