대한산업공학회/한국경영과학회 2002 춘계공동학술대회
한국과학기술원(KAIST) 2002년 5월 3일~4일

제약 프로그래밍과 메타휴리스틱을 활용한 차량 일정계획 시스템 개발에 관한 연구

김용환 · 장용성 · 유환주
(주) KSTEC 정보기술연구소

A Study on Developing Vehicle Scheduling System using Constraint Programming and Metaheuristics

Yong-Hwan Kim · Yong-Sung Jang · Hwan-Ju Ryu

Abstract

Constraint Programming is an appealing technology for modeling and solving various real-world problems, and metaheuristic is the most successful technique available for solving large real-world vehicle routing problems. Constraint Programming and metaheuristic are complementary to each other. This paper describes how iterative improvement techniques can be used in a Constraint Programming framework(ILOG Solver and ILOG Dispatcher) for Vehicle Routing Problem. As local search gets trapped in local solution, the improvement techniques are used in conjunction with metaheuristic method.

1. 서론

1.1 연구목적 및 배경

물류란 하나의 공급지에서 다양한 수요지로 제품과 서비스를 공급하는 것이다(Eilon *et al.*, 1971). 물류센터는 크게 중앙물류센터(Central Distribution Center), 지역물류센터(Regional Distribution Center), 최종배송센터(Final Delivery Center)로 나뉘어지며, 중앙물류센터의 역할은 공장에서 생산된 제품을 보관하고, 지역물류센터로 수송하는데 있고, 지역물류센터는 최종배송센터나 대리점으로 수·배송 하게 되며, 마지막으로 최종배송센터에서 고객들에게 물건이 배송되게 된다. 일반적으로 수송이란 중앙물류센터에서 지역물류센터로 이동시키는 것을 말하며, 배송은 최종배송센터에서 고객에게 물건을 배달하는 것을 말한다(Tilanus, 1997).

물류 시스템에서 중요한 척도 중 하나가 효율적인 차량경로를 발견하는 것이다(Tan, 2001). 차량경로문제(Vehicle Routing Problem)는 주어진 다양한 제약을 만족시키면서 차량의 최적화된 경로를 찾는 것이다. 차량경로문제는 Danzig와 Ramser(1959)의 연구에 의해 NP-hard로 알려져 있으며, 시간제약이 있는 차량경로문제(VRPTW: Vehicle Routing Problem with Time Windows)로 확장할 수 있다.

차량경로문제에 대한 연구는 Clarke 와 Wright(1964)에 의한 연구를 비롯하여, Balinski 와 Quandt(1964), Wren(1971), Fisher 와 Jaikumar(1981) 등 다양한 배차제약조건 등을 고려한 많은 연구가 되어 왔다(Laporte *et al.*, 2000). 이러한 차량운행경로에 대한 연구를 바탕으로 국내외적으로 많은 배차지원시스템이 개발되어 왔다. 국내에서는 박순달 외(1987)는 관광버스배차계획시스템을 개발하였고, 이영해 외(1994), 변의석(2000), 송성헌 외(2001), 등이 차량배차시스템을 연구 개발하였다. 초기 차량배차시스템은 지리적 정보와 연계되어있지 않고, 특정한 배차 상황만 적용할 수 있도록 개발되어 왔으며, 지리정보시스템(GIS)과 연계가 되더라도 동적인 상황(Dynamic Scheduling)과 재일정계획(Rescheduling)등은 고려되지 못하고 있다. 국외적으로는 Golden *et al.*(1987)의 음료산업, List *et al.*(1991)의 폐기물 수거 문제, Bowerman(1995)의 통학버스 경로 문제, Spasovic *et al.*(2001)의 New Jersey 학교버스 경로문제 등이 있다. 외국의 사례도 국내 사례와 마찬가지로 범용적이지 못하다는 단점과 함께 최적해가 아닌 가능해(Feasible Solution) 수준에서 머무르고 있다.

본 연구에서는 제약프로그래밍의 효과적인 모델링과 제약 전파에 의한 효율적인 탐색트리구성, 객체지향언어로 구현되어 유연성과 이식성이 뛰어나고 발견적 기법의 적용 및 구현이 용이한 ILOG Solver와 ILOG Dispatcher를 활용하여, GIS 정보와 연계되고 차량경로의 동적일정계획, 재일정계획, 지역탐색(Local Search)기법과 메타휴리스틱을 적용하여 고품질의 해(Near Optimal Solution)를 구하도록 시도하였다. 실제 사례로 국내 대형 종합물류회사인 T사를 대상으로 프로토타입 버전(Prototype Version)으로 개발된 차량일정계획시스템에 대해 설명하고자 한다.

1.2 제약프로그래밍

제약프로그래밍은 인공지능(Artificial Intelligent)기법 중 하나로 제약조건을 만족하는 가능해를 찾아내는 방법을 가리킨다. 이 기법은 최적화 알고리즘인 분지한계(Branch & Bound)법과 마찬가지로 모든 탐색 영역을 탐색하여 해를 찾아낸다. 따라서 최적해를 찾아내는데 걸리는 검색 시간이 오래 걸린다는 단점이 있다. 이러한 단점을 극복하기 위하여 제약프로그래밍에서는 Back-tracking, Back-checking, Forward-checking 기법들이 사용된다.

