물류시스템에서 운송비를 줄이기 위한 차량 할당 및 경로 설정 알고리즘 개발 및 성능평가

Development and Performance Evaluation of a Car Assignment and Routing Algorithm for Reducing Transportation Cost in a Logistics System

조병헌*, 정성훈**, 김기형***, 오하령*, 성영락*
Byeong Heon Cho, Sung Hoon Jung, Ki Hyung Kim, Ha Ryoung Oh, Yeong Rak Seong

Abstract

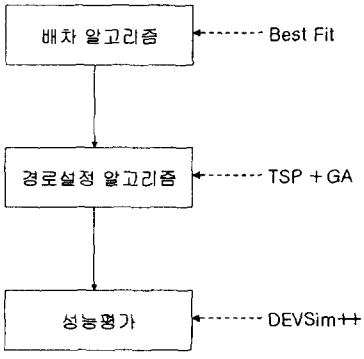
This paper proposes an algorithm which reduces transportation cost while goods are delivered in time in a logistics system. The logistics system assumed in this paper is the system in which multiple cars moves various goods from spatially distributed warehouses to stores. For reducing transportation cost, the car assignment algorithm which allocates goods to minimal cars employs the BF method; routing of each car is modelled as the TSP and is solved by using the genetic algorithm. For evaluating the proposed algorithm, the logistics system is modelled and simulated by using the DEVS formalism. The DEVS formalism specifies discrete event systems in a hierarchical, modular manner. During simulation, each car is modelled as a message and traverses warehouses and stores. When a car arrives at a warehouse or a store, predetermined amount of goods are loaded or unloaded. The arrival time and departure time of cars are analyzed and eventually whether goods are delivered in the desired time bound is verified.

* 국민대학교 전자공학부
** 한성대학교 정보전산학부
*** 영남대학교 컴퓨터정보통신공학부

1. 서론

최근 들어 경제 환경의 경쟁 압력과 자원의 제한은 물류 관리를 전략적인 수준으로 올려놓았다[1]. 이는 기업활동의 일부로서 기업 물류 활동의 중요성이 급속히 부각되기 때문이다. 이에 따라 최근 많은 연구들은 물류의 전략 개발 및 모델링을 목표로 하고 있고, 이 연구들은 물류의 전략, 구조 및 성능 등을 포함한다[2].

물류 시스템이란 공급지, 중간지점(창고), 소비자 사이에서의 제품의 흐름 경로를 네트워크로 표현한 것으로서, 그 형태는 유통 단계와 제품 흐름의 특성에 따라 다양한 형태로 나타날 수 있다. 따라서 물류 시스템의 형태는 기업에 따라 다르며, 특정기업의 물류 시스템은 하나 이상의 형태를 가질 수 있다.



<그림 1> 논문 내용의 순서도

본 논문에서는 물류 시스템에서 운송 비용을 줄이면서 주어진 시간내에 상품들을 운송하기 위한 알고리즘을 개발한다. 그리고 시뮬레이션을 통해 제안된 알고리즘의 성능을 검증한다. <그림 1>은 본 논문의 내용을 순서도의 형태로 나타낸 것이다. 본 논문에서 다루는 물류 시스템은 다수의 차량을 이용하여 다수의 창고에서 다종의 상품들을 다수의 판매처로 운송하는 시스템이다. 상품을 차량에 적재하는 배차 알고리즘은 BF(Best Fit) 방법을 사용하였고, 각 차량의 경로는 TSP(Traveling Salesman Problem)로 모델링하여 유전자 알고리즘을 이용하여 해결하였다.

제안된 알고리즘의 성능 측정과 분석을 위하여 DEVS 형식론을 이용하여 물류 시스템을 모델링하고 시뮬레이션 하였다. DEVS 형식론은 이산 사건 시스템을 계층적이고 모듈화된 형태로 기술하는 수학적인 언어이다[3]. 각각의 창고에서는 판매처에서 요구한 상품들을 적재하여 판매처에서는 원하는 상품들을 하차하고 정해진 시간 내에 배달되는지를 검증했다. 물류 시스템은 크게 입력데이터를 발생시키고 출력데이터를 분석하는 실험장치 *EF* 모델과, 판매처들의 집합 *SSET* 모델, 창고들의 집합 *WSET* 모델, 창고들과 판매처들 사이의 도로망을 나타내는 *CONNECT* 모델들로 구성된다. 각각의 모델들은 *GENR, TRANSD* 등의 원소 모델들로 계층적으로 분할된다.

