韓國經營科學會誌
第24卷 第2號
1999年 6月

서비스 납기가 주어진 다목적차량일정문제를 위한 혼성유전알고리듬의 개발*

박 양 병**

A Hybrid Genetic Algorithm for the Multiobjective Vehicle Scheduling Problems with Service Due Times*

Yang-Byung Park**

■ Abstract ■

In this paper, I propose a hybrid genetic algorithm(HGAM) incorporating a greedy interchange local optimization procedure for the multiobjective vehicle scheduling problems with service due times where three conflicting objectives of the minimization of total vehicle travel time, total weighted tardiness, and fleet size are explicitly treated. The vehicle is allowed to visit a node exceeding its due time with a penalty, but within the latest allowable time. The HGAM applies a mixed farming and migration strategy in the evolution process. The strategy splits the population into sub-populations, all of them evolving independently, and applys a local optimization procedure periodically to some best entities in sub-populations which are then substituted by the newly improved solutions. A solution of the HGAM is represented by a diploid structure. The HGAM uses a modified PMX operator for crossover and new types of mutation operator. The performance of the HGAM is extensively evaluated using the Solomons test problems. The results show that the HGAM attains better solutions than the BC-saving algorithm, but with a much longer computation time.

1. 서 론

차량일정문제(vehicle scheduling problems)는 지리적으로 산재해 있는 고객들의 욕구를 충족하기 위해서 차량들이 차고지(depot)로부터 고객들을 방문하는 최적의 경로를 결정하여 시간적으로 방

* 본 연구는 한국과학재단 핵심전문연구(981-1014-075-1) 지원으로 수행되었음.
** 경희대학교 기계·산업시스템공학부

문일정을 표현하는 모든 문제를 포함한다. 여기서 차량들은 할당된 고객들을 서비스한 후 출발한 차고지로 되돌아 온다. 차량일정문제는 추구하는 목적과 제약조건, 그리고 기타 환경요소에 따라 다양한 성격의 문제로 구축될 수 있으며, 물적 수·배송과 함께 서비스 제공문제에도 적용된다.

차량을 이용하여 고객을 서비스하는 경우 고객은 자신의 일정을 고려하여 서비스 받기를 원하는 납기(service due times)와 상한시각(latest allowable service times)을 지정할 수 있다. 고객의 입장에서 납기의 준수는 매우 중요한 문제이며, 납기위반에 따른 고객의 피해정도 및 불만도는 고객에 따라 다르게 나타날 수 있다. 서비스를 제공하는 입장에서는 고객이 요구하는 납기를 지키기 위해 차량과 인력을 추가로 필요하게 되며, 이로 인해 물류비용은 증가하게 된다. 따라서 차량소요대수의 최소화와 총가중순수납기지연(total weighted tardiness)의 최소화는 총차량이동시간의 최소화와 함께 동시에 그리고 명시적으로 다루어져야 할 차량일정문제의 중요한 목적들이 된다.

고객의 서비스 납기가 주어진 상황에서 이러한 세 가지 목적들을 동시에 고려하면서 차량일정을 계획하는 문제는 실제 기업의 수·배송활동에서 빈번히 발생하고 있는 문제로서, 최근에 들어 이에 대한 적절한 해법의 개발이 절실히 요구되고 있는 실정이다. 특히, 우리 나라의 대도시와 같이 도로망이 복잡하고 교통체증이 극심한 환경에서 차량대수와 차량이동시간 그리고 고객의 서비스 납기를 고려하여 차량일정을 효율적으로 계획하는 일은 기업의 물류비용 절감은 물론 고객 서비스수준의 향상 및 생산시스템의 원활한 활동을 위해서도 매우 중요하다. 예로서, 물류창고에서 백화점, 또는 도·소매점에 물품을 납품하는 문제, 원자재나 부품을 공급하는 문제, A/S 방문문제, 물품을 수거하는 문제 등을 들 수 있다.

유전알고리듬(genetic algorithm)은 복잡한 목적함수와 다양한 제약식을 갖는 문제에서 최적 해에 대한 탐색기능이 뛰어난 것으로 이론적 그리고 실험적으로 모두 증명되어 있다[1,9]. 특히, NP-hard 문제, 함수최적화 및 컨트롤 문제 등에 그 성능이 탁월한 것으로 알려져 있으며, 최근에는 공학, 과학, 경영학 분야에도 그 적용이 널리 확산되고 있다. 유전알고리듬을 이용하여 차량일정문제의 해를 구하는 연구는 최근에 여러 학자들에 의해 시도되고 있다.

Cheng and Gen[4]은 삼각분포를 가정으로 한 퍼지(fuzzy) 납기 개념을 토대로 고객서비스 만족도의 최대화를 추구하는 차량경로문제를 구성하여 유전알고리듬을 이용하는 해법을 제안하였다. Thangiah et al.[18]은 고객의 서비스시각 상한이 존재하는 차량경로문제에 대해 cluster-first route-second 발견적 해법을 개발하였는데, 첫 번째 단계에서 고객들의 그룹을 구성하는데 유전알고리듬을 이용하였다. Potvin and Dube[14]는 일반 차량경로문제에 대해 parallel insertion 기법을 적용하기 위해 필요한 여러 파라미터들의 최적 값을 유전알고리듬을 이용하여 구하였다. Blanton and Wainwright[3]는 서비스 시간대가 존재하는 차량일정문제에 기존의 order-based 유전알고리듬을 적용하면서 새로운 두 종류의 교차변이 연산자를 개발하여 그 성능을 평가하였다. Thangiah[17]는 유전알고리듬을 이용하여 차량 수에 해당하는 개수의 원을 구성한 후, 각 원에 속한 노드들의 차량경로를 결정하였다. 유전알고리듬은 각 원의 중심과 반지름을 결정하는데 사용되었다. Gabbert et. al[7]은 대규모 철도망에서 이동하는 화물수송기차의 일정계획을 위해 유전알고리듬을 적용하였다. Malmborg[10]는 차량들이 작업센터간의 물자이동 요구를 수행하는 경우 주어진 작업시간 동안에 각 작업센터들에서 수집된 물자가 목적지까지 배달되기까지 발생하는 총시간지연을 최소화하기 위한 차량일정계획에 유전알고리듬을 적용하였다.