제약프로그래밍에 대한 연구로는 1980년대 초 Mark F. 에 의해 시작되었으며, 제약 프로그래밍 언어로는 CHIP, clp(FD), ECLiPSe, ILOG Solver, PROLOG III 등을 들 수 있다. 이러한 제약프로그래밍의 기본적인 특징은 첫째, 모델 부분과 알고리즘 부분의 분리, 둘째, 변수에 할당 가능한 값의 집합(Domain)의 범위가 변수간의 관계(Constraint)에 의해 제약전파(Constraint Propagation)가 일어남으로써 집합감소(Domain Reduction)가 일어난다는 것이다. 여기서 제약전파란 가능해의 영역을 자동적으로 줄여 주는 것을 말한다.

본 연구에서는 제약프로그래밍 언어 중 가장 우수한 ILOG Solver와 ILOG Dispatcher를 이용하였다. ILOG Solver는 제약프로그래밍을 객체지향(Objective Oriented Programming)환경으로 구현한 C++ 프로그래밍 라이브러리이고, ILOG Dispatcher는 차량경로문제에 알맞게 ILOG Solver를 좀더 객체 지향적으로 표현해 놓은 라이브러리이다.

2. 차량일정계획시스템모형

2.1 차량일정계획시스템의 개발방향

본 연구에서 T사의 차량일정계획시스템은 하나의 본점(Single Depot), 차량의 경로가 1회 이상(Multi-routing), 시간제약(Time Windows)이 있는 고객에게 제품과 서비스를 제공하는 문제이다.

T사의 하루 주문 수는 약 1000여건이었으며, 한 고객이 여러 개의 물품을 주문하는 경우와 대리점에서 주문하는 경우도 있었다. 만약 대리점과 같이 주문의 양이 많아 한 대의 차량이 배송하지 못할 경우 고객의 지점에 같은 좌표를 새로이 생성시켜 다른 차량이 배송 하도록 하였다. 고객간의 거리는 두 고객과의 직선거리(Euclidean Distance)를 사용하였으며, 고객의 좌표는 고객의 인적 사항과 주문 내용을 입력받아 지리정보시스템(GIS) 데이터베이스(DB)와 연계시켜 지리적 위치(위, 경도)를 파악하였다. 제품의 종류(Item)는 약1500가지 정도였으며, 비슷한 종류끼리 약 200개의 그룹으로 묶었다. 모든 제품은 설치와 배달 2가지 종류로 나누어졌으며, 제품의 설치시간은 20분 - 150분까지 세분화되어 있었다.

차량에 대한 용적은 제품의 부피를 기준으로 하였으며, 차량의 종류는 2가지로 1톤 차량과 2.5톤 차량으로 나뉘어지며 총 56대가 있었다. 또한 차량별로 제품의 상차 시간은 각각 차이가 있다. 차량의 운행거리는 미터 단위로 표현 했으며 직선거리/차량의 속도를 단위비용으로 하였다.

ILOG Dispatcher를 이용한 최적화 엔진의 해법 구조는 <그림 1> 과 같다. 본 연구에서는 Tan *et al.*(2001)의 모형을 응용하였고, 모델링 하기 어려운 현실적인 제약으로 통계적 교통상황을 반영한 차량의 속도, 운전 기사들의 점심 시간 및 휴식 시간, 동적일정계획(Dynamic Scheduling), 재일정계획(Rescheduling) 등의 제약을 ILOG Dispatcher 프로그램에 추가하였다.

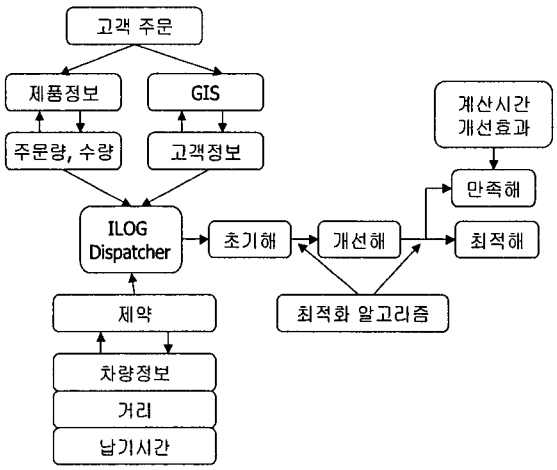


그림 1. ILOG Dispatcher 해법 구조

2.2 기호의정의및모형

문제를 정형화하기 위해 다음과 같은 기호를 사용한다.

K: 전체 차량의 수
N: 전체 고객의 수
d_{ij}: 고객 i와 j 사이의 직선(Euclidean) 거리
c_{ij}: 고객 i와 j 사이에서 발생되는 비용
t_{ij}: 고객 i와 j 사이의 운행 시각
m_i: 고객 i의 수요
q_k: 차량 k의 용적
e_i: 고객 i가 요구하는 가장 이른 시각
l_i: 고객 i가 요구하는 가장 늦은 시각
f_i: 고객 i에서의 서비스 시각
t_i: 고객 i에 도착 시각
w_i: 고객 i에 대기시각
r_k: 차량 k가 운행할 수 있는 시작시각
r'_k: 차량 k의 최대운행시각
x_{ijk}: $\in \{0, 1\}$ 결정변수, 차량 k가 고객 i에서 j로 이동할 때 1, otherwise 0. $i \neq j;\ i, j \in \{0,1,2,\ldots,N\}$

이를 수리 모형으로 표현하면 식(1) - 식(10)과 같다. 본 차량일정계획시스템 모형의 목적식은 식(1)로서, 차량의 총 운행 거리를 최소화하거나 최소 차량의 대수를 구하는 것이다. 이때 차량의 운행 거리는 비용과 일대일로 대응할 수 있으므로 식(1)은 차량의 총 운행 비용을 최소화하는 목적식이 될 수 있다.

