Journal of the Society of Korea Industrial and Systems Engineering
Vol, 29, No. 3, pp.62－69, September 2006.

시간제약이 있는 차량경로문제에 대한 Hybrid 탐색

이화기* · 이홍희* · 이성우* · 이승우**

*인하대학교 산업공학과
**한국기계연구원 지능형정밀기계연구부

Hybrid Search for Vehicle Routing Problem with Time Windows

Hwa-Ki Lee* · Hong-Hee Lee* · Sung-Woo Lee* · Seung-Woo Lee**

*Department of Industrial Engineering, Inha University
**Intelligence & Precision Machine Department, KIMM

Vehicle routing problem with time windows is determined each vehicle route in order to minimize the transportation costs. All delivery points in geography have various time restriction in camparision with the basic vehicle routing problem. Vechicle routing problem with time windows is known to be NP-hard, and it needs a lot of computing time to get the optimal solution, so that heuristics are more frequently developed than optimal algorithms. This study aims to develop a heuristic method which combines guided local search with a tabu search in order to minimize the transportation costs for the vehicle routing assignment and uses ILOG programming library to solve. The computational tests were performed using the benchmark problems.

Keywords : Vechicle routing problem with time windows, Tabu search, Guided local search

1. 서 론

오늘날 물류에 대한 관심이 증대되면서 물류비용의 감소를 위한 의사결정문제는 아주 중요한 문제로 대두되고 있다. 특히 공장, 분배센터, 지역 도·소매점 사이의 구성요소에 대한 방문 결정은 급변하는 현대사회에 빈번하고 신속하게 이루어져야 한다. 따라서 차량의 운행거리를 최소화하고, 운행차량의 수를 줄이며, 차량의 적재용량을 최대한으로 활용하기 위해서 차량경로문제(vehicle routing problem : VRP)에 대한 연구가 필요하다. 시간제약이 있는 차량경로문제(vehicle routing problem with time windows : VRPTW)는 기본적인 차량경로문제에 방문서비스 시간준수를 고려한 것으로써 중앙 차고지를 출발한 일련의 차량들이 지리적으로 산재해 있는 고객들이 요구하는 서비스 시간대에 맞추어 방문하여 주어진 서비스를 수행하고 다시 차고지로 돌아오는 최소 이동시간의 차량경로를 결정하는 문제이다. VRP 문제는 계산의 복잡함에 있어서 NP-hard 문제이고 VRPTW 역시 Solomon[10]이 지적했듯이 NP-hard 문제로써 최적해를 구할 수는 있으나 문제의 크기에 따라 계산량이 매우 커져 최적해를 구하기는 매우 제한적이다.

차량경로문제에 대해 Kohl et al.[9]과 Cook and Rich [3]가 최적해법으로 일부문제에 대해 최적해임을 입증하였으나 Chiang and Russell[1], Homberger and Cehring [6], Czech and Czarnes[4] 등 대부분의 학자들이 빠른 시간 내에 목적하는 수준의 해를 구하는 휴리스틱기법을 적용하고 있다.

본 연구에서는 타부탐색(tabu search)을 안내지역탐색(guided local search)과 접목시켜서 차량경로문제의 해를 구하는 발견적 해법을 제시하는 것이다.

2. 이론적 고찰

2.1 타부탐색(tabu search)

타부탐색 기법은 인간의 기억 과정을 이용한 복잡한 해 영역에서 근사 최적해를 찾기 위한 경험적 해법이다. 어려운 문제를 쉽게 다루고 많은 제약 사항을 동시에 고려할 수 있으며 다른 휴리스틱 과정을 제어할 수 있는 메타휴리스틱의 하나인 타부탐색은 Glover[5]에 의해 현재의 형태로 발전되었고, 작업자 스케쥴링, 전화통신 경로배정, job shop 스케쥴링, graph- coloring 등 다양한 분야에 적용한 결과가 발표되고 있다.

타부탐색은 언덕-등산 휴리스틱기법(hill-climbing heuristic)과 유사한 면이 많으며 언덕-등산 휴리스틱기법을 살펴보면 다음과 같다.

단계 1. 초기값 $s_0 \in X$을 선택한다.
단계 2. $c(s) < c(s_0)$를 만족하는 어떤 $s \in S(x)$를 선택한다.
단계 3. $s_0 = s$라 둔다. 단계 2로 돌아간다.