모델링된 시스템을 시뮬레이션하기 위해 DEVSim++를 이용했다. DEVSim++는 DEVS 형식론을 C++ 언어로 표현한 것이다[4][5]. 구현된 시뮬레이터는 정해진 스케쥴을 가진 차량을 메시지로 전달하여 창고에 도착하면 정해진 상품들을 정해진 양만큼 적재하고, 판매처에 도착하면 정해진 상품들을 정해진 양만큼 하차한다. 그리고 도착시간, 출발시간 등을 분석하고, 최종적으로 원하는 시간 내에 배달되었는지 검증한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 이론적인 기초가 되는 DEVS 형식론과 유전자 알고리즘에 대해서 간략하게 살펴본다. 3장에서는 본 논문에서 제안하는 물류 배차 알고리즘을 설명하고 4장에서는 DEVS 형식론을 이용하여 물류 시스템을 모델링하였다. 5장에서는 3장에서 제안된 알고리즘을 시뮬레이터에 적용하여 시뮬레이션하고 결과를 분석한다. 6장은 본 논문의 결론이다.

2. 관련 연구

이 장에서는 물류 시스템의 배차 시뮬레이터를 구현하기 위하여 필요한 DEVS 형식론과 유전자 알고리즘에 대하여 알아보겠다.

2.1 DEVS 형식론

Zeigler에 의해서 제안된 DEVS(Discrete Event System Specification) 형식론은 이산사건 시스템을 기술하는 언어이다[3]. DEVS 형식론은 계층적이고 모듈화한 방법으로 이산사건 시스템의 모델들을 기술한다. 즉, DEVS 형식론은 시스템을 작은 구성요소들로 나누어 모듈화된 형태로 모델링하고 그것들을 계층적으로 구성한다. 이때 각각의 구성요소들은 원소 모델로 표현되며 계층적인 구성은 결합 모델로 나타낸다.

원소 모델은 다음과 같이 정의된다[5].

$AM = < X, S, Y, \delta_{ext}, \delta_{int}, \lambda, ta >$

여기서,

X : 입력 사건의 집합, 유한집합
S : 순차적 상태변수 집합, 유한집합
Y : 출력 사건의 집합, 유한집합
δ_{ext} : 외부 전이 함수($Q \times X \rightarrow S$)
δ_{int} : 내부 전이 함수($S \rightarrow S$)
λ : 출력 함수($S \rightarrow Y$)
ta : 시각 전진 함수($S \rightarrow R_0^{+\infty}$)

원소 모델의 7개의 요소 중에서 처음 3개의 요소는 시스템의 입력, 상태 그리고 출력 집합이다. 그리고 다음 4개의 요소는 앞의 3개 요소의 제약사항을 규정한다. 또, $R_0^{+\infty}$은 0을 포함한 양의 실수의 집합이고 Q는 원소 모델 AM의 전체상태로 다음과 같이 정의된다.

$Q = \{(s,e) \mid s \in S \text{ and } 0 \le e \le ta(s)\}$

원소 모델은 모델의 행위를 나타내는 반면 모델의 계층적인 구조를 나타내는 결합 모델은 복합형 구성요소로서 원소 모델들이나 다른 결합 모델들로 구성된다. 또한 그 자체보다 큰 규모의 결합 모델의 구성요소가 될 수 있다. DEVS 형식론에서는 결합 모델을 이용해서 계층적인 구조를 갖는 복잡한 모델을 쉽게 구성할 수 있게 된다. 결합 모델의 정의는 다음과 같다[5].

$CM = < X, Y, M, EIC, EOC, IC, SELECT >$

여기서,

X : 입력 사건의 집합, 유한집합
Y : 출력 사건의 집합, 유한집합
M : DEVS 구성요소 모델의 집합, 유한집합
EIC : 외부입력 연결관계
(EIC $\subseteq$ CM.IN $\times$ M.IN)
EOC : 외부출력 연결관계
(EOC $\subseteq$ M.OUT $\times$ CM.OUT)
IC : 내부 연결관계 (IC $\subseteq$ M.OUT $\times$ M.IN)
SELECT : subset of M $\rightarrow$ M, 여러 구성요소 모델들이 같은 시각에 스케줄을 원할 때 그 중에서 하나를 고르는 함수. 같은 시각에 내부 상태전이를 해야 할 모델이 여럿일 경우에 그것들을 순서적으로 처리하는 역할을 한다.