유효한 차량의 경로를 결정하기 위한 제약식으로는 식(2) - 식(10)이 사용된다. 식(2)는 본점(Depot)에서 출발

할 수 있는 차량의 수이며 식(3)은 각 차량은 본점에서 시작해서 본점으로 끝난다는 제약이다. 식(4)와 식(5)는 모든 고객은 한 대의 차량에 의해 단 한번 방문한다는 일회방문의 제약을 의미한다. 식(6)는 차량의 용적 제한이다. 식(7)은 본점에서 차량이 출발할 수 있는 시간(출근시간)과 본점으로 되돌아 와야 하는 시간(퇴근시간)을 의미한다. (8) - (10)은 고객의 요구하는 시간(Time Windows)에 대한 제약이다.

$$Minimize \sum_{i=0}^{N} \sum_{j=0, i\neq j}^{N} \sum_{k=1}^{K} c_{ij}\, x_{ijk} \quad (1)$$

subject to :

$$\sum_{k=1}^{K} \sum_{j=1}^{N} x_{ijk} = K \quad \text{for} \quad i=0 \quad (2)$$

$$\sum_{j=1}^{N} x_{ijk} = \sum_{j=1}^{N} x_{ijk} \le 1 \quad \text{for} \quad i=0$$

$$\text{and} \quad k \in \{1,\ldots,K\} \quad (3)$$

$$\sum_{k=0}^{K} \sum_{j=0, j\neq i}^{N} x_{ijk} = 1 \quad \text{for } i \in \{1,\ldots,N\} \quad (4)$$

$$\sum_{k=0}^{K} \sum_{i=0, i\neq j}^{N} x_{ijk} = 1 \quad \text{for } j \in \{1,\ldots,N\} \quad (5)$$

$$\sum_{i=1}^{N} m_i \sum_{j=0, j\neq i}^{N} x_{ijk} \le q_k \quad \text{for } k \in \{1,\ldots,K\} \quad (6)$$

$$\sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} r_k \le x_{ijk}\,(t_{ij}+f_i+w_i) \le r'_k$$

$$\text{for } k \in \{1,\ldots,K\} \quad (7)$$

$$t_0 = w_0 = f_0 = 0 \quad (8)$$

$$\sum_{k=1}^{K} \sum_{i=0, i\neq j}^{K} x_{ijk}\,(t_i+t_{ij}+f_i+w_i) \le t_j$$

$$\text{for } j \in \{1,\ldots,N\} \quad (9)$$

$$e_i \le (t_i+w_i) \le l_j \quad \text{for } i \in \{1,\ldots,N\} \quad (10)$$

3. 차량일정계획시스템

본 시스템의 엔진개발은 ILOG Solver 5.1와 ILOG Dispatcher 3.1를 사용하여 Pentium III 866MHz CPU, RAM 256MB를 장착하고 OS로 Windows2000를 사용하는 PC에서 개발하였다.

3.1 초기해생성

ILOG Dispatcher엔진에서 제공하는 초기해 생성방법에는 차량을 본점에서 가장 가까운 고객부터 배차하는 Nearest Depot 알고리즘과, 연속해서 고객과 거리가 가장 짧은 거리로 경로를 구성하는 Nearest Addition 알고리즘, Clarke 과 Wright(1964)의 Savings 알고리즘, Wren(1971)의 Sweep 알고리즘, 전체 고객들을 한 명씩 차례로 삽입시켜 경로를 구성하는 Insertion 알고리즘, 탐색영역전체를 탐색하는 방법 등을 제공한다. 본 연구에서 Insertion 알고리즘과 탐색 영역전체를 탐색하는 알고리즘은 효율적이지 못하고 계산시간이 너무 오래 걸리는 단점이 있기 때문에 초기해를 구하는 알고리즘에서는 제외하였다.

초기해를 구한 4가지의 알고리즘 중 Nearest Addition 알고리즘이 가장 우수하게 나왔으며, 사용된 차량의 수는 본점에 있는 56대 모두를 사용하였다. 각 알고리즘에 대한 계산시간과 해는 <표 1>과 같다.

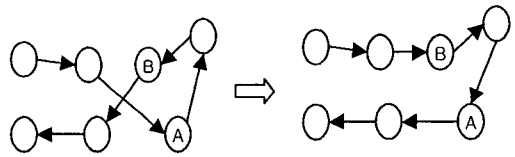
알고리즘 / 시간및해	Nearest Depot	Nearest Addition	Savings	Sweep
시간(초)	33.62	19.82	1279.12	34.68
해	34281.8	31082.9	34204.2	35652.1
사용한 차량대수	56	56	56	56

표 1. 초기해 결과

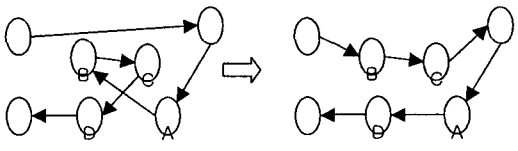
3.2 해의개선알고리즘

초기해를 개선시키는 알고리즘으로 반복적 향상 알고리즘(Improvement Algorithm)을 사용하였다(Shaw, 1997; ILOG Dispatcher 3.1 User's Manual). 같은 경로 내에서는 2-Opt와 Or-Opt 알고리즘을 사용하였고, Cross, Exchange, Relocate 알고리즘을 사용하여 경로간 해를 개선 시켰다 <그림 2>.