여기선 X는 해의 공간이고 S(x)는 이웃해 집합이다. 즉, 초기해 s_0를 현재해로 한 후 탐색을 시작하여 이웃해 집합을 구한다. 그리고 그 이웃해 중에서 최소화가 되는 이웃해이면 그 해를 현재해로 한 후 탐색을 계속하고 그렇지 않으면 끝낸다. 이 기법은 시작점에서 지역 최적해를 향하여 유일한 한쪽 방향으로만 진행하기 때문에 지역 최적해를 얻기는 하나 전체 최적해라는 보장이 없다는 한계점을 가진다. 이런 문제점을 보완하기 위해서 타부탐색은 통제에 관련된 기술로 메모리구조를 이용한 타부목록과 열망수준을 사용하여 자유롭게 탐색함으로써 지역 최적해에 머무르는 단점을 보완하고 있다.

타부목록이란 일반적으로 해의 이동정보를 나타내는 타부 속성을 일정기간 동안 기억하는 것으로, 최근에 생성된 한정된 해의 이동정보를 타부목록에 기억하는 단기기억(short term memory)과 처음부터 현재까지를 기억하는 장기기억(long term memory)을 이용하는 것 등이 있다. 본 논문은 단기기억 타부탐색 기법을 이용한 것으로 타부목록의 크기를 정해주어야 한다. 타부목록 크기는 타부속성의 개수를 나타낸 것으로써 정해진 개수만큼 타부속성이 채워지고 새로운 타부속성이 들어오면 기존의 타부목록에 있던 속성 중 가장 먼저 들어왔던 타부속성이 빠져나가는 선입선출 방식으로 변화한다. 그러나 이러한 타부목록의 사용이 도리어 좋은해가 있을 수 있는 곳으로 이동하는 것을 방해할 수 있다. 이런 한계를 극복하기 위해서 어떤 때에는 타부목록을 무시할 수 있는 새로운 조건이 필요한데 이것을 열망수준이라고 한다. 열망수준은 현재의 해가 tabu 상태에 있으나 해가 어느 수준 이상이면 해의 이동을 허락하는 기준으로 사용되는데, 일반적으로 현재까지 발견된 가장 좋은 해를 열망수준으로 이용한다.

일반적인 단순 타부서치의 절차를 살펴보면 다음과 같다.

단계 1. 초기화
(1) 타부속성, 타부목록크기, 열망수준과 종료조건 등을 결정한다.
(2) 초기해를 구해 현재해와 최선해로 둔다.
(3) 타부목록을 비워둔다.

단계 2. 이웃해 생성
현재해로부터 타부이동이 아니거나 타부이동이지만 열망수준을 만족하는 이동에 대해 이웃해를 생성한다.

단계 3. 이웃해 평가
(1) 생성된 이웃해 중에서 가장 좋은해를 현재해로 한다.
(2) 현재해가 최선해보다 좋으면 현재해를 최선해로 둔다.

단계 4. 타부목록 수정
새로운 현재해의 이동속성을 타부목록에 기록한다. 그리고 만약 타부목록에 저장된 타부속성의 수가 타부목록 크기보다 크면 먼저 기록된 타부속성을 삭제한다.

2.2 안내지역탐색(guided local search)

안내지역탐색(GLS)은 제약만족문제나 최적화문제에서의 해법을 위한 메타 확률 탐색(meta-stochastic search) 해법중의 하나이다. GLS는 언덕-등산 알고리즘 등이 지역해(local optima)에서 벗어나도록 제안된 관리 전략이라 할 수 있다[11].

GLS는 벌점(penalty terms) 집합을 포함한 문제의 확장된 비용 함수를 통하여 탐색영역에서 국부적 탐색을 안내하는 탐색과 관련된 정보에 이점을 가진다. 국부적 탐색은 벌점에 의해 제한되어지고, 탐색영역의 발전된 가능한 영역으로 집중시킨다. GLS에서의 탐색 방법은 반복을 통하여 국부적 탐색을 실행하고, 각 탐색에서 국부적 최소해를 얻어 다시 벌점을 수정하고, 수정된 비용함수를 최소화하기 위한 탐색을 다시 실행한다. 여기서 벌점 수정은 탐색동안 얻어진 정보나 사전 정보에 따른 탐색에서 얻어진 해를 조정하는 것이다. 사전 정보를 통하여 문제를 정의하는 제약식으로 전환하고, 고

려되는 해의 후보의 수를 줄이게 된다. GLS는 이 정보를 기초로 하여 탐색동안 또다른 제약식들을 추가함으로써 얻어진 정보를 이용한다.

GLS는 SA나 Tabu 탐색과 같은 다른 메타 휴리스틱과는 달리 국부적 탐색의 내부 작용을 수정하지 않는다. 대신 국부적 탐색의 절차에서 연속적인 반복을 통한 비용함수를 수정한다. 탐색을 하기 전 GLS는 해를 유동적으로 제약 가능하게 하기 위하여 벌점(penalty terms)을 포함한 확장된 비용함수로 변환한다.