비록 유전알고리듬이 차량일정문제에 탁월한 성능을 보이고 있지만, 유전알고리듬의 적용과정에서 해의 조기수렴이나 지나친 다양화 현상이 발생하여 해 탐색에 부정적 영향을 미치기도 하는 것으로

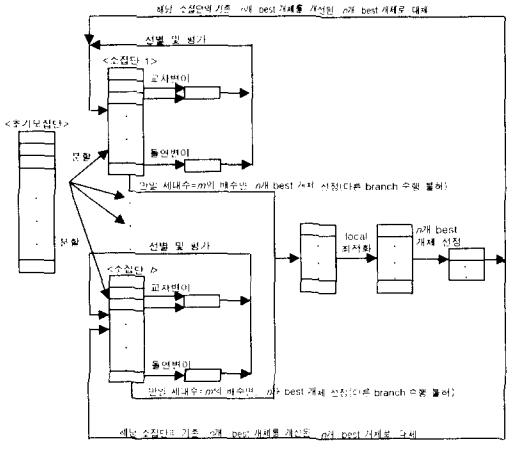
알려져 있다. 이러한 문제를 해결하는 한가지 방법으로서, 모집단에 대한 global exploration을 수행하기 위한 유전알고리듬과 각 개체에 대한 local exploitation을 위한 local 최적화 기법의 혼합된 사용을 고려할 수 있다. 즉, 유전알고리듬의 진화과정에서 생성된 각각의 자손들에게 local 최적화 기법을 적용하여 해의 개선을 도모하는 방법이다. 이러한 형태의 유전알고리듬을 혼성유전알고리듬(hybrid genetic algorithm) 이라 부른다. 유전알고리듬과 local 최적화 기법은 상호 보완적 성질을 지니고 있기 때문에, 혼성유전알고리듬은 두 기법 중 어느 하나만을 사용하는 경우보다 더 뛰어난 성능을 보일 수 있다. 혼성유전알고리듬의 사용에 관한 주요 문헌으로서 Davis[5], Gen and Cheng[8], Goldberg[9], Reeves[15] 등이 있다.

본 연구에서는 고객의 서비스 납기와 상한시각이 주어진 상황에서 총차량이동시간 최소화, 총가중순수납기지연 최소화, 차량소요대수 최소화의 상충하는 세가지 목적을 동시에 달성하고자 하는 새로운 다목적차량일정문제(multiobjective vehicle scheduling problems with service due times : MVSPDT)를 설정한 다음, 혼성유전알고리듬을 이용하여 최적, 또는 최적에 근사한 차량일정을 탐색하는 해법(hybrid genetic algorithm for MVSPDT : HGAM)의 개발을 소개한다. 개발된 HGAM의 성능은 다양한 실험문제를 이용한 계산실험을 통해 평가된다. MVSPDT에서는 서비스 상한시각 이후의 차량도착은 불허하며, 납기 전에 도착한 차량은 곧바로 서비스를 시작할 수 있다. 서비스 납기 이후의 도착은 순수납기지연(tardiness)으로 간주되며, 고객에 따라 지연에 대한 가중치가 달라진다. MVSPDT에서는 서비스 상한시각 외에 차량귀환시각과 차량적재용량의 제약조건이 존재한다.

2. HGAM의 개발

HGAM은 필요할 때마다 repair 작업을 시행하여 항상 해의 feasibility를 유지하면서 진화한다(4.1 참조). 또 좋은 해가 진화과정에서 사라질 수 있는 가능성을 고려하여 elitist 전략을 적용한다. HGAM은 진화과정에서 탐색의 조기수렴을 방지하면서 우수한 인자를 가진 개체들의 생존확률을 높게 유지할 수 있도록 virtual parallel computing에서 사용하는 mixed farming and migration 전략[11]을 응용한다. 이 전략의 기본 개념은 초기 모집단을 몇 개의 소집단(sub-populations)으로 나누어 각 소집단에 대해 독립적으로 유전알고리듬을 적용하는 것이다. 그리고 진화과정에서 주기적으로 소집단들로부터 일정한 개수 씩의 가장 좋은 해를 모아서 local 최적화 기법을 적용하여 소집단 해의 일부를 이들의 개선된 해들로써 대체한다.

HGAM의 구조는 [그림 1]에 묘사되어 있으며, 진행과정은 다음과 같이 정리된다.



[그림 1] HGAM의 구조

순서 1 : 초기 모집단을 동일한 개체수의 l개 소집단으로 나눈다.

순서 2 : 각 소집단에 대해 교차변이와 돌연변이를 수행하여 개체들의 적합도를 구한다. 만일 현재의 세대수가 m의 배수이면 순서 3으로 진행한다. 아니면 순서 4로 진행한다.

순서 3 : 각 소집단으로부터 n개씩의 가장 좋은(그리고 상이한) 개체를 선정하여, 총 ln개의 개체에

대해 local 최적화 기법을 적용한다. 이 결과 얻어진 개체들 중 가장 좋은(그리고 상이한) n개의 개체들로써 각 소집단내 n개의 기존 가장 좋은(그리고 상이한) 개체들을 모두 대체한다.

순서 4 : 만일 종료조건을 만족하면, 멈춘다. 아니면, 각 소집단에 대해 roulette wheel 선별방법[15]을 적용하여 다음 세대의 개체들을 생성한다. 그리고 순서 2로 되돌아간다.

HGAM을 설계요소별로 설명한다.

(1) 해의 표현(coding)

하나의 해는 2개의 string으로 이루어진 이중구조(diploid structure)로써 표현된다. [그림 2]의 예에서 A_1은 노드번호를 그리고 A_2는 차량번호를 나타낸다. 따라서 이 예의 경우는 모두 3대의 차량이 사용되며, 각 차량은 0-3-9-1-10-7-0, 0-2-4-5-6-0, 0-8-0의 경로를 따른다. 여기서 노드번호 0은 중앙차고지를 뜻한다.

A_1	3	9	2	1	4	10	8	5	6	7
A_2	1	1	2	1	2	1	3	2	2	1

[그림 2] 해의 표현 예

(2) 초기 모집단(initial population)

먼저, 노드 수 만큼의 난수를 발생하여 A_1의 맨 왼쪽 인자부터 순서대로 노드번호를 할당한다. 다음에, 차량번호를 string의 맨 왼쪽 노드에 대해서부터 순서대로 결정하는데 serial insertion 기법[13]을 부분적으로 적용하여 초기 해를 얻는다. 즉, 대상 노드에 대해 1번 차량의 루트부터 시작하여 경로순서에 따라 첫 번째 feasible 삽입위치를 찾아 string의 해당위치에 대상 노드와 차량의 번호를 할당한 다음, 해당 루트에서 후속 노드들의 string 위치를 한 순서씩 뒤로 수정한다. 만일 기존의 모든 루트에서 feasible 삽입위치가 존재하지 않으면, 새로운 차량번호를 할당한다. string의 첫 번째 노드의 차량번호로는 항상 1을 할당한다.