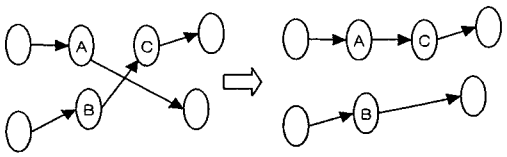
- 2-Opt : 경로 내에서 2곳의 경로(Arcs)를 제거한 후 새롭게 경로를 구성한다.
- Or-Opt : 경로 내에서 여러 곳의 경로(Arcs)를 제거 한 후 새롭게 경로를 구성한다.
- Cross : 서로 다른 두 개의 경로에서 두 경로의 적당한 끝 부분을 서로 바꾼다.
- Exchange : 서로 다른 두 개의 경로에서 여러 경로의 적당한 끝 부분을 서로 바꾼다.
- Relocate : 서로 다른 두 개의 경로에서 한 경로 속한 노드를 다른 경로로 이동시킨다.



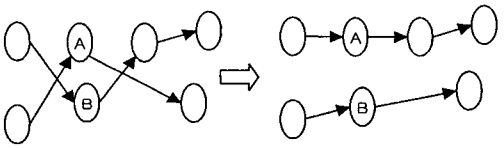
(a) 2-Opt



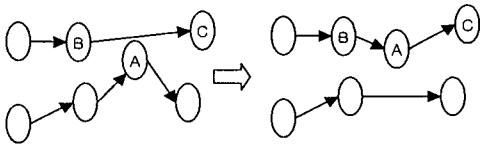
(b) Or-Opt



(c) Cross



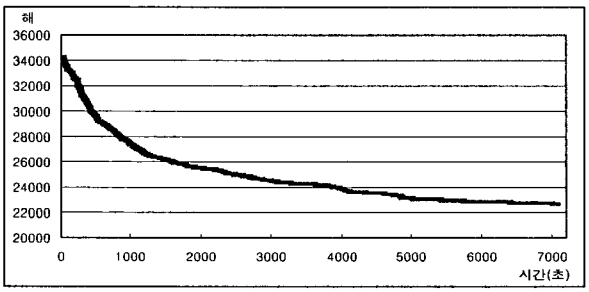
(d) Exchange



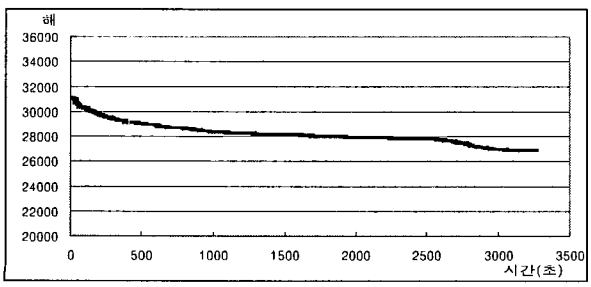
(e) Relocate

그림 2. 반복적 향상 알고리즘

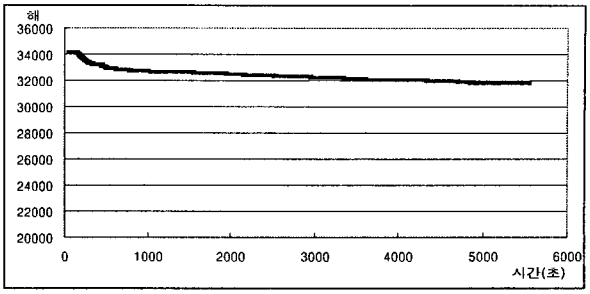
각 알고리즘에서 구한 초기해를 반복적 향상 알고리즘으로 개선시킨 결과, Nearest Depot 알고리즘을 사용한 초기해가 가장 많이 개선되었다. 또한 해의 개선이 가장 안된 것은 초기해를 Savings 알고리즘으로 구한 것으로 5498초까지 해가 느린 속도로 개선되었다. 종료조건으로는 전체 7200초 동안 개선시키도록 하였으며, 만약 해의 개선이 600초 이상 이루어지지 않으면 종료시켰다 <그림 3>.



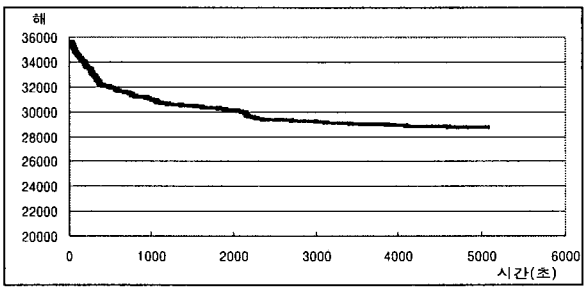
(a) Nearest Depot Algorithm



(b) Nearest Addition Algorithm



(c) Savings Algorithm



(d) Sweep Algorithm

그림 3. 반복적 향상기법에 의한 해의 개선 그래프

각 알고리즘 별 초기해의 개선 상황은 Nearest Depot 알고리즘으로 구한 해가 약 33.7% 가량 개선이 되었으며, 차량의 수도 56대에서 44대로 획기적으로 줄어들었음을 알 수 있다. 하지만 해의 탐색시간이 7029초로 다른 알고리즘에 비해 많이 걸림을 알 수 있다 <표 2>

알고리즘 / 개선해	Nearest Depot	Nearest Addition	Savings	Sweep
시간(초)	7029	3240	5498	5024
개선된해	22726.1	26879	31821.1	28711.8
개선율(%)	33.7%	13.5%	7.0%	19.5%
차량대수	44	51	56	54

표 2. 반복적 향상기법에 의한 해의 개선

3.3 메타휴리스틱

본 연구에서는 지역해를 빠져 나오기 위해 타부서치(Tabu Search)와 GLS(Guide Local Seach)을 사용하였다. 타부서치는 조합최적화문제(Combinatorial Optimization Problem)들에서 폭 넓게 응용되고 있는 제3세대 인공지능(AI)기법으로써 특히 차량경로문제에 좋은 해(Near Optimal Solution)를 제공해 준다(Backer *et al.*, 1997; Barbarosoglu *et al.*, 1999; Laporte *et al.*, 2000; Shaw, 1998).