그러므로 GLS는 해를 특정지우기 위해 비용에 관한 특성을 다음의 식 (2.1)과 같이 특정지우고 벌점을 포함한 확장된 비용함수로서 특성 f_i를 제약한다.

$$f_i = I_i(s_*) = \begin{cases} 1, & \text{해 } s_* \text{가 특성 } i \text{를 가질때} \\ 0, & \text{그렇지 않으면} \end{cases}, \quad s \in S \cdots (2.1)$$

확장된 비용함수(augmented cost function)는 다음과 같이 정의한다.

$$h(s) = g(s) + \lambda \cdot \sum_{i=1}^{M} p_i \cdot I_i(s) \cdots\cdots (2.2)$$

여기서 g(s)는 벌점을 부과하기전의 원래 비용함수이고, M은 해에서 정의된 특성들의 수이며, Pi는 특성 f_i와 표준(regularization) 파라미터 λ에 대응하는 벌점 파라미터이다.

f_i는 지역 탐색에 의해 방문되어지는 해들이 어떤 국부적 최소에 포함된 탐색 절차에 대한 정보를 나타낸다. 즉 국부적 최소에 도달했다 가정하면 f_i=1이 되고 해의 비용 특성을 고려하여 벌점을 부과하기 위한 특성을 나타내는 것이다. 또한 λ는 탐색 절차상에서 해에 대한 정보의 영향을 제어하기 위한 방법을 제공하고 해의 비용에 관한 벌점의 중요한 관계를 나타낸다. 즉 벌점을 이용하여 원래의 비용함수와의 관계를 생각하여 알맞은 표준을 정하여 지역해를 벗어날 수 있는 해의 개선을 나타낸다.

GLS는 반복적으로 지역 탐색을 실시하고 벌점을 수정한다. 각각의 지역 탐색은 국부적 최소해를 형성하고, 조정은 정보의 기본을 만든다.

처음에 모든 벌점 파라미터들은 0으로 형성(즉 어떠한 특성들도 제약을 받지 않는다.)되고 확장된 비용 함수의 국부적 최소를 찾기 위해 지역 탐색을 실시한다. 처음에 국부적 최소와 계속되는 각각의 국부적 최소점 이후에 알고리즘은 확장된 비용 함수에 관해 조정을 실시하고, 다시 지역 탐색을 재실시한다. 조정은 하나 또는 그 이상의 국부적 최소에 벌점 파라미터를 증가시키게 된다. 조정이 실시되기 전의 정보는 선택되어진 곳에 벌점 파라미터를 증가시킴으로써 확장된 비용 함수에 점차적으로 적용시키게 된다.

특성들의 수를 가지는 어떤 특정의 국부적 최소 s_*가 있다고 하면, f_i=1, 즉 $I_i(s_*)=1$이 되며, 국부적 최소 s_*에서 벌점 파라미터는 다음의 식 (2.3) 표현이 최대가 되는 모든 특성들 f_i에 대해 하나씩 증가된다.

$$util(s_*, f_i) = I_i(s_*) \cdot \frac{c_i}{1+p_i} \cdots\cdots (2.3)$$

즉, 특성 f_i의 벌점 파라미터의 증가는 식 (2.3)의 함수 $util$에 조정을 고려한다는 것이다. 여기서의 $util$은 식 (2.2)에서의 p_i의 벌점 조정을 위한 조정을 나타내고, 국부적 최소에서 최대 $util$을 가진 조정이 선택되어지고 행해진다. 벌점 파라미터 p_i는 전체적으로 높은 비용의 특성들에만 편향되게 벌점을 부여하는 것을 막기 위하여 같이 고려된다. 식 (2.3)에서 벌점 파라미터의 역할은 어떠한 특성이 몇 번의 벌점이 부과되었는지를 세는 카운터이다. 어떠한 특성이 탐색과정의 반복 동안 여러번의 벌점을 부과 받았다고 하면 $\frac{c_i}{1+p_i}$의 식에서 해당되는 특성은 감소되고, 따라서 다른 특성들로 벌점이 부과되는 선택의 기회가 주어지게 되는 것이다.

3. 제안된 차량경로문제 알고리듬

3.1 연구모델의 변수정의와 정식화

[기호]

V : 사용 가능한 차량 대수
N : 고객 지점 수
i,j : 고객 지점 번호($i, j = 0, 1, 2, \cdots N, N+1$)
i, j가 0인 경우 중앙창고(*Depot*)를 의미하며
i, j가 $N+1$인 경우 모든 차량이 귀환하는 가상 *Depot*를 나타내며 중앙 *Depot*와 동일한 위치이다.
T : 차량 이동 시간 한계
C : 차량 적재 용량
D_i : 고객 i의 배달물량
t_{ij} : 고객 i에서 고객 j로의 이동시간
s_i : 고객 i의 서비스시간
e_i : 고객 i의 서비스시간대 하한
l_i : 고객 i의 서비스시간대 상한
M : 매우 큰 숫자 상수