(3) 적합도 평가(fitness evaluation)

세 목적이 가중치로써 결합된 단일목적 함수식을 이용하여 개체 k의 정규화된 가중목적값 $C(S_k)$를 구한다.

$$C(S_k)=\left(w_1\frac{t_k}{t^0_{\max}}+w_2\frac{d_k}{d^0_{\max}}+w_3\frac{v_k}{v^0_{\max}}\right)*100 \quad (1)$$

단, $\sum_{i=1}^{3} w_i=1$

여기서 w_1 = 총차량이동시간 최소화 목적의 가중치 $(0\le w_1\le 1)$,

w_2 = 총가중순수납기지연 최소화 목적의 가중치 $(0\le w_2\le 1)$,

w_3 = 차량소요대수 최소화 목적의 가중치 $(0\le w_3\le 1)$,

$t^0_{\max}$ = 초기 모집단내 개체 중에서 얻어지는 최대 총차량이동시간,

$d^0_{\max}$ = 초기 모집단내 개체 중에서 얻어지는 최대 총가중순수납기지연,

$v^0_{\max}$ = 초기 모집단내 개체 중에서 얻어지는 최대 차량소요대수,

t_k = 개체 k의 총차량이동시간,

d_k = 개체 k의 총가중순수납기지연,

v_k = 개체 k의 차량소요대수.

소집단에서 개체 k의 적합도는 아래의 $f(S_k)$ 식을 이용하여 구한다. $0\le f(S_k)\le 1$이며, $f(S_k)$가 클수록 해의 적합도는 높다. 해의 적합도를 나타내는데 $C(S_k)$ 대신 $f(S_k)$를 사용하는 이유는 소집단내 개체들의 우열을 명확히 하여 상대적으로 좋은 개체들이 다음 세대에 선별될 확률을 높게 하기 위해서 이다.

$$f(S_k)=\frac{C(S^k_{\max})-C(S_k)}{C(S^k_{\max})-C(S^k_{\min})} \quad (2)$$

여기서 $C(S^k_{\max})$ = 개체 k가 속한 소집단에서 얻어지는 최대 가중목적값,
$C(S^k_{\min})$ = 개체 k가 속한 소집단에서 얻어지는 최소 가중목적값.

(4) local 최적화

local 최적화 기법으로는 Dror and Levy[6]가 개발한 greedy interchange 기법을 변형 사용한다. local 최적화 과정은 두 단계로 이루어진다. 첫 번째 단계에서는 개체내 각 루트에 대해 제약조건을 만족하는 최선의 2-노드교환을 찾아 먼저 실시한 후, 이 개선된 루트에 대해 최선의 2-opt 아크교환을 찾아 실시한다. 2-opt 아크교환은 루트에서 인접하지 않은 두 개의 아크를 선택하여, 이들을 한 아크의 출발(도착) 노드와 다른 한 아크의 출발(도착) 노드를 각각 연결한 두 개의 아크와 교환하여 새롭게 루트를 구성해 봄으로써 해의 개선을 꾀하는 방법이다. 루트 해의 제약조건 만족여부를 조사할 때 서비스 상한시각 조건에 대해서 규칙 1을 적용할 수 있다. 노드나 아크의 교환에 따른 가중감소(weighted saving)의 계산은 다음 식을 이용한다.

$$WS = w_1 \frac{\nabla t}{t^0_{\max}} + w_2 \frac{\nabla d}{d^0_{\max}} \qquad (3)$$

여기서 ∇t = 2-노드교환에 따른 차량이동시간의 감소분,
∇d = 2-노드교환에 따른 가중순수납기지연의 감소분.

(규칙 1) 서비스 상한시각 제약조건 조사

두 아크 (i, j)와 (k, l)의 교환을 고려할 때(단, $j<k$), 교환전 i와 l사이의 경로에 속한 노드들의 방문시각이 교환후 서비스 상한시각을 초과하지 않고 또 차량이동시간이 감소하면, 교환후 l을 포함하여 경로상 l 이후 노드들은 서비스 상한시각을 초과하지 않는다.

두 번째 단계에서는 개체내 각 루트 쌍에 대해 모든 가능한 1-노드교환과 2-노드교환을 고려하여 제약조건을 만족하는 최선의 교환을 찾아 실시한다. 서비스 상한시각 제약조건 조사를 위해 규칙 2를 적용할 수 있다. 1-노드교환의 경우 가중감소는 식 (3)에 $w_3 \nabla v / v^0_{\max}$을 추가로 더하여 구한다. ∇v는 1-노드교환에 따른 차량의 감소분을 나타낸다.

(규칙 2) 서비스 상한시각 제약조건 조사

두 노드 i와 $j(i<<j)$가 속한 루트와 노드 k, r, $l(k<<r<<l)$이 속한 루트간에 r을 가지고 1-노드교환을 시행할 때

(i) 만일 $t_{ir} + t_{rj} - t_{ij} \le 0$면, j를 포함하여 경로상 j이후 노드들은 서비스 상한시각을 초과하지 않는다. 단, 노드 r에 대한 방문시각은 조사가 필요하다. 여기서 t_{ir}은 노드 i에서부터 r까지의 차량이동시간을 나타낸다.

(ii) 만일 $t_{ir} + t_{rj} - t_{ij} = IC > 0$면, j를 포함하여 경로상 j이후 노드들의 새로운 도착시각은 $a'_u = a_u + IC$가 된다.

(5) 교차변이(crossover)

교차변이를 위해 partially mapped crossover (PMX)[12]를 변형 사용한다. 변형된 방법은 교차변이 과정에서 다수 차량이 동시에 고려되고, 또한 해의 repair 작업이 함께 이루어지는 특징을 가진다. 변형 PMX는 자손개체(offspring)의 인자를 항상 맨 왼쪽에서부터 그리고 노드와 차량의 번호를 함께 결정한다. 그리고 swapping 과정에서 노드번호의 중복이 발생하면 정의된 mapping에 따라 노드와 차량의 번호를 정하고, 만일 infeasible 노드가 존재하면 해당 노드의 차량번호를 1번부터 차례로 바꾸면서 feasibility를 조사하여 차량번호를 수정한다.

(6) 돌연변이(mutation)

한 개체는 여러 차량들의 루트를 함께 표현하고 있기 때문에, 정상적인 돌연변이 방법은 subtour의 생성, 불필요한 차량증가, 중복된 고객 등의 현상을 야기시켜 이전의 진화과정을 거치면서 구해진 해의 질을 크게 저하시킬 수 있다. 따라서 개체의 루트정보를 이용하여 해의 개선과 다양화를 기대할 수 있는 총 5개의 돌연변이 연산자를 새로이 만들어 사용한다.