임의의 해 s의 이웃 $N(s)$의 부분집합 U^*에서 최선의 해를 s'라하고, 함수를 f, 계산의 횟수를 k라 한다면 타부서치는 <그림 4>와 같다.

step1. Local Search로 구한 해를 s라 한다.
$s^* := s$, $k = 1$;
step2. 정지조건을 만날 때까지 수행
$k = k + 1$;
$U^* \subseteq N(s, k)$;
가장 좋은 s'를 U^*에서 고른다.
$s = s'$;
만약 $f(s') < f(s^*)$이면 $s^* := s'$;

그림 4. Tabu Search Algorithm

GLS는 타부서치와 유사한 금지비용(Penalty Cost)에 기반을 둔 발견적 기법(Metaheuristic)으로 TSP, Quadratic Assignment, Function Optimization, Partial Constraint Satisfaction, Resource Allocation등 조합최적화문제에서 우수한 알고리즘임이 증명되었다(Kilby *et al.*, 1997). 타부서치는 해를 가장 많이 개선시키는 요소(Factor)가 수정되려할 때 타부(Tabu)를 부여하여 해가 지역해를 탈출하도록 시도하는 반면, GLS는 이전에 탐색했던 지역해을 다시 탐색하려고 할 때 목적함수에 금지비용을 더하여 지역해를 탈출하려고 시도하는 기법이다.

임의의 해 s의 이웃 $N(s)$의 부분집합 U^*에서 최선의 해를 s'라하고, 목적함수를 f, 계산 횟수 k, k번째 비용벡터 c_k, k번째 페널티(Penalty) 벡터 p_k, 페널티요소(Penalty Factor)를 λ라 한다면 GLS <그림 5>와 같다. 그리고 식 $c_k/(p_k+1)$은 가장 비용이 많이 드는 아크(Arc)에게 페널티(Penalty)를 부여하게 된다.

step1. Local Search로 구한 해를 s라 한다.
$s^* := s$, $k = 1$;
step2. 정지조건을 만날 때까지 수행
$k = k + 1$;
$U^* = f(s_k) + \lambda \sum_{k=1}^{K} N(s)\ p_k\ c_k$;
가장 좋은 s'를 U^*에서 고른다.
$s = s'$;
만약 $f(s') < f(s^*)$이면 $s^* := s'$;

그림 5. Guide Local Search

일반적으로 차량경로문제에서 GLS가 타부서치보다 우수하다고 알려져 있으며 파라메터를 0.1 - 0.3으로 설정하는 것이 우수한 해를 탐색하는 것으로 알려져 있다(Backer *et al.*, 1997). 본 연구에서의 타부서치와 GLS의 종료조건은 계산횟수는 200회로 제한하였으며, 각 파라메터는 좋은 해를 탐색하기 위하여 계산횟수에 따라 변화시켜 보았다 <표 3>. 본 연구에서도 가장 좋은 해는 Nearest Depot 알고리즘을 초기해로 GLS을 사용한 해가 가장 우수한 해를 제공하고 있음을 알 수 있다. 하지만 타부서치에서는 지역해에 계속 머무르고 있음을 살펴볼 수 있다. 특히 지역해에서 해의 개선율이 가장 좋은 알고리즘은 Nearest Addition 알고리즘을 초기해로 지역해를 탈출시키기 위하여 타부서치로 개선시킨 것이 약 11%의 해의 향상을 가져왔다. 본 연구에서도 GLS로 탐색한 해가 타부서치 보다 더 좋은 해를 탐색했으며 <표 4>, <그림 6>과 <그림 7>은 Nearest Depot 알고리즘과 Nearest Addition 알고리즘을 반복적 향상기법을 이용한 지역해를 타부서치와 GLS가 해를 개선시키는 것을 보여주는 그래프이다. 특히 타부서치를 이용하여 해를 구할 경우 반복적 향상기법 중 Cross 알고리즘과 같이 사용할 경우 해의 사이클을 발생시킬 수 있기 때문에 주의하여야 한다(ILOG Dispatcher User's Manual).

알고리즘 \ 계산횟수	1-69	70-149	150-199
Tabu	12	20	5
GLS	0.2	0.15	0.2

표 3. Tabu Search와 GLS의 파라메터

알고리즘 \ 지역해		Nearest Depot	Nearest Addition	Savings	Sweep
Tabu	시간(초)	2726	1263	6938	5476
	해	22726.1	23703.2	31799	28529.1
	지역해 개선율	0%	11.8%	0.07%	0.63%
	차량대수	44	48	56	54
GLS	시간(초)	10106	310	8512	6198
	해	22689.3	26847.4	31801.1	28541.7
	지역해 개선율	0.16%	0.12%	0.08%	0.59%
	차량대수	44	51	56	54