[변수]

$x_{ijk} = \begin{cases} 1, & \text{차량 } k\text{가 고객 } i\text{에서 고객 } j\text{로 이동시} \\ 0, & \text{이동이 없을시} \end{cases}$

q_{ik} : 차량 k가 고객 i에 도착할 때의 총 적재물량

z_{ik} : 차량 k가 고객 i에 도착 때의 총 잔여 배달량

a_{ik} : 고객 i에 차량 k의 도착시각

앞에서 사용된 기호와 변수를 이용하여 수리 모형을 다음과 같이 표현한다.

$$Min\ Z = \sum_{k=1}^{V} a_{N+1,k} \cdots\cdots (1)$$

s.t

$$\sum_{k=1}^{V}\sum_{i=0}^{N} x_{ijk} = 1, \quad k = 1, \cdots, N,\ j \neq i \cdots\cdots (2\text{-a})$$

$$\sum_{k=1}^{V}\sum_{j=1}^{N+1} x_{ijk} = 1, \quad k = 1, \cdots, N,\ i \neq j \cdots\cdots (2\text{-b})$$

$$\sum_{j=1}^{N+1} x_{0jk} = 1, \quad k = 1, \cdots, V \cdots\cdots (3)$$

$$\sum_{k=1}^{V}\sum_{j=1}^{N+1} x_{0jk} = V \cdots\cdots (4\text{-a})$$

$$\sum_{k=1}^{V}\sum_{i=0}^{N} x_{i,N+1,k} = V \cdots\cdots (4\text{-b})$$

$$q_{ik} \leq C, \quad k = 1, \cdots, V,\ i = 0, \cdots, N+1 \cdots\cdots (5)$$

$$a_{N+1,k} \leq T, \quad k = 1, \cdots, V \cdots\cdots (6)$$

$$q_{jk} - q_{ik} \leq M(1 - x_{ijk}) - D_i, \quad k = 1, \cdots, V,\ i = 0, \cdots, N,\ j = 1, \cdots, N+1,\ i \neq j \cdots\cdots (8)$$

$$z_{jk} - z_{ik} \leq M(1 - x_{ijk}) - D_i, \quad k = 1, \cdots, V,\ i = 0, \cdots, N,\ j = 1, \cdots, N+1,\ i \neq j \cdots\cdots (9)$$

$$a_{jk} - a_{ik} \geq t_{ij} + s_i - M(1 - x_{ijk}), \quad k = 1, \cdots, V,\ i = 0, \cdots, N,\ j = 1, \cdots, N+1,\ i \neq j \cdots\cdots (10)$$

$$z_{N+1,k} = 0, \quad k = 1, 2, \cdots, V \cdots\cdots (11)$$

목적함수 (1)은 총 차량운행시간을 최소화한다. 식 (2-a)와 식 (2-b)는 고객지점에서 단지 한 대의 차량만이 도착하고 출발하도록 한다. 식 (3)은 모든 차량이 중앙창고로부터 출발하도록 한다. 식 (4-a)와 식 (4-b)는 창고를 출발한 차량대수와 창고로 돌아온 차량대수를 모두 V로 동일하게 한다. 식 (5)는 차량적재능력을 초과하지 못하도록 하고 식 (6)은 차량운행시간을 제한한다. 식 (7)은 각각의 고객지점에서 배달작업 후 차량적재물량의 변화를 계산한다. 식 (8)은 창고를 출발하는 차량의 적재물량을 정의하고, 식 (9)는 배달지점에서 작업 후 차량에 적재되어 있는 잔여 배달물량을 계산한다. 식 (10)은 각각의 차량이 고객에게 배달 후 이동하여 다음 고객에게 도착하는 시간을 계산한다. 마지막으로 식 (11)은 차량이 depot에 도착할 때 차량에 배달물량이 남지 않도록 보장한다.

3.2 Hybrid 탐색 절차

본 장에서는 타부탐색 기법에 안내지역탐색 기법을 접목하여 차량경로문제에 적용하기 위해서 제안한 알고리듬을 제시한다. <그림 1>은 제안한 알고리듬 과정을 도식화 한 것이다.