DEVS로 모델링된 시스템의 시뮬레이션을 위하여 DEVS 추상화 시뮬레이터 알고리즘이 제안되었다. DEVS 추상화 시뮬레이터는 DEVS 모델들을 이용한 시뮬레이션 알고리즘으로서, DEVS 모델들의 행위를 해석해 준다. DEVS 추상화 시뮬레이터는 원소 모델을 위한 Simulator와 결합 모델을 위한 Coordinator, 전체 스케줄 관리를 위한 Root coordinator로 나뉜다. DEVS 시뮬레이션은 DEVS 형식론에서 정의된 여러 함수들을 수행함으로서 진행되는데, 이들은 DEVS 추상화 시뮬레이터에서 내부사건과 외부사건을 처리할 때 호출된다.

DEVSim++[5]는 이산 사건 시스템을 시뮬레이션하기 위해 모델링과 추상화 시뮬레이터를 연계시킨 개념에서 DEVS 형식론을 실현한 것이다. DEVSim++는 일반인들에게 잘 알려진 C++ 언어를 기반으로 구축되어 사용자에게 손쉬운 코딩과 빠른 시뮬레이션 속도를 제공한다.

2.2 유전자 알고리즘

1975년 Holland에 의해 소개된 유전자 알고리즘은 자연 생태계의 진화이론을 이용한 탐색과정이라 할 수 있다. 유전자 알고리즘은 탐색공간에서 찾고자하는 해의 후보를 나타내는 염색체(chromosome)와 염색체들 사이의 정보교환을 위한 연산자들과, 염색체의 적합성을 나타내는 함수(fitness function)

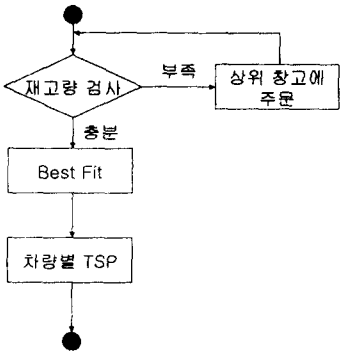
```
Procedure SimpleGeneticAlgorithm()
begin
   // Population: a pool of chromosomes
   Population P, M;
   initialize P;
   for each chromosome of P
      evaluate fitness of the chromosome;
   while termination condition is not satisfied do
      M = apply crossover operator to chromosomes of P
         with a selection rule using a roulette wheel
         with slots sized according to fitness;
      P = apply mutation operator to chromosomes of M;
      for each chromosome of P
         evaluate fitness of the chromosome;
   endwhile
end
```

<그림 2> 기본적인 유전자 알고리즘

등으로 이루어져 있다. 염색체는 일반적으로 이진 문자열로 표현되며, 염색체를 처리하기 위한 연산자로는 부모의 형질을 자식에게 유전시키기 위한 교배 연산자(crossover operator)와 잃어버린 유전 형질을 복구하기 위한 돌연변이 연산자(mutation operator) 등이 있다. 연산자들은 염색체 표현방법이나 적용할 문제의 특성에 따라 변형되어 사용되기도 한다[6]. 유전자 알고리즘은 풀고자 하는 문제에 대한 가능한 해들을 정해진 형태의 자료구조(염색체)로 표현한 다음 점차적으로 변형함으로써 더 좋은 해들을 생성한다.<그림 2>는 기본적인 유전자 알고리즘이다. 전형적인 유전자 알고리즘은 임의의 값으로 초기화된 개체들의 집합으로 시작한다. 각각의 개체는 상대적인 문제해결 능력에 따라 그 적합도(fitness)가 평가되며 적합도에 따라 다음 세대에 부모의 유전자가 복제되는 정도를 달리 함으로써 우성 형질을 지닌 개체들은 열성 형질을 지닌 개체들에 비하여 더욱 많은 자식을 생성할 수 있도록 유도된다. 이러한 선택 메카니즘은 다아윈의 진화론에서의 적자 생존의 원리에 연유한다. 선택 복제된 개체들은 교배, 돌연변이 연산자들에 의해 재결합되어 다음 세대의 개체군을 형성한다. 이와 같은 세대 교체는 원하는 수준의 해가 개체군 내에 존재하거나 또는 다른 종료 조건이 만족될 때까지 반복된다. 유전자 알고리즘이 기존의 탐색 또는 최적화 방법과 달리 점(point)이 아닌 군(population)에 기반한 탐색 방법으로서 알고리즘에서의 탐색은 확률적 연산자를 사용하여 수행되고 탐색공간에 대해 연속성이나 미분가능성 등의 제약을 요구하지 않는다.