표 4. Tabu Search와 GLS 결과

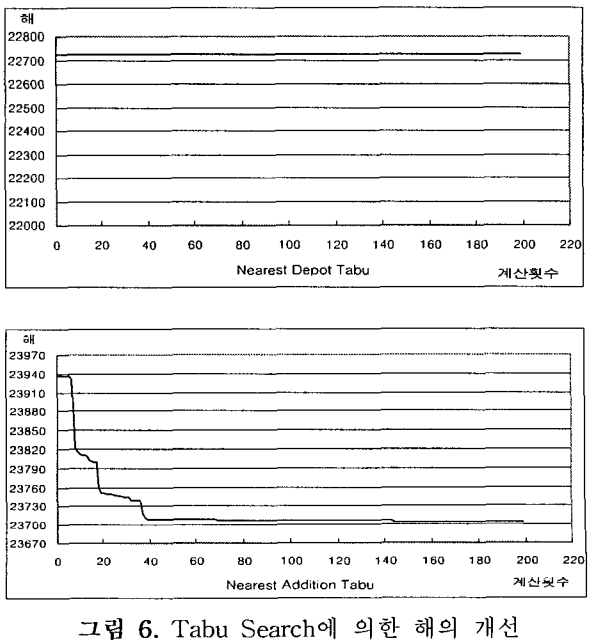


그림 6. Tabu Search에 의한 해의 개선

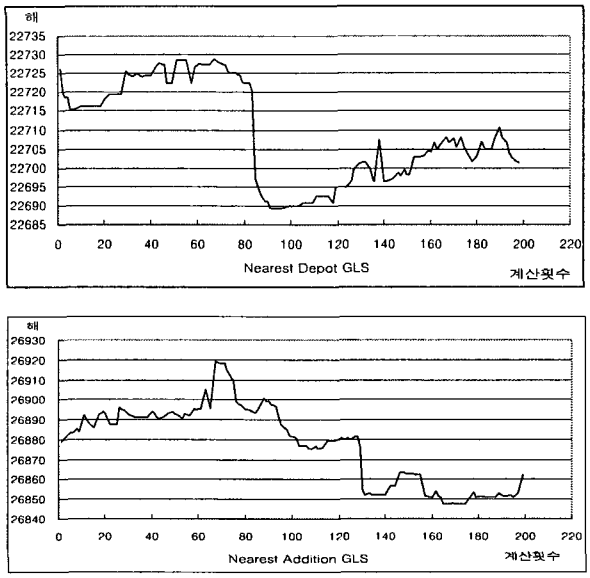


그림 7. Guide Local Search에 의한 해의 개선

3.4 Dynamic Scheduling 과 Rescheduling

앞서 설명한 알고리즘의 활용은 고객의 주문이 이미 알려져 있다는 상황에서 좋은해(Near Optimal Solution)를 탐색하기 위한 시도였다. 하지만 현실 상황에 효율적이고 능동적으로 대처하기 위해서는 동적일정계획(Dynamic Scheduling)과 재일정계획(Rescheduling)을 고려하여야 한다.

동적일정계획은 중요한 고객의 긴급주문이나 취소, 대리점의 재고고갈 등 긴급상황이 발생했을 때와 Thin Client에서 현재 차량의 배차 상황 및 용적 등을 실시간으로 고려하여 고객과의 납기시간 등을 효과적으로 제안하기 위하여 개발되었다. 동적일정계획은 실시간으로 모든 정보를 주고받기 때문에 빠른 시간에 가능해를 효과적으로 구할 수 있는 Farthest Insertion 알고리즘을 활용하였다. Farthest Insertion 알고리즘은 방문지 v가 삽입될 수 있는 위치의 집합을 $\tau_v = \{ p_1, \ldots, p_n \}$ 이라 한다면, v을 위치 p에 삽입시켰을 때 비용을 c_p라 한다면 c_p는 v의 수가 늘어날수록 커지게 되는데, c_p가 가장 적은 위치에 v를 삽입시키는 방법을 말한다.

재일정계획(Rescheduling)은 도로상황이나 차량의 고장, 운전기사의 결근, 고객의 주문시간 변경, 고객 서비스시간 지연 등의 상황에 효과적으로 대처하기 위해서 개발되었다.

차량이 본점에서 제품을 싣고 출발한 이후, 불가피한 상황, 즉 번잡한 교통상황 혹은 고객에게 제공할 예측 서비스시간이 급격히 늘어난 경우 그 이후 고객에게는 약속한 시간에 배달이 불가능하게 될 뿐 아니라, 본점에 회귀하여 2회차 운행을 할 때도 고객과 약속한 시간을 모두 어기게 된다. 따라서 본 차량일정계획시스템에서는 운전기사로부터 시간 지연에 대한 정보를 활용하여 2회차 운행에서 고객과의 약속 시간을 최대한 맞추기 위하여 재일정계획을 도입하게 되었다. <그림 8>은 ILOG Dispatcher를 활용한 재일정계획에 대한 예이다.

```
// 2회차 운행에서 이미 출발한 차량에 대해서는 경로를 고정한다.
IloVisit next = solution.getNext(visit);
IloConstraint nct = visit.getNextVar() == next;
mdl.add(nct);
//차량이 지연된 시간 만큼 2회차 운행에서 고려하여
//새롭게 경로를 구성하라는 제약
IloConstraint ct = visit.getCumulVar(time) <=
                    order->getMaxMinute () + delay;
mdl.add(ct);
```

그림 8. ILOG Dispatcher을 활용한 Rescheduling 예

4. 개발된 차량일정계획시스템

본 장에서는 사용자인터페이스(GUI) 중심으로 개발된 차량일정계획시스템을 설명하고자 한다.

사용자인터페이스(GUI) 화면은 크게 2가지로 구분이 되며, Fat Client는 ILOG JViews 5.0을 사용하였으며, Thin Client는 Tomcat를 사용하였다.