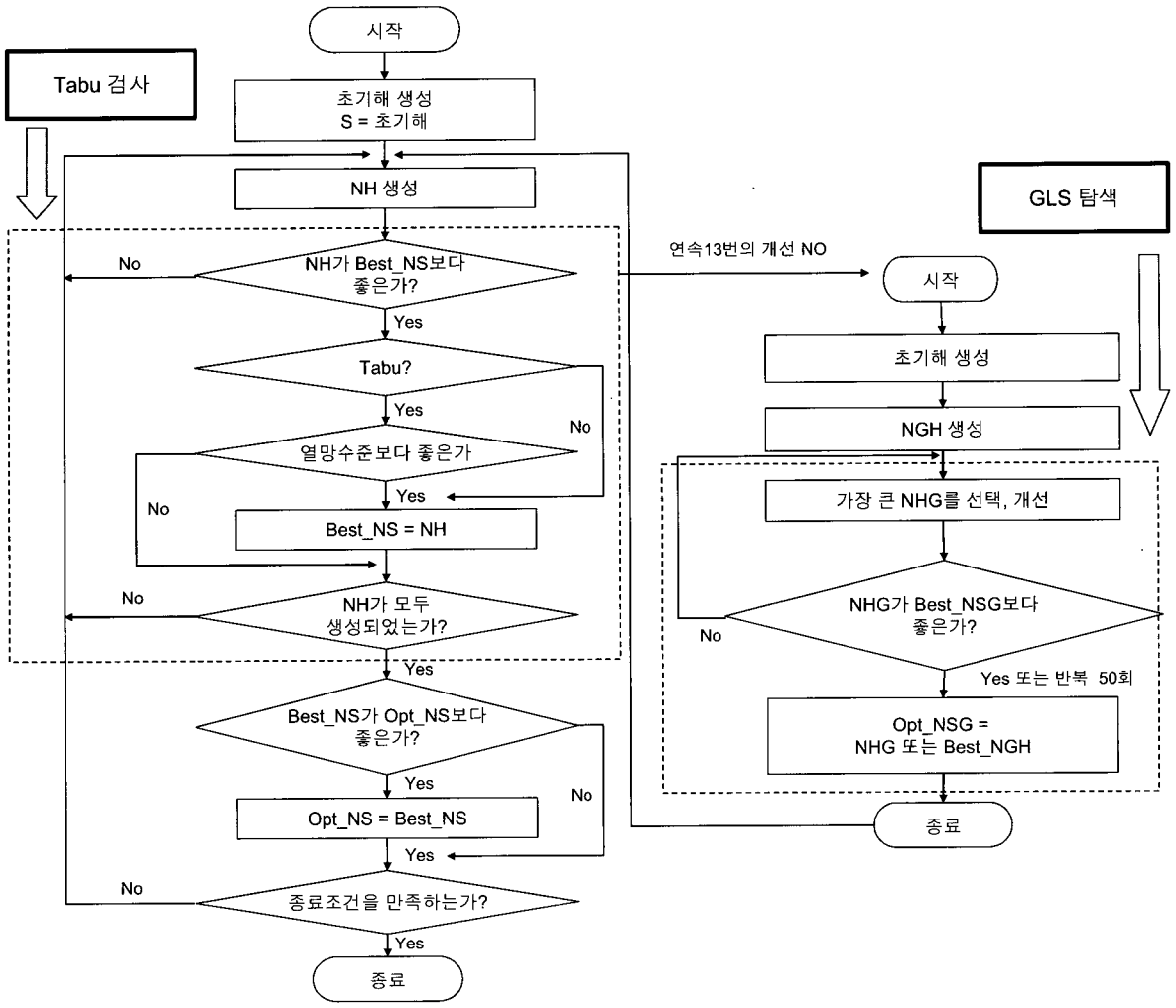
〈여러 가지 기호〉

- NH : 현재해에서 생성한 하나의 이웃해
- Best_NS : 현재해로부터 생성한 모든 이웃해 중 가장 좋은 이웃해
- Opt_NS : 최선해로써 종료조건을 만족할 때까지 생성한 Best_NS중 가장 좋은 해
- NHG : 안내지역탐색에서 현재해로 생성한 하나의 이웃해
- Best_NSG : 안내지역탐색에서 현재해로부터 생성된 모든 이웃해중 가장 좋은 이웃해
- Opt_NSG : 안내지역탐색에서 최선해로써 종료조건을 만족할 때까지 생성한 Best_NSG 중 가장 좋은 해
- S : 현재해
- M(S) : 현재해의 총 경로 배정비용

[Phase Ⅰ] : 초기화

단계 1. 초기해를 구한다. Clarke and Wright[2]의 절약해법(savings algorithm)을 적용하여 생성된 배정절차를 초기해로 한다. 비용절약 알고리즘은 중앙창고(depot)와 고객을 바로 연결하는 N개의 직접 경로로부터 시작하여 임의의 두 개의 차량경로를 합칠 경우 비용이 절약되면 두 개의 경로를 합쳐 새로운 차량 경로를 만다는 과정을 반복하는 알고리즘이다.

단계 2. 여기서 구한 해와 경로를 현재해와 최선해로 저장한다.



<그림 1> 제안된 알고리듬 절차

[Phase Ⅱ] : 이웃해 생성

단계 1. 초기해에서 NH들을 모두 초기화한다.