- merge mutation : 노드 수가 둘이하인 임의의 루트에서 한 노드의 차량번호를 기존의 다른 차량번호 중 하나로 변경한다. 만일 제약조건 위반으로 인해 대상 루트 내의 어떤 노드에 대해서도 차량번호의 변경이 불가능하면, merge mutation을 포기한다.
- time mutation : 차량이동시간이 가장 긴 루트에서 가장 긴 이동시간 경로(path)의 도착노드 차량번호를 차량이동시간이 가장 짧은 루트의 차량번호로 변경한다. 이로 인해 해당 루트의 제약조건이 위반되면 이 노드의 차량번호를 차량이동시간이 다음으로 짧은 루트의 차량번호로 바꾸어 보는 시도를 feasibility 조사를 만족할 때까지 반복한다. 만일 기존의 어떤 차량번호로도 변경이 불가능하면, time mutation을 포기한다.
- tardy mutation : 총가중순수납기지연이 가장 큰 루트에서 가중납기지연이 가장 큰 노드의 차량번호를 총가중순수납기지연이 가장 작은 루트의 차량번호로 변경한다. 이로 인해 해당 루트의 제약조건이 위반되면 이 노드의 차량번호를 총가중순수납기지연이 다음으로 작은 루트의 차량번호로 바꾸어 보는 시도를 feasibility 조사를 만족할 때까지 반복한다. 만일 기존의 어떤 차량번호로도 변경이 불가능하면, tardy mutation을 포기한다.
- random mutation : 개체 내에서 임의의 한 노드의 차량번호를 기존의 다른 차량번호 중 하나로 변경한다. 만일 제약조건 위반으로 인해 기존의 어떤 차량번호로도 변경이 불가능하면, 새로운 차량번호를 할당한다.
- exchange mutation : 개체 내에서 임의로 두 노드를 선택하여 차량번호와 함께 서로의 위치를 맞바꾼다. 만일 제약조건 위반으로 인해 어떠한 노드위치 교환도 불가능하면, exchange mutation을 포기한다.

위의 돌연변이 연산자중 merge mutation, time mutation, tardy mutation은 각각 차량소요대수, 총차량이동시간, 총가중순수납기지연의 개선을 시도한다. 세 목적의 중요도와 돌연변이 대상 개체의 특성에 따라 특정 돌연변이를 선택 적용하여 해의 개선을 꾀할 수 있다. 이것은 유전알고리듬에 gradient descent method를 묵시적으로 결합하는 것과 같다. 그리고 random mutation과 exchange mutation은 해 탐색의 다양화를 제공한다.

돌연변이 대상이 되는 개체에 대해 하나의 돌연변이 연산자를 선택하는 과정에서 연산자를 고려하는 순서는 세 목적의 중요도에 따라 결정된다. 이것은 각각의 연산자가 추구하는 개선목적이 다르기 때문이다. 아래의 선택과정은 차량소요대수, 총차량이동시간, 총가중순수납기지연의 최소화 순서로 중요도가 주어진 경우를 가정한 것이다. 순서 2와 3에서 최소값의 배수는 조정할 수 있다.

순서 1 : 개체에서 고객수가 둘 이하인 루트가 존재하면, merge mutation을 선택한다. 대상 루트가 존재하지 않거나 merge mutation의 적용이 불가능하면, 순서 2로 이동한다.

순서 2 : 루트들의 차량이동시간 최대값이 최소값의 1.3배 이상이면, time mutation을 선택한다. 배수를 만족하는 루트가 존재하지 않거나 time mutation의 적용이 불가능하면, 순서 3으로 이동한다.

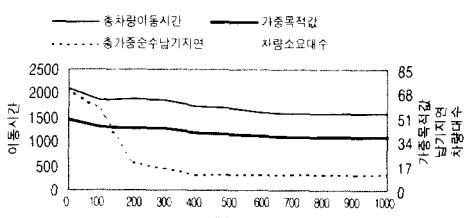
순서 3 : 루트들의 총가중순수납기지연 최대값이 최소값의 1.3배 이상이면, tardy mutation을 선택한다. 배수를 만족하는 루트가 존재하지 않거나 tardy mutation의 적용이 불가능하면, 순서 4로 이동한다.

순서 4 : random mutation과 exchange mutation중 하나를 번갈아 선택한다.

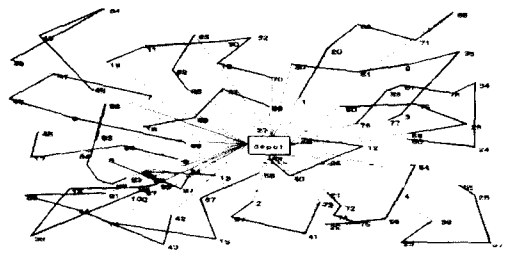
3. 예 제

Solomon[16]의 실험문제 중에서 R101 문제를 MVSPDT에 맞게 보완하여 HGAM을 적용하였다. R101 문제에서 고객들의 x와 y좌표 값은 일양분포에 의해 랜덤하게 결정되어 있다. R101 문제의 특성은 <표 1>을 참고하시오. 세 목적의 가중치는 총차량이동시간을 0.6, 총가중순수납기지연을 0.3, 그리고 차량소요대수를 0.1로 가정하였다. HGAM의 적용에서 초기 모집단의 개체수는 90개, 소집단수는 3개, 교차변이율은 0.4, 돌연변이율은 0.4, local 최적화 주기는 5세대, 각 소집단에서 local 최적화 대상 개체수는 3개로 설정하였다.

예제를 풀기 위해 HGAM을 Visual Basic 5.0으로 프로그래밍 하였으며, 이를 IBM PC 호환 586 (32M RAM, Intel MMX 233)에서 실행하였다. 1000세대동안 예제에 HGAM을 적용하면서 얻어진 최선 해의 세 목적값과 가중목적값의 변화가 [그림 3]에 나타나 있다. 그림에서 세대수가 증가함에 따라 해가 점진적으로 개선되어 지는 것을 확인할 수 있다. 그러나 600세대에서부터 개선 효과는 크게 둔화되어 가중목적값은 거의 변화가 없다. 1000세대 진화 후 구해진 최선 해의 총차량이동시간은 1586.4, 총가중순수납기지연은 11.7, 차량소요대수는 21, 그리고 가중목적값은 37.4로 계산되었다. 최선 해의 차량경로는 [그림 4]에 나타나 있다. 1000세대 진화까지의 컴퓨터 실행시간은 약 30분 정도 소요되었다.



[그림 3] 예제에서 세대수 증가에 따른 세 목적값과 가중목적값의 변화



[그림 4] 예제에서 1000세대 진화 후 구해진 차량경로 해

4. 계산실험

개발된 HGAM의 성능을 평가하기 위해 3가지의 실험을 수행하였다. 첫째, infeasible 해를 포함하는 진화방법과의 성능비교; 둘째, 유전파라미터(genetic parameters) 값들의 변화에 대한 민감도 분석; 셋째, BC-saving 기법[2]과의 성능비교. 계산실험을 위해 HGAM과 BC-saving 기법을 Visual Basic 5.0으로써 프로그래밍 하였으며, 모든 실험을 IBM PC 호환 586(32M RAM, Intel MMX 233)에서 수행하였다.