Fat Client의 화면 <그림 9>는 크게 3부분으로 구성이 되어 있다. 좌측 부분은 각 차량에 대한 정보를 가지고 있으며, 각 차량별 하루 작업지시서<그림 10>는 팝업(Pop-up) 메뉴로 제공하게 된다. 또한 차량에 대한 운행정보 및 차량에 대한 공차율 등을 알기 쉽게 구성되어 있다.

메뉴부분에는 중앙에 위치한 지도를 확대, 축소 할 수 있는 Zoom-in, Zoom-out 기능이 있고, 일정계획, 동적일정계획, 재일정계획 등의 기능을 수행 할 수 있도록 구성되어 있다.

중앙화면은 이미 알려진 고객에 대한 주문은 ILOG Dispatcher 엔진을 활용하여 좋은해(Near Optimal Solution)를 제공, 차량에 대한 경로를 나타내고, 새로운 고객이 삽입되어 동적일정계획을 하거나 납기 지연에 따른 재일정계획 등을 수행할 때 기존의 경로와 새로 구성된 경로를 보여 주는 것을 물론 각 차량이 어떤 경로를 몇 회차로 구성하는지 보여주고 있다.

마지막으로 화면 아래쪽은 간트 차트로 차량이 어떤 고객에게 어떤 순서로 배송을 하며, 설치시간, 대기시간, 운행시간 등을 알 수 있다. 이러한 정보는 실무자를 비롯한 관리자가 전체 현황을 신속하게 알아보기에 적합하게 구성이 되어 있다.

Thin Client는 웹(Web)상에서 시간대별 주문현황과 차량종류별 작업현황 및 이용률, 지역별 차량의 방문 대수를 살펴 볼 수 있으며, 고객에 대한 주문 및 조회, 차량별 고객서비스 및 작업조회, 주문 처리내역, 납기지연에 따른 시뮬레이션, 실시간으로 고객의 정보를 입력하여 차량의 실시간 배치 현황을 고객에게 서비스할 수 있다. <그림 11>은 주문 입력 양식의 팝업(Pop-up) 메뉴이고, <그림 12>는 차량별 작업조회 현황이다.

본 시스템은 이러한 기능 외에도 주문에 대한 시간대별 현황과 차량의 증차 여부 등 의사결정에 대한 여러 도움을 줄 수 있다.

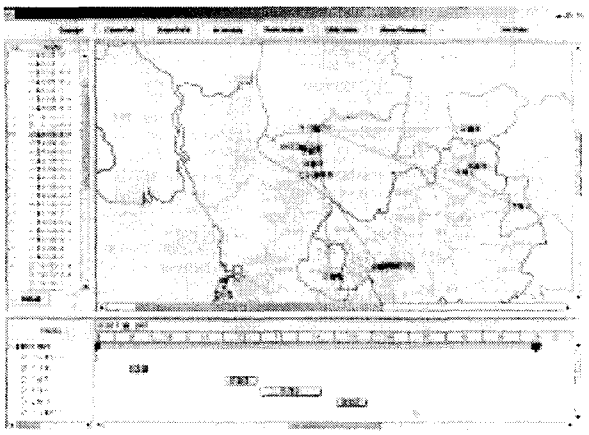


그림 9. 재일정계획된 Fat Client 화면

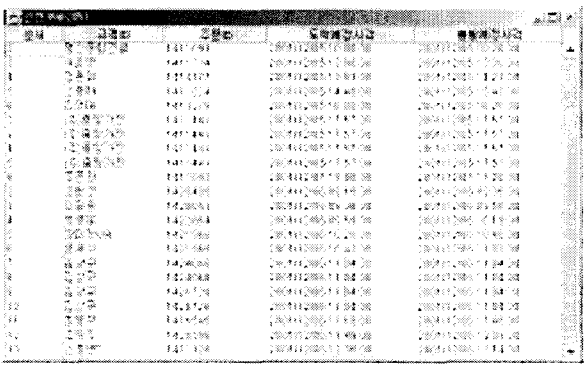


그림 10. 차량별 작업지시서 화면

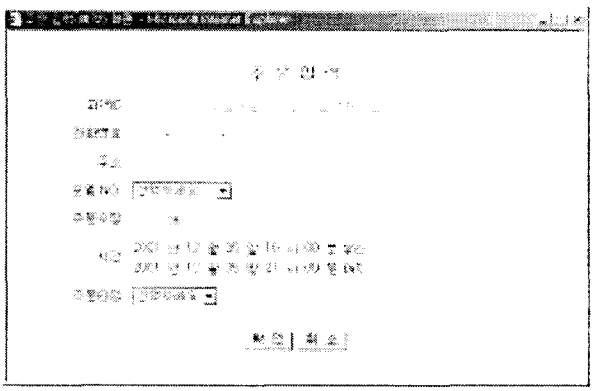


그림 11. 주문입력 Thin Client 화면

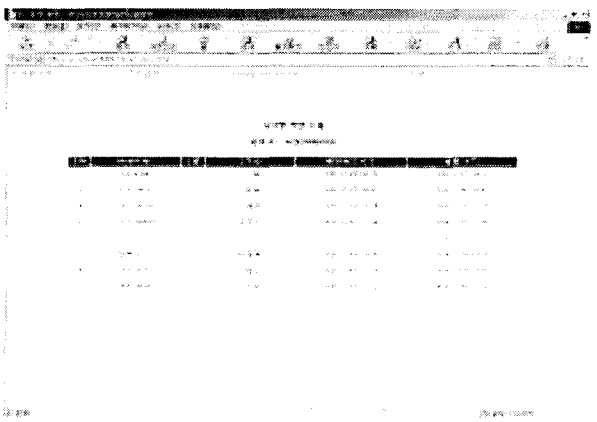


그림 12. 차량별 작업 조회현황 화면

5. 결론 및 향후 연구 방향

본 논문에서는 제약프로그래밍 기법에 따른 ILOG Solver와 ILOG Dispatcher, ILOG JViews, Tomcat 등을 활용하여 국내 종합물류회사인 T사를 대상으로 시간제약이 있는 차량경로문제모형을 산업현장에서 요구하는 현실상황을 충분히 반영한 시스템을 중심으로 설명하였다.