3. 배차 알고리즘

이 장에서는 휴리스틱 방법과 2장에서 살펴본 유전자 알고리즘을 이용하여 적재 알고리즘과 차량 스케쥴링 알고리즘을 개발한다. 각 창고에서는 판매처로부터 주문을 받으며 주문 받은 상품들이 창고에 있는지 재고량을 검사한 후, 부족하면 상위 창고에 주문을 하고 충분하면 상품들을 배달한다. 배달 시에는 차량의 수를 최소화하기 위해 BF(Best Fit) 방법을 적용하고 차량별 TSP (Traveling Salesman Problem) 과정을 거쳐서 운송 스케쥴을 생성한다(<그림 3> 참조).



<그림 3> 배차 알고리즘

제안하는 배차 알고리즘에서는 최소 차량에 상품들을 적재하기 위하여 상자 채우기 문제를 적용하였다. 상자 채우기 문제는 다양한 크기의 항목들을 최소의 상자에 채우는 문제이다. 상자 채우기 문제를 해결하는 방법에는 FF(First Fit), FFD (First Fit Decreasing), BF(Best Fit), BFD(Best Fit Decreasing) 등 여러 가지가 있다. 각각의 방법은 다음 <표 1>과 같다[7].

<표 1> 상자 채우기 문제를 위한 알고리즘들

알고리즘	내용
FF	항목을 채울 수 있는 첫 번째 상자에 채운다.
FFD	항목들을 내림차순으로 정렬한 후 FF를 적용한다.
BF	항목을 채웠을 경우 남는 영역의 크기를 모든 상자에 대해 검사한 후 남는 영역이 가장 작은 상자에 채운다.
BFD	항목들을 내림차순으로 정렬한 후 BF를 적용한다.

제안하는 알고리즘에서는 성능이 좋으면서도 빠른 BF 방법을 사용하였다. 우선 주문한 상품을 다음 형식으로 구분한다.

<표 2> 상품 주문량

	d101	d1o2	d1o3	d2o1	d2o2	d2o3	...
차량1	10	20					
차량2							
...							

*d1o2*은 판매처1에서 주문한 상품2을 뜻한다. 즉 같은 종류의 상품이라도 도착할 판매처가 다르면 다른 상품으로 취급한다. 그 후 BF를 적용하여 차량에 적재될 상품들을 결정한다. BF는 모든 차량을 검사해서 그 상품을 적재해서 남는 영역이 가장 적은 차량에 적재한다. 예를 들어 음영으로 표시된 항목은, 차량1에는 판매처1에서 주문한 상품1 10개가 적재된다는 내용이다. 상품이 어떤 창고에 있는지는 설정으로써 알고 있기 때문에 위 테이블이 완성되면 각 차량별로 어떤 창고에 들려서 어떤 상품을 몇 개 적재하고, 어떤 판매처에 들려서 어떤 상품을 몇 개 하차할 것인지가 결정된다. 다음으로 할 일은 운송비를 줄이기 위해 최소 경로를 찾는 것이다.

제안하는 배차 알고리즘에서는 최소 경로를 결정하기 위해 TSP를 적용하였다. TSP란 방문할 도시들이 정해져 있는 상인이 최소의 거리로 모든 도시를 방문하도록 순서를 정하는 문제이며, 이 문제는 다항 시간 내에 해결하는 최적의 알고리즘이 존재하지 않는다. 더욱이 본 논문에서 해결하려는 TSP는 전통적인 TSP와는 달리 방문지를 방문하는 순서에 제한이 있다. 즉 특정 판매처에 도착하기 전에 그 판매처에서 주문한 상품들이 차량에 적재되어 있어야 한다. 이를 위해서는 차량은 항상 창고를 먼저 방문하고 판매처를 나중에 방문하도록 한다. 본 논문에서는 TSP를 해결하기 위하여 유전자 알고리즘을 적용하였다. 이를 위해서는 염색체의 구조, 교배 연산자, 돌연변이 연산자, 적합도 함수 등이 정의되어야 한다.