4.1 HGAM과 infeasible 해를 포함하는 진화방법과의 성능비교

MVSPDT에 대해 HGAM이 사용하고 있는 항상 해의 feasibility를 유지하면서 진화하는 방법과 infeasible 해의 제약조건 위반에 대해서 벌칙을 부과하면서 해의 영역을 탐색하는 방법의 수행도를 비교한다. infeasible 해를 포함하는 경우 개체의 가중목적값은 아래 식에 의해 계산된다.

$$C(S_k) = (w_1 \frac{t_k}{t^0_{\max}} + w_2 \frac{d_k}{d^0_{\max}} + w_3 \frac{v_k}{v^0_{\max}} + \rho_1 l_k + \rho_2 r_k + \rho_3 q_k) * 100 \quad (4)$$

여기서 ρ_1 = 노드의 서비스 상한시각 제약조건 벌칙율 ($\rho_1 \geq 0$),

ρ_2 = 루트의 차고지 귀환시각 제약조건 벌칙율 ($\rho_2 \geq 0$),

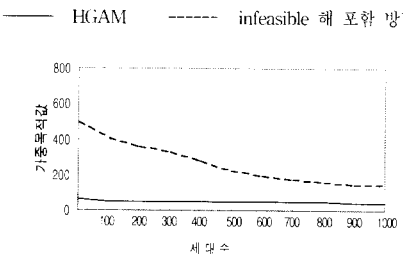
ρ_3 = 루트의 차량적재용량 제약조건 벌칙율 ($\rho_3 \geq 0$),

l_k = 개체 k에서 서비스 상한시각을 초과한 노드수,

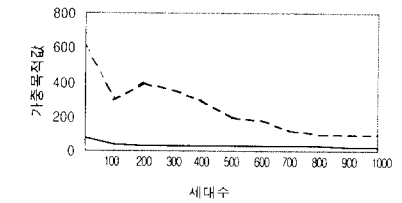
r_k = 개체 k에서 차고지 귀환시각을 초과한 루트수,

q_k = 개체 k에서 차량적재용량을 초과한 루트수.

실험을 위해 Solomon[16]의 R101과 RC201 문제를 사용하였다. 두 방법에서 w_1=0.8, w_2=0.1, w_3=0.1 그리고 ρ_1=1, ρ_2=1, ρ_3=1로 가정하였다. 세대수 증가에 따른 두 방법의 수행도 변화가 [그림 5]에 정리되어 있다. 두 실험문제에서 두 방법 모두 다 세대수의 증가에 따라 해의 개선이 이루어지고 있으나, HGAM이 더 빨리 그리고 더 좋은 해에 수렴하는 것을 알 수 있다.



(a) R101 문제의 경우



(b) RC201 문제의 경우

[그림 5] HGAM과 infeasible 해를 포함하는 진화방법과의 성능비교

이러한 결과는 infeasible 해를 포함하는 경우 혼성유전알고리듬이 진화과정을 거치면서 최적 또는 최적에 근사한 해를 탐색하기보다는 계속해서 생성되는 많은 infeasible 해들 속에서 우성개체로서 feasible 해를 찾아내는데 주로 작용하기 때문인 것으로 판단된다. 다시 말해서, 혼성유전알고리듬의 적용이 진화과정에서 지속적으로 발생하는 infeasible 해들의 제약조건 위반에 대한 개선작업에 주로 작용하면서 좋은 feasible 해를 탐색하는 데는 더디게 영향을 미친다는 것이다.

4.2 HGAM의 유전파라미터 값 변화에 대한 민감도 분석

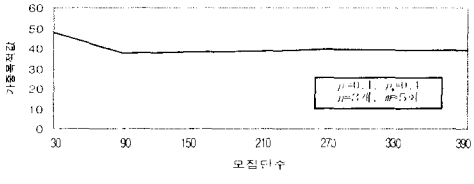
HGAM의 적용에서 유전파라미터들의 결정은 필요하다. 주요 유전파라미터로는 진행 세대수(GEN), 초기 모집단수(pop_size), 소집단수(l), 교차변이율(p_c), 돌연변이율(p_m), 각 소집단에서 local 최적화 대상 개체수(n), local 최적화 주기(m)를 포함한다. 유전파라미터 값은 일반적으로 유전알고리듬의 수행도(탐색시간, 해의 최적성 등)에 영향을 미치는 것으로 알려져 있지만, 최적 값을 구하는 분석적 방법은 아직 개발되어 있지 않다. 유전파라미터의 최적 값이 대상문제에 따라 항상 달라지기 때문이다.

HGAM의 적용에서 유전파라미터 값의 변화에 따라 구해진 최선 해의 가중목적값이 어떻게 달라지는 지를 분석한다. 실험에서 유전파라미터들의 교호작용에 의한 영향은 모두 무시하고 단순히 한 유전파라미터 값의 변화에 따른 가중목적값의 영향을 분석하였기 때문에, 실험결과에 절대적인 의미를 부여할 수 없다는 것을 지적해 둔다. 총 4가지의 유전파라미터에 대해 실험을 수행하였으며, 이 실험결과는 4.3의 성능비교 실험에서 HGAM의 최적 유전파라미터 값 설정에 반영하였다. 모든 실험에서 GEN=1000회, 그리고 w_1=0.6, w_2=0.3, w_3=0.1로 가정하였다. 실험을 위해 Solomon[16]의 R101과 RC201 문제를 사용하였으며, 두 문제에 대한 실

험결과는 유사하게 나타났다. 여기서는 R101 문제에 대한 실험결과 만을 소개하기로 한다.

(1) 모집단 크기의 변화

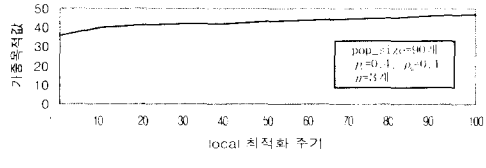
모집단수를 30~390개 사이에서 60개씩 변화하면서 실험문제에 HGAM을 적용하였다. [그림 6]에 나타난 바와 같이, 가중목적값이 모집단수 90개까지는 크게 줄어드나 그 이후 부터는 거의 변화없이 40에 가까운 값을 유지하는 것을 알 수 있다. 실험결과, 모집단수의 증가에 따라 HGAM은 항상 더 좋은 해를 구한다고 말할 수 없다.



[그림 6] 모집단 크기에 따른 가중목적값의 변화(R101 문제의 경우)

(2) local 최적화 주기의 변화

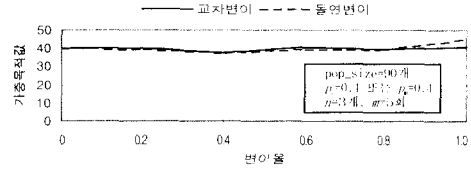
local 최적화 주기를 1~100회 사이에서 10회씩 변화하면서 실험문제에 HGAM을 적용하였다. [그림 7]에 보여진 바와 같이, local 최적화 주기가 짧을수록 해의 개선효과가 크다는 것을 알 수 있다. 그러나 HGAM의 계산소요시간은 local 최적화 주기가 1/2씩 짧아짐(즉, local 최적화 적용회수가 2배로 늘어남)에 따라 대략 1.7~2.5배씩 증가하였다. 이것은 해의 질과 계산시간간에 trade-off가 발생하고 있음을 의미한다.