단계 2. 현재해에서 선택 가능한 모든 이웃해 중에서 비용이 가장 큰 경로를 선택 한다.

단계 3. 선택된 경로를 재배치(relocation), 교환(exchange)과 2-Opt의 방법으로 이웃 해를 개선한다. 2-opt 방법은 임의의 한 경로를 선택하여 경로상의 2개의 가지(arc)를 제거하고 새로운 2개의 가지를 추가하여 경로의 효율성 유무를 판별하여 선택하는 교환 이동 방식이며, 교환(exchange) 방법은 임의의 두 경로를 선택하여 각 경로상의 가능 마디(node)쌍을 경로 간에 교환하여 경로의 효율성 유무를 선택하는 방식이다. 한편, 재배치(relocation)방법은 임의의 한마디(node)를 다른 경로에 재배치하여 경로의 가능성 유무를 선택하는 삽입이동 방식이다.

[Phase Ⅲ] : 타부탐색

단계 1. 이웃해 생성절차 단계에서 생성된 하나의 현재해(S)에서 타부속성과 M(S)를 구한다. 만약 M(S) < M(Best_NS)이면 단계 2로 넘어가고 그렇지 않으면 단계 5로 간다.

단계 2. 설정한 타부목록의 크기만큼 타부속성을 쌓아가고 현재의 타부속성이 현재의 타부목록에 포함 여부를 검사한다.

단계 3. M(S) < 열망수준 이면 열망수준을 M(S)로 하고 다음 단계로 간다.

단계 4. Best_NS = S로 하고 다음 단계로 간다.

단계 5. 이웃해가 모두 생성되었으면 끝내고 그렇지 않으면 단계 1로 간다.

[Phase Ⅳ] : 안내지역탐색 검사

단계 1. 타부탐색에서 13회 연속 해의 개선이 없을 때

타부탐색에서의 현재 Best_NS를 안내지역탐색상에서 현재의 초기해로 한다. 단, 타부탐색에서 안내지역탐색으로의 진행상 순환을 방지하기 위하여 타부탐색에서 안내지역 탐색으로 진행할 때 같은 Best_NS가 다시 안내지역탐색의 현재해가 될 때 Best_NS가 아닌 Best_NSG가 안내 지역탐색의 현재해가 되고, 다시 타부탐색에서 안내지역탐색으로 진행시 Best_NS 나 Best_NSG 둘 중에 하나와 같을시 타부탐색 진행에서의 현재해를 초기해로 한다.

단계 2. 초기해에서의 NHG들을 생성한다.

단계 3. 가장 큰 NHG를 선택하고 그 노드를 재배치(Relocation), 교환(Exchange)과 Two-Opt의 방법을 통하여 이웃해 를 개선한다. 이 때 전에 방문했던 노드를 연속 방문시 벌금(penalty) 0.2를 부여해 노드의 반복을 피한다.

단계 4. 안내지역탐색 내에서 생성된 하나의 S에서 M(S)를 구한다.
M(S) < M(Best_NSG)이면 다음 단계로 가고 그렇지 않으면 단계 3으로 간다.

단계 5. 단계 3과 4의 반복이 50회가 되거나, 단계 4에서의 바로 다음 단계로의 진행이었을 때 Opt_NSG를 타부검사에서 현재해로 주어지면서 다시 타부검사로 진행한다. 이 때 타부검사로의 진행시 순환을 방지하기 위하여 두 번째 안내 지역탐색에서 타부검사로의 진행에서 같은 Opt_NSG가 타부검사에서 현재해로 되었을 때 Opt_NSG가 아닌 안내지 역탐색에서의 현재 진행 중인 해를 타부검사에서의 현재해로 정한다.

4. 실험 및 분석

4.1 실험과정

실험을 하기 전 개발한 해법은 3가지의 필요한 파라미터가 있다. 타부목록의 크기, 타부탐색에서 안내지역탐색으로 진행할 때 필요한 퇴화된 회수, 안내지역탐색 내에서의 반복 회수에 따라 해의 좋은 정도가 달라진다. 따라서 반복적인 실험을 통해 파라미터의 값을 정할 필요가 있다. 본 논문에서는 타부목록의 크기는 7, 9, 12, 15를 선택하고, 타부탐색에서 안내지역탐색으로의 퇴화회수는 8, 9, 10, 11, 13, 15를 선택, 안내지역탐색 내에서의 반복회수는 30, 40, 50을 선택하여 우선적인 실험을 하였다. Solomon[10]의 문제 중 R형에 대한 실험을 통해 가장 좋은 해가 많이 나오는 경우들을 종합하였으며, 타부목록의 크기는 12, 타부탐색에서 안내지역탐색으로의 퇴화회수는 13, 안내지역탐색 내의 반복회수는 50을 파라미터 값으로 사용하였다.