염색체는 유전자 알고리즘에서 문제에 대한 해의 후보를 표현하는 것으로 일반적으로 이진스트링으로 표현한다. 그러나 이진 스트링은 TSP에는 부적합하기 때문에 제안하는 알고리즘에서 염색체를 <그림 4>와 같이 경로 형식으로 표현하였다. 염색

체의 각 부분은 순서에 따라 차량이 방문해야 할 장소들을 표현한다.

창고1	창고2	판매처1	판매처2

<그림 4> 염색체(경로표현방식)

여러 교배 방법이 TSP 해결을 위해 제안되었으나, 본 논문에서는 EER(Enhanced Edge Recombination) 방법을 적용하였다[8]. <그림 5>는 EER 방법을 이용한 교배연산자의 알고리즘을 나타내었다.

부모 염색체 M과 F에 대해 교배 연산을 하기 위해서는 우선 부모 염색체들로부터 에지 리스트를 생성해야 한다. 에지 리스트는 여행에서 특정 도시에 전후로 방문할 도시들을 표시하는 것이다. 이때 특정 도시가 두 번 나타나면 음수로 한번 나타나면 양수로 표시한다. 예를 들어 M=(1 2 3 4 5 6 7 8 9), F=(4 1 2 8 7 6 9 3 5)의 두 부모가 있는 경우를 생각해보자. 각각의 부모 염색체는 9개의 도시로 구성되어 있다. 예를 들어 여행에서 도시 1을 전후로 방문할 도시들을 찾아보면 M의 경우 도시 9와 도시 2, F의 경우 도시 4와 도시 2이다. 그러므로 공통적으로 나타난 도시 2는 -2로 표시하고 도시 9와 도시 4는 그냥 9와 4로 표시한다. <표 3>은 전체 도시에 대한 에지 리스트가 생성된 결과이다.

<표 3> 염색체 (1 2 3 4 5 6 7 8 9)와 (4 1 2 8 7 6 9 3 5)의 에지 리스트

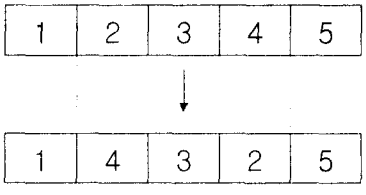
도시	에지 리스트			
1	9	-2	4	
2	-1	3	8	
3	2	4	9	5
4	3	-5	1	
5	-4	6	3	
6	5	-7	9	
7	-6	-8		
8	-7	9	2	
9	8	1	6	3

```
Function chromosome crossover(chromosome M, chromosome F)
// chromosome: a sequence of cities to be visited
begin
   chromosome C;
   generate edge lists from M and F;
   city = select a city randomly;
   C.visit(city);
   while C is not filled do
      if city.edgelist contains only one negative city then
         city = the negative city;
      else if city.edgelist contains two negative cities then
         city = one of the two negative cities which has a shorter edge list;
      else then     // all cities are positive
         city = one of cities in the city.edgelist which has the shortest edge list;
      endif
      C.visit(city);
   endwhile
   return C;
```