[그림 7] local 최적화 주기에 따른 가중목적값의 변화(R101 문제의 경우)

(3) 교차변이율과 돌연변이율의 변화

교차변이율과 돌연변이율을 각각 0.0과 1.0사이에서 0.2씩 변화하면서 실험문제에 HGAM을 적용하였다. [그림 8]에 나타난 바와 같이, 구해진 최선해의 가중목적값이 변이율의 변화에 따라 큰 차이를 보이지는 않는다. 이 실험문제에서는 교차변이율이 0.4일 때 가장 좋은 해가 구해진다. 이것은 교차변이율이 너무 작으면 교차변이에 의해 우성인자를 다음 세대에 상속할 수 있는 가능성이 줄어들고, 반대로 교차변이율이 너무 크면 교차변이의 역효과가 발생하기 때문이다. 돌연변이의 경우, 돌연변이율이 0.4일 때까지는 작으나마 해의 개선효과가 있으나 돌연변이율이 0.4이상으로 증가하면 해가 점점 나빠지는 것을 알 수 있다. 이것은 지나치게 많은 돌연변이는 해 탐색의 영역을 크게 넓혀주는 효과가 있지만 좋은 해로 수렴해 가는 과정에 부정적 영향을 미칠 수 있기 때문이다.



[그림 8] 교차변이율과 돌연변이율에 따른 가중목적값의 변화(R101 문제의 경우)

4.3 HGAM과 BC-saving 기법과의 성능비교

MVSPDT를 다루는 아주 최근의 해법으로서 BC-saving 기법[2]이 있다. BC-saving 기법은 한 루트의 마지막 노드 i와 다른 루트의 첫 번째 노드 j를 연결함으로써 기대되는 차량이동시간의 saving (ST_{ij})과 가중순수납기지연의 saving(SD_{ij})의 합이 가장 큰 두 루트를 결합하면서 배차계획을 수립해 가는 발견적 기법이며, 그 성능이 매우 뛰어나다. 원래 BC-saving 기법은 차량속도가 변화하는 경우의 문제를 위해 개발되었지만 두 노드간 이동시간이 항상 일정한 경우에도 그대로 적용될 수 있

다. 본 실험의 BC-saving 기법 적용에서는 차량대수의 saving을 추가하여 정규화된 가중 saving을 사용하였으며, 그 계산식은 다음과 같다.

$$BC_{ij} = w_1 \frac{ST_{ij}}{t^o_{\max}} + w_2 \frac{SD_{ij}}{d^o_{\max}} + w_3 \frac{1}{v^o_{\max}} \quad (5)$$

위 식의 부호설명은 식 (1)에서와 동일하다.

두 해법의 성능비교 실험을 위해 Solomon[16]의 R1, R2, RC1, RC2 문제 유형별로 각각 5문제씩을 선정하여 총 20개의 MVSPDT를 구축하였다. R1과 R2유형의 문제에서는 노드들의 좌표 값이 일양분포에 의해 랜덤하게 결정되어 있으며, RC1과 RC2 유형의 문제에서는 노드들의 위치가 일양분포와 cluster를 혼합한 형태로 결정되어 있다. 각 문제에서 서비스 상한시각은 지점간 평균이동시간의 크기를 고려하여 납기에 적당한 수치를 더하여 정하였다. 20개의 실험문제가 <표 1>에 정리되어 있다.

4.2의 실험결과를 토대로HGAM의 적용에서 GEN=1000회, pop_size=90개, p_c=0.4, p_m=0.4, n=3개, m=5회로 설정하였다. 모든 실험문제에서 세 목적의 중요도는 3가지 경우를 가정하였다. 즉, (w_1, w_2, w_3) = (0.6, 0.3, 0.1), (0.3, 0.6, 0.1), (0.3, 0.1, 0.6). 두 해법의 성능비교 실험결과가 <표 2>에 정리되어 있다. 이 표에서 감소율(%)은 (BC-saving 기법에 의한 가중목적값 HGAM에 의한 가중목적값)*100/BC-saving 기법에 의한 가중목적값 식에 의해 계산되었다. HGAM에 의한 가중목적값의 감소율이 <표 3>에 요약 정리되어 있다.

모든 경우의 문제에서 HGAM이 BC-saving 기법보다 항상 더 나은 차량일정 해를 구해주는 것을 <표 2>와 <표 3>으로부터 알 수 있다. <표 3>에 의하면, 가중목적값의 감소율이 전체적으로 평균 8.0%로 계산된다. HGAM에 의해 얻어진 차량일정 해는 BC-saving 기법에 의한 해와 비교하여 가중목적값 뿐만 아니라 세 목적 모두에 대해서도 거의

〈표 1〉 실험문제의 요약

유형	문제	고객수	차량용량	차고지 귀환시각상한	서비스 상한시각	서비스시간
R1	R101	100	200	230	납기+10	10
	R102	100	200	230	납기+10	10
	R103	100	200	230	납기+10	10
	R105	100	200	230	납기+30	10
	R106	100	200	230	납기+30	10
R2	R201	100	1000	1000	납기+116(평균)	10
	R202	100	1000	1000	납기+57(평균)	10
	R203	100	1000	1000	납기+103(평균)	10
	R205	100	1000	1000	납기+240	10
	R206	100	1000	1000	납기+240	10
RC1	RC101	100	200	240	납기+30	10
	RC102	100	200	240	납기+30	10
	RC103	100	200	240	납기+30	10
	RC105	100	200	240	납기+54(평균)	10
	RC106	100	200	240	납기+60	10
RC2	RC201	100	1000	960	납기+120	10
	RC202	100	1000	960	납기+120	10
	RC203	100	1000	960	납기+240	10
	RC205	100	1000	960	납기+223(평균)	10
	RC206	100	1000	960	납기+ 240	10