제안한 해법의 성능평가를 위한 데이터는 다음과 같이 선정하였다. C형은 모든 heuristic이 모두 동일한 성능을 보이고 있어 제외하였다. 여러 heuristic이 다양한 결과를 보여주는 R형을 실험 대상으로 선정하였으며, RC형 중에서 optimal이 알려진 RC형 1개와 optimal이 알려지지 않은 RC형 1개를 추가로 선정하였다. 이 문제들은 지리적으로 산재된 100개의 수요지로 구성되어 있다. 우선 각 문제들의 특성을 살펴보면 R타입 12문제는 3개의 그룹으로 구분된다. R101~R104 문제는 시간대 간격이 10, R105~R108 문제는 시간대 간격이 30, R109~R112 문제는 시간대 간격이 60이다. 또한 각 그룹에서 100개의 수요지 중에서 시간대를 요구하는 수는 각각 100개, 75개, 50개, 25개로 설정되어 있다. RC타입은 RC105와 RC106문제를 사용하였는데, RC106은 시간대 간격이 60으로 일정하나 RC105는 시간대 간격이 일정하지 않고 시간간격이 60보다 작은 수요지가 많다.

실험을 위해 사용된 컴퓨터는 Windows2000을 OS로 사용한 CPU P-Ⅲ 800Mhz 인 삼보 PC로 ILOG Solver 5.3과 ILOG Dispatcher 3.3 라이브러리[7, 8]를 Visual C++ 6.0에 구현하여 실험하였다.

4.2 실험결과 및 분석

실험에 사용된 Solomon[10]의 데이터에 대한 많은 연구가 이루어 있으나, 이중 4가지는 최적해가 알려져 있지 않다. 그러나 많은 휴리스틱 기법들에 의해 구해진 좋은 해들이 제시되어 있다.

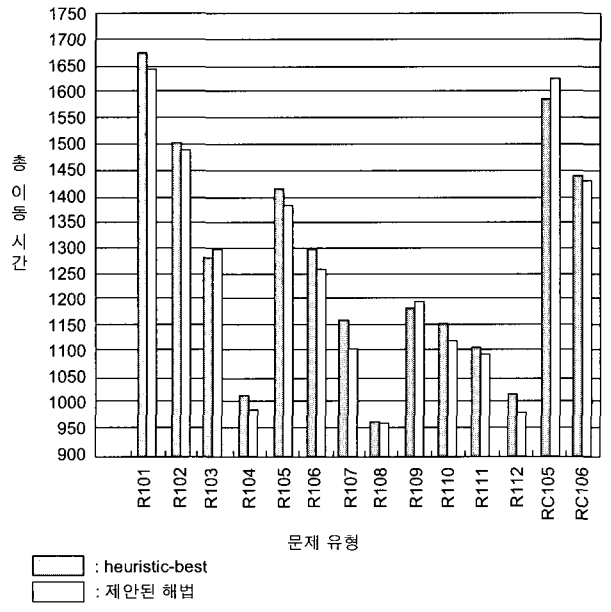
<표 1>은 제안한 알고리듬을 수행시간 600초에서 문제 유형에 따라 구해진 해들을 기존의 휴리스틱기법으로 구한 해들 중 가장 좋은 해들과 비교해본 표이다. 본 연구에서 제한시간 600초의 선정 이유는 제안된 알고리즘이 한 문제를 제외하고는 대부분의 문제들에서 600초 이후에서는 더 이상의 해의 개선이 거의 이루어지지 않았기 때문이다. 단지 R108문제의 경우, 2000초의 종료조건으로 구해진 해는 958으로 기존의 heuristic - best보다 좋은 값이 나왔으나 600초 제한시간의 해와 별 차이는 없었다. 우선 <표 1>을 토대로 만든 <그림 2>을 살펴보면 데이터 R103, R109, RC105에서 기존에 알려진 휴리스틱 Best 값보다 좀 더 좋은 해가 도출되었음을 알 수 있다. 그리고 나머지 다른 데이터에 대해서도 값의 차이가 크지 않다는 것을 파악할 수 있다.

<그림 2>에서 결과값을 통해 살펴보면 타부탐색에 안내지역탐색을 접목시킨 제안한 해법은 많은 참고 자

료[12]에서 정리되어 있는 다른 휴리스틱스 기법들과 비교하여 전혀 손색없는 성능을 나타내고 있음을 알 수 있다. 또한 600초에서 구한 값을 heuristics-best와 optimal과 비교하여 오차율을 구한 표가 <표 2>에 나타나 있다. 표를 통해 Optimal에서의 오차율의 평균이 4.97%로 R104는 964, R108은 918.3, R112는 966.2, RC106은 1369.9 정도에서 최적해가 이루어 질 것이라고 생각해 볼 수 있다.