최적화 엔진부분에서는 제약프로그래밍 도구인 ILOG Dispatcher를 활용하여 초기해를 생성시킨 후 반복적 향상기법을 활용하여 해를 개선시켰다. 또한 지역해를 탈출하기 위해서 메타휴리스틱 기법인 타부서치와 GLS를 활용하였다. 초기해의 생성시 Nearest Depot 알고리즘과 Nearest Addition 알고리즘이 Savings 알고리즘과 Sweep 알고리즘보다 해를 탐색하는 시간과 결과 값이 우수하였으며, 반복적 향상 기법을 적용하여 구한 해도 우수하였다. 일반적으로 타부서치 보다 GLS 방법이 차량경로문제에서 우수함을 본 실험에서도 증명되었으며, 시간은 GLS가 타부서치 보다 오래 걸렸다. 또한 산업현장의 상황을 충분히 반영하기 위하여 동적일정계획(Dynamic Scheduling)과 재일정계획(Rescheduling)을 고려하였다.

향후 연구과제로는 본점이 여러 개(Multi-depot)인 경우와 운전기사의 기술습득능력에 따른 차량별 배치, Pick-up & Delivery등을 고려한 연구 등을 들 수 있다.

참고문헌

박순달, 정봉주, 장병만(1987), 관광버스 배차계획 Software(TBS), *한국경영과학*, 15(2), 201-210

양병희, 이영해(1994), 다목적 최적화를 고려한 배차계획시스템, *한국경영과학회지*, 19(3), 63-79

변의석(2000), 정시배송체계의 실용적 설계를 위한 방법론, *IE Interfaces*, 13(3), 445-461.

송성헌, 강승우(2001), 전자수치지도를 이용한 배차지원시스템 개발 및 활용, *IE Interfaces*, 14(1), 39-46.

Achuthan, N.R., Caccetta L., Hill S.P.(1997), On the Vehicle Routing Problem, *Nonlinear Analysis, Theory, Methods & Application*, 30(7), 4277-4288.

Barbarosoglu, G., Ozgur D.(1999), A Tabu Search Algorithm for the Vehicle Routing Problem, *Computers & Operations Research*, 26, 255-270.

Clark G, Wright, J.W.(1964), Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, *Operations Research*, 12, 568-581.

Dantzig, G. B., Ramser, J. H.(1959), The Truck Dispatching Problem, *Management Science*, 6(1), 80-91.

De Backer, B., Furnon, V., Kilby, P., Prosser, P., Shaw, P.(1997), Solving Vehicle Routing Problems using Constraint Programming and Meta-heuristics, *Journal of Heuristics*, 1-16.

Eilon S, Watson-Grandy C, Christofides N.(1971), Distribution Management: Mathematical Modeling

and Practical Analysis. *Hafner*, New York.

Golden, B., Wasil, E.(1987), Computerized Vehicle Routing in the Soft Drink Industry, *Operations Research*, 35, 6-17.

ILOG Dispatcher 3.1, User's Manual, ILOG.

ILOG Solver 5.1, User's Manual, ILOG.

Kilby, P., Prosser P., Shaw, P.(1997), Guided local search for the vehicle routing problem, *In Proceedings of the 2nd International Conference on Meta-heuristics.*

Laporte, G., Gendreau M., Potvin J-Y., Semet F.(2000), Classical and Modern Heuristics for the Vehicle Routing Problem, *International Transaction in Operational Research*, 7, 285-300.

List, G., Mirchandani, P.(1991), An Integrated Network/Planar Multiobjective Model for Routing and Siting for Hazardous Materials and Wasters, *Transportation Science*, 25(2), 146-156.

Rodriguez P., Nussbaum M., Baeza R., Leon G, Sepulveda M., Cobian A.(1998), Using Global Search Heuristics for the Capacity Vehicle Routing Problem, *Computers & Operations Research,* 25(5), 407-417.

Shaw, P.(1997), A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems, Goasgow, Scotland, University of Strathclyde,
<URL:http://www.cs.strath.ac.uk/~ps/GreenTrip/>

Shaw P.(1998), Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming*, 417-431.

Solomon M.(1987), Algorithms for Vehicle Routing and Scheduling Problems with Time Window Constraints, *Operations Research,* 35(2), 254-265.

Spasovic, L., Chien S., Kelnhofer-Feeley Y., Wang Y., Hu Q.(2001), A Methodology for Evaluation of School Bus Routing-A Case Study of Riverdale, New Jersey, *Transportation Research Board 80th Annual Meeting.*

Taillard E.(1993), Parallel Iterative Search Methods for Vehicle Routing Problem, *Networks* , 23, 661-676.

Tan, K.C, Lee, L.H, Zhu, Q.L, Ou, K.(2001), Heuristic Methods for Vehicle Routing Problem with Time Windows, *Artificial Intelligence in Engineering,* 15, 281-295.

Tilanus, B.(1997), *Information Systems in Logistics and Transportation,* Pergamon press.

William D., Harvey and Matthew L.(1995), Ginsberg, Limited Discrepancy Search, *Proceedings of the fourteenth International Joint Conference on Artificial Intelligence,* 606-615.