〈표 2〉 HGAM과 BC-saving 기법의 성능비교

실험문제		(0.6, 0.3, 0.1)[a]					(0.3, 0.6, 0.1)					(0.3, 0.1, 0.6)				
		HGAM		BC-saving		감소율(%)	HGAM		BC-saving		감소율(%)	HGAM		BC-saving		감소율(%)
		가중목적값	세 목적값	가중목적값	세 목적값		가중목적값	세 목적값	가중목적값	세 목적값		가중목적값	세 목적값	가중목적값	세 목적값	
R1	R101	37.4	(1586, 12, 21)[b]	39.0	(1642, 19, 23)	5.2	22.8	(1634, 3, 21)	24.6	(1695, 5, 24)	7.3	56.7	(1580, 40, 20)	61.3	(1622, 45, 22)	7.5
	R102	42.0	(1527, 906, 19)	44.7	(1578, 1025, 21)	6.1	33.5	(1712, 599, 24)	35.3	(1758, 646, 26)	5.6	56.6	(1503, 1089, 18)	61.2	(1515, 1118, 20)	7.6
	R103	43.5	(1420, 4480, 18)	46.0	(1478, 4956, 19)	5.5	39.5	(1659, 3557, 21)	42.1	(1669, 3942, 23)	6.1	59.7	(1399, 6020, 15)	65.8	(1406, 6497, 17)	9.3
	R105	37.7	(1511, 145, 19)	39.9	(1553, 189, 21)	5.4	25.2	(1726, 18, 22)	27.7	(1790, 31, 26)	8.9	60.0	(1598, 250, 19)	66.6	(1550, 282, 22)	9.9
	R106	39.8	(1414, 1911, 17)	42.9	(1471, 2326, 19)	7.3	32.6	(1667, 1302, 22)	34.8	(1671, 1542, 24)	6.4	56.0	(1401, 2597, 16)	61.3	(1415, 2785, 18)	8.7
R2	R201	27.1	(978, 1158, 8)	29.2	(1023, 1319, 9)	7.2	19.6	(1027, 554, 9)	20.3	(1088, 582, 9)	3.4	40.1	(960, 1750, 6)	50.2	(969, 1769, 8)	20.2
	R202	27.2	(912, 4633, 7)	29.3	(951, 5007, 8)	7.1	15.7	(658, 3011, 5)	16.6	(894, 6260, 7)	5.6	42.6	(889, 5897, 5)	56.2	(910, 6349, 7)	24.2
	R203	28.5	(929, 11458, 6)	31.2	(973, 12919, 7)	8.5	26.2	(1058, 9674, 7)	28.0	(1077, 10063, 8)	6.3	46.5	(870, 10112, 5)	55.1	(909, 15169, 6)	15.5
	R205	27.1	(941, 2122, 7)	29.0	(979, 2228, 8)	6.2	21.8	(1014, 1075, 9)	23.1	(1058, 1277, 10)	7.3	37.7	(894, 7985, 4)	44.4	(911, 7670, 5)	15.2
	R206	27.2	(874, 5721, 7)	28.5	(929, 6373, 7)	4.7	23.0	(1041, 4399, 7)	25.0	(1051, 4959, 8)	8.7	40.3	(823, 15987, 4)	54.1	(851, 9172, 6)	25.6
RC1	RC101	40.1	(1875, 314, 20)	42.0	(1924, 352, 22)	4.6	25.5	(2093, 17, 25)	27.2	(2161, 49, 26)	6.2	56.8	(1865, 398, 19)	61.3	(1878, 422, 21)	7.3
	RC102	42.6	(1710, 3478, 19)	45.5	(1772, 3796, 19)	6.4	36.1	(2109, 2131, 23)	38.5	(2096, 2518, 24)	6.3	51.1	(1690, 3216, 17)	56.1	(1727, 3787, 19)	5.5
	RC103	42.2	(1604, 6049, 15)	44.8	(1648, 6595, 17)	5.9	42.1	(1877, 4627, 22)	43.6	(1897, 4950, 22)	3.4	53.9	(1574, 8745, 14)	58.3	(1590, 7823, 16)	8.0
	RC105	38.2	(1645, 991, 18)	39.4	(1677, 1033, 19)	2.9	28.0	(2068, 203, 25)	29.7	(2148, 262, 26)	5.8	53.7	(1622, 1022, 17)	53.9	(1637, 1048, 17)	0.3
	RC106	35.1	(1588, 1024, 17)	36.4	(1614, 1122, 18)	3.6	26.0	(1985, 207, 25)	27.7	(2093, 322, 24)	6.1	49.1	(1522, 1346, 16)	49.6	(1540, 1547, 16)	1.0
RC2	RC201	20.7	(945, 573, 7)	22.8	(1009, 692, 8)	9.2	16.0	(1079, 156, 9)	17.7	(1188, 175, 10)	9.7	39.1	(927, 2487, 6)	43.2	(1029, 1014, 7)	9.5
	RC202	22.7	(915, 6145, 7)	24.8	(982, 6569, 8)	8.5	22.1	(1146, 4117, 10)	23.5	(1208, 4834, 10)	5.7	40.6	(880, 8873, 6)	40.9	(891, 9975, 6)	0.8
	RC203	26.0	(934, 13322, 7)	27.9	(1043, 14328, 7)	6.3	25.8	(1188, 7856, 10)	27.6	(1229, 8344, 11)	6.6	41.2	(900, 12542, 5)	55.3	(969, 14834, 7)	25.5
	RC205	21.2	(899, 2589, 5)	22.5	(923, 2626, 6)	5.8	17.1	(1037, 821, 8)	18.1	(1042, 832, 9)	5.4	37.3	(1015, 5678, 5)	40.8	(905, 3177, 6)	8.4
	RC206	20.0	(935, 1504, 6)	22.0	(964, 1659, 7)	6.9	16.2	(1011, 1048, 7)	17.7	(1027, 1176, 8)	7.9	35.1	(921, 3156, 5)	40.7	(914, 3615, 6)	13.8

(주) a : (총차량이동시간의 가중치, 총가중순수납기지연의 가중치, 차량소요대수의 가중치)
b : (총차량이동시간, 총가중순수납기지연, 차량소요대수)

〈표 3〉 HGAM에 의한 가중목적값의 감소율(%) 요약

가중치 : (w_1, w_2, w_3)	R1 유형	R2 유형	RC1 유형	RC2 유형	평균
(0.6, 0.3, 0.1)	5.9	6.7	4.7	7.3	6.2
(0.3, 0.6, 0.1)	6.9	6.3	5.6	7.0	6.4
(0.3, 0.1, 0.6)	8.6	20.1	5.1	11.6	11.4
평균	7.1	11.0	5.1	8.6	8.0

대부분 더 좋은 결과를 보여준다. 특히, R2유형의 문제에서 세 목적의 가중치를 (0.3, 0.1, 0.6)으로 가정한 경우에 HGAM은 BC-saving 기법과 비교하여 그 성능이 아주 뛰어남을 알 수 있다. 이러한 현상은 R2유형의 문제가 특성상 다른 문제들과 비교하여 상대적으로 적은 수의 차량을 필요로 하여 HGAM에 의한 차량대수의 감소가 높은 가중치(0.6)에 의해 가중목적값에 크게 영향을 미치기 때문인 것으로 판단된다.