<표 1> 600초 수행시간 결과값의 비교

NO	파일□	(a) 제안한 해법(600초)	(b) Heuristics Best	(c) Optimal
1	R101	1671.1	1645.79	1637.7
2	R102	1498.14	1486.12	1466.6
3	R103	**1275.32**	1292.68	1208.7
4	R104	1011.95	982.01	-
5	R105	1414.74	1377.11	1355.3
6	R106	1298.49	1252.03	1234.6
7	R107	1159.04	1104.66	1064.6
8	R108	963.95	960.88	-
9	R109	**1184.53**	1194.73	1146.9
10	R110	1149.7	1118.84	1068
11	R111	1107	1096.72	1048.7
12	R112	1014.21	982.14	-
13	RC105	**1591.68**	1629.44	1513.7
14	RC106	1437.98	1424.73	-



<그림 2> <표 1>의 수행시간의 결과값

<표 2> 성능평가를 위한 오차율 비교

NO	Heuristics와 비교오차(%) 100*{(a)-(b)}/(b)	Optimal과 비교오차(%) 100*{(a)-(c)}/(c)
1	1.53	2.04
2	0.8	2.15
3	*	5.5
4	3.05	-
5	2.73	4.39
6	3.71	5.17
7	4.92	8.87
8	0.32	-
9	*	3.28
10	2.76	7.65
11	0.94	5.56
12	3.27	-
13	*	5.15
14	0.93	-

주) * 본 연구에서 제한된 해가 더 좋은 경우

5. 결 론

본 연구에서 다루는 문제는 기본적인 차량경로문제 즉, 하나의 중앙창고에서 다수의 수요지점들의 수요량을 충족시키며 차량의 운행비용의 최소화를 목적으로 하는 문제에 시간제약을 추가한 것이다. 이 문제를 해결하기 위해 타부 탐색에 안내지역 탐색을 접목한 알고리듬을 제시하여 좋은 해를 도출하고자 하였다.

제안한 알고리듬의 성능 평가에 관련해서는 Solomon [10]의 R타입과 RC타입의 데이터를 사용하여, 다양한 휴리스틱 기법들에의해 도출된 것 중 가장 좋은 해들과 비교해 보았다. R103, R109, RC105 데이터는 기존의 해보다 나은 해를 도출하였고 나머지 다른 데이터들도 적절한 시간 내에 근사하게 접근하였다. 이것을 통해 제안한 알고리듬의 성능은 확인되었고 실험하지 않은 다른 데이터에도 충분히 적용하고 해를 도출할 수 있을 것이다.

추후 연구 과제로서는 타부 탐색에서 안내지역 탐색으로 전환할 때 제한 횟수가 해의 개선에 관련해서 어느 정도의 영향을 끼치는지에 대해 심도깊은 연구를 통해, 좀 더 나은 해를 보장하는 해법을 개발하는 것과 기본적인 차량경로문제에 추가되는 여러 가지 제약에 대해서 제시한 알고리듬을 보완하여 해를 도출해 보는 과정 등이 있을 것이다.

참고문헌

[1] Chiang, W. C. and Russell, R. A.; "Simulated Annealing Metaheuristics for the VRPTW," *Annals of Operations Research*, 63 : 3-27, 1996.

[2] Clarke, G. and Wright, J. W.; "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, 2(4) : 568-581, 1964.

[3] Cook, W. and Rich, J. K.; "A Parallel Cutting Plane Algorithm for the Vehicle Routing Problem with Time Windows," Technical Report, Computational and Applied Mathematics, Rice University, Huston, TX, 1999.

[4] Czech, Z. and Czarnas, P.; "Parallel Simulated Annealing for the Vehicle Routing Problem with Time Windows," 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands-Spain, pp. 376-383, 2002.

[5] Glover, F.; "Tabu Search : A Tutorial," *Interfaces*, 20(4) : 74-94, 1990.

[6] Homberger, J. and Gehring, H.; "Two Evolutionary Metaheuristics for the Vehicle Routing Problems with Time Windows," *INFOR*, 37 : 297-318, 1999.

[7] ILOG; "ILOG Dispatcher manual," ILOG, 2001.

[8] ILOG; "ILOG Solver manual," ILOG, 2001.

[9] Kohl, N.J., Desrosiers, J., Madsen, O., Solomon, M., and Soumis, F.; "2-pathcuts for the Vehicle Routing Problem with Time Windows," *Transportation Science*, 33 : 101-116, 1999.

[10] Solomon, M.; "Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Operations Research*, 35(2) : 254-265, 1987.

[11] Voudouris, C.; 'Guided Local Search for Combinatorial Optimisation Problems," Ph.d Dissertation, Department of Computer Science, University of Essex, Colchester, United Kingdom, 1997.

[12] http://neo.lcc.uma.es/radi-aeb/WebVRP/