<그림 5> 교배연산자

에지 리스트가 생성되면 자식 염색체의 각 부분을 채우는 과정이 수행된다. 처음 장소는 무작위로 선택된다. 그 다음에는 에지 리스트에서 음수로 표시된 도시의 수에 따라 처리가 다르다. 음수로 표시된 도시는 양수로 표시된 도시에 비해서 높은 우선 순위를 가진다. 이것은 부모 염색체에서 공통적으로 나타난 유전 요인이 자식에게 유전될 확률이 한쪽 염색체에서만 가진 유전 요인의 유전 확률보다 높은 것을 의미한다. 같은 우선 순위의 도시에 대해서는 도시의 에지 리스트의 길이가 짧은 쪽이 선택된다. 예를 들어 처음에 도시1이 선택되었다면 음수로 표시된 도시의 개수가 2 하나이므로 도시 2가 선택된다. 도시 2의 에지 리스트를 검사하면 도시 1은 이미 선택되었고, 도시 3과 도시 8 중 에지 리스트가 짧은 도시 8이 선택된다. 도시 8의 에지 리스트를 검사하면 도시 2는 이미 선택되었고 도시 7과 도시 9 중 음수인 7을 선택한다. 도시 7의 에지 리스트를 검사하면 도시8은 이미 선택되었기 때문에 도시6을 선택한다. 지금까지 선택된 장소를 연결하면 (1 2 8 7 6 x x x x)이다. 위 과정을 반복하여 모든 장소를 채우면 자식이 생성된다.

TSP를 위한 돌연변이 연산자 또한 여러 방법들이 제시되었다. 본 논문에서는 단순 역순법(simple inversion method)을 적용하였다[8]. <그림 6>에 나타난 바와 같이 단순 역순법은 선택 구간을 역순으로 바꾼다.



<그림 6> 돌연변이 연산자

각 염색체의 적합도는 다음 세대로 유전되는 비율에 영향을 미친다. 제안하는 알고리즘에서는 다음의 적합도 함수 F_i를 사용하였다.

식(1)에서 t_r은 해당 판매처까지의 운행 소요 시간을, t_c는 해당 판매처의 배달 요구 시간을 나타낸다. d는 해당 염색체 내에 있는 판매처의 개수이다. f_{adt}는 염색체 내에 있는 모든 판매처까지의 운행 소요 시간의 합으로서, 배달 요구 시간을 만족하는 염색체를 보다 많이 유전시키기 위해 사용된다. f_{min}는 염색체 내에 있는 모든 판매처의 배달 요구 시간의 합으로서, F_i'가 음수일 수 있기 때문에 F_i를 양수로 만들어 주기 위한 항목이다. F_i'은 판매처의 요구시간보다 어느 정도나 빨리 도착했는지를 나타내는 함수이다. 적합도 함수 F_i는 판매처에 빨리 도착하는 스케줄을 갖는 염색체를 많이 유전시키고, 또한 요구시간을 만족하지 않는 염색체보다는 요구시간을 만족하는 염색체를 많이 유전시키는 역할을 한다.

위에서 정의된 염색체의 형식, 교배 연산자, 돌연변이 연산자, 적합도 함수들을 <그림 1>의 기본 유전자 알고리즘에 적용하여 TSP를 해결하였다. 제안한 알고리즘에서 사용한 교배연산 확률과 돌연변이 연산 확률은 각각 06, 0.05이다.

본 논문에서 제안한 배차 알고리즘의 목적은 최소의 차량에 상품들을 할당한 후 TSP를 적용하여 최단경로로 운행하게 함으로써 물류 비용을 줄이는 것이다. 그러나 제안한 배차 알고리즘에는 유전자 알고리즘이 포함되어 있어서, 이 알고리즘이 최적해 또는 최적에 근사한 해를 찾는다는 보장을 할 수는 없다.

$$F_i = \begin{cases} F_i' + f_{\min} + f_{adt} & \text{배달 요구 시간을 만족하는 경우} \\ F_i' + f_{\min} & \text{배달 요구 시간을 만족하지 않는 경우} \end{cases} \quad (1)$$

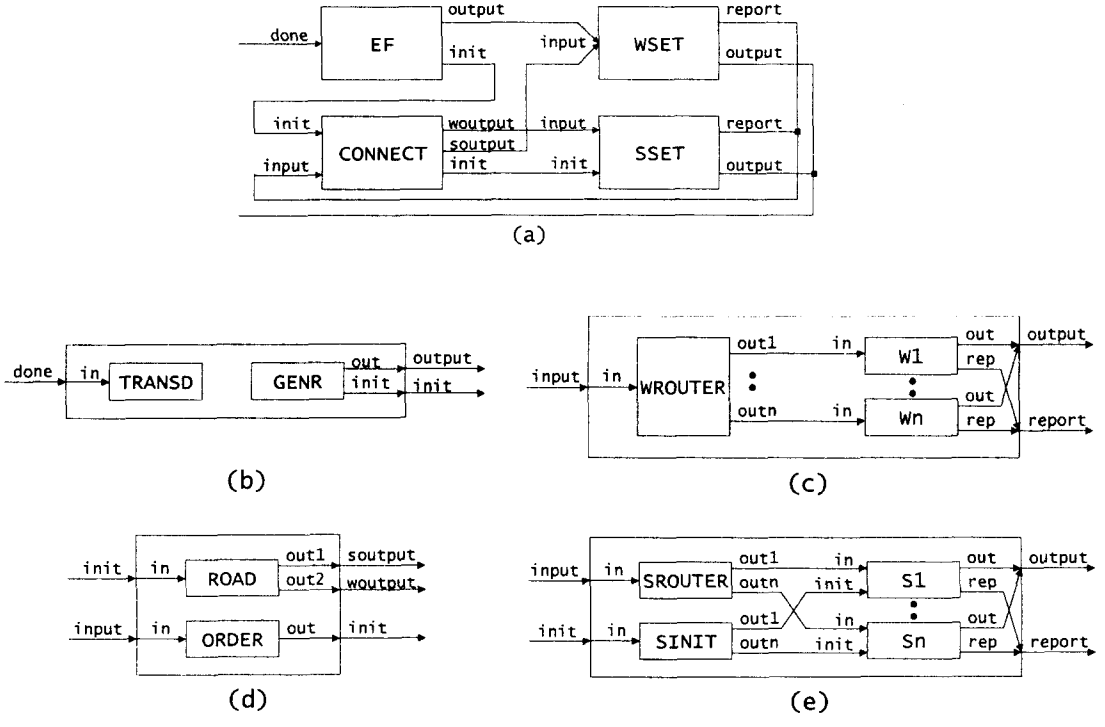
$$F_i' = \sum_{j=1}^{d} \frac{t_c^j - t_r^j}{t_r^j}, \quad f_{\min} = \sum_{j=1}^{d} t_c^j, \quad f_{adt} = \sum_{j=1}^{d} t_r^j$$

4. 물류 시뮬레이터 설계 및 모델링

물류 시뮬레이터의 대상 시스템은 다수의 차량을 이용하여 다수의 창고에서 다종의 상품들을 다수의 판매처로 운송하는 시스템이다. 각 차량은 들려야 할 창고와 판매처, 각 창고와 판매처에서 적재하거나 하차할 상품의 종류와 양을 스케줄로서 갖고 있다. 물류 시뮬레이터는 정해진 스케줄을 받은 차량을 메시지로 전달하여 창고에 도착하면 정해진 상품들을 정해진 양만큼 적재하고, 판매처에 도착하면 정해진 상품들을 정해진 양만큼 하차한다. 물류 시뮬레이터는 판매처에서 요구한 시간 내에 상품들이 도착하는지 검증하기 위해 창고나 판매처간의 이동 소요시간 등을 분석할 수 있도록 한다.

<그림 7>은 본 논문에서 구현된 물류 시뮬레이터를 나타낸 것이다.

시스템은 크게 *EF, WSET, SSET, CONNECT* 등 4개의 결합 모델로 구성된다. 창고들과 판매처들을 운행하며 상품들을 운송하는 차량들은 모델사이에 전달되는 메시지로 표현되며, 각각의 차량은 배차알고리즘을 통하여 얻어진 운행 경로를 포함한다. *EF*는 실험장치로서, 시뮬레이션을 위한 입력을 생성하고, 시뮬레이션 결과를 분석하는 역할을 한다. *EF*는 차량을 내보내는 *GENR*, 결과를 출력하는 *TRANSD* 등의 원소 모델로 구성된다. *WSET*은 창고의 집합으로서 상품들을 저장하고 있으며, 차량이 도착하면 차량에 상품을 적재시킨다. *WSET*은 창고를 나타내는 *W1*, ..., *Wn*과 스케줄에 따라 창고로 차량을 보내주는 *WROUTER*



(a) 전체시스템 (b) 결합모델 *EF* (c) 결합모델 *WSET* (d) 결합모델 *CONNECT* (e) 결합모델 *SSET*

<그림 7> 물류 시뮬레이터의 구성도

등의 원소 모델로 구성된다. *SSET*은 판매처의 집합으로서 상품들을 주문하고, 차량