또한 <표 2>로부터, 세 목적의 가중치 크기 변화에 따라 두 해법 모두에서 목적값들이 적절하게 trade-off되고 있음을 확인할 수 있다. 예컨데, <표 2>에 정리되어 있는 모든 실험문제에서 총차량이동시간의 가중치가 0.6으로 가장 큰 경우의 총차량이동시간은 총가중순수납 기지연과 차량소요대수의 증가에 의해 세 가지 가중치 조합중에서 가장 작은 값을 가진다.

HGAM의 컴퓨터 계산시간은 분단위로 그리고 BC-saving 기법은 초단위로 소요되어 두 해법에 대한 계산시간의 상세한 비교는 생략하였다. 결국, HGAM의 적용에 의해 기대되는 좋은 차량일정 해는 비싼 계산비용의 결과라고 말할 수 있다. 추후, HGAM의 계산시간 단축을 위한 지속적인 연구가 필요하다.

5. 결 론

고객의 서비스 납기와 상한시각이 주어진 상황에서 총차량이동시간 최소화, 총가중순수납기지연 최소화, 차량소요대수 최소화의 세가지 상충하는 목적을 함께 달성하고자 하는 다목적차량일정문제에 대해 local 최적화 기법을 결합한 혼성유전알고리듬을 이용하여 최적, 또는 최적에 근사한 차량일정을 구하는 해법(HGAM)을 개발하였다. HGAM은 해를 2개의 string으로 이루어진 이중구조로써 표현하며, 항상 해의 feasibility를 유지하면서 진화한다. 그리고 HGAM은 진화과정에서 탐색의 조기수렴을 방지하고 우수인자를 가진 개체들의 생존확률을 높게 유지하도록 mixed farming and migration 전략을 응용하며, local 최적화를 위해 greedy interchange 기법을 변형 사용한다. HGAM에서 교차변이는 변형된 PMX에 의해 그리고 돌연변이는 새로이 만들어진 5개의 연산자에 의해 이루어진다.

개발된 HGAM의 성능을 평가하기 위해 3가지의 실험을 수행하였다. 맨 먼저 infeasible 해를 포함하는 진화방법과 성능을 비교해 본 결과, HGAM이 더 빨리 그리고 더 좋은 해에 수렴하는 것으로 나타났다. 다음에는 HGAM의 유전파라미터 값 변화에 대한 민감도 분석을 실시하였다. 분석결과, 모집단수의 증가에 따라 HGAM은 항상 더 좋은 해를 구하지 못하였으며, local 최적화 주기가 짧을수록 해의 개선효과가 컸으며, 그리고 교차변이율과 돌연변이율의 변화에 따라 HGAM의 해는 큰 차이를 보이지는 않았지만 두 변이율은 대략 0.3~0.5가 적절한 것으로 나타났다.

마지막으로, HGAM과 BC-saving 기법과의 성능비교 실험결과, 모든 경우의 문제에서 HGAM이 BC-saving 기법보다 더 나은 차량일정 해를 구해주었다. 가중목적값의 감소율은 전체평균 약 8.0%로 계산되었다. 특히, HGAM은 R2유형의 문제에서 총차량이동시간, 총가중순수납기지연, 차량소요대수 목적의 가중치를 각각 0.3, 0.1, 0.6으로 가정한 경우에 그 성능이 아주 뛰어났다. 또한 세 목적의 가중치 크기 변화에 따라 두 해법 모두에서 목적값들이 적절하게 trade-off되고 있음을 확인할 수 있었다. HGAM의 컴퓨터 계산시간은 분단위로 그리고 BC-saving 기법은 초단위로 소요되었다. 결국, HGAM의 적용에 의해 기대되는 좋은 차량일정 해는 비싼 계산비용의 결과라고 말할 수 있

다. 추후, HGAM의 계산시간 단축을 위한 지속적인 연구가 필요하다.

참 고 문 헌

[1] 김여근, 윤복식, 이상복, 메타휴리스틱, 영지문화사, 1997.

[2] 박양병, "시간대 및 구역의존 차량이동 속도를 고려하는 다목적 차량일정문제 : 일정계획 해법과 전문가시스템", 대한산업공학회지, Vol. 23, No.4(1997), pp.621-633.

[3] Blanton, J. L., and R. L. Wainwright, "Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms," *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Los Altos, CA, pp. 452-459(1993).

[4] Cheng, R. and M. Gen, "Fuzzy Vehicle Routing and Scheduling Problem Using Genetic Algorithms," in Herrera, F. and J. Verdegay, editors, *Genetic Algorithms and Soft Computing*, pp.683-709, Springer-Nerlag (1996).

[5] Davis, L., "Job Shop Scheduling with Genetic Algorithms," *Proceedings of the 1st international Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, pp.136-140 (1985).

[6] Dror, M. and L. Levy, "A Vehicle Routing Improvement Algorithm Comparison of A "GREEDY" and A Matching Implementation for Inventory Routing," *Computers and Operations Research*, Vol.13, No.1(1986), pp.33-45.

[7] Gabbert, P., D. Brown, C. Huntley, B. Markowicz, and D. Sappington, "A System for Learning Routes and Schedules with Genetic Algorithms," *Proceedings of 4th International Conference on Genetic Algorithms*, pp.430-436(1991).

[8] Gen, M. and R. Cheng, *Genetic Algorithms & Engineering Design*, John Wiley & Sons, Inc.(1997).

[9] Goldberg, D. E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, MA(1989).

[10] Malmborg, C., "A Genetic Algorithm for Service Level Based Vehicle Scheduling," *European Journal of Operational Research*, Vol.93(1996), pp.121-134.

[11] Marin, F., O. Trelles-Salazar, and F. Sandoval, "Genetic Algorithm on LAN-Based Message Passing Architectures using PVM : Application to the Routing Problem," *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, Israel, pp. 534-543(1994).

[12] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd Edition, Springer-Verlag(1994).

[13] Mole, R. H. and S. R. Jameson, "Sequential Route Building Algorithm Employing a Generalized Savings Criterion," *Operational Research Quarterly*, Vol.27(1976), pp.503-511.

[14] Potvin, J., D. Dube, and C. Robillard, "A Hybrid Approach to Vehicle Routing Using Neural Networks and Genetic Algorithms," *Applied Intelligence*, Vol.6(1996), pp.241-252.

[15] Reeves, C., "Genetic Algorithms and Neighborhood Search," *Evolutionary Computing*, pp.115-130, Springer-Verlag, Berlin(1994).

[16] Solomon, M. M., "Survey Paper : Time Window Constrained Routing and Scheduling Problems," *Transportation Science*, Vol.22,

No.1(1988), pp.1-13.

[17] Thangiah, S. R., "An Adaptive Clustering Method using Geometric Shape for Vehicle Routing with Time Windows," *Proceedings of the 6th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Los Altos, CA, pp.536-543(1995).

[18] Thangiah, S. R., Vinayagamoorthy R., and Gubbi, A. V., "Vehicle Routing with Time Deadlines using Genetic and Local Algorithms," *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Los Altos, CA, pp.506-513(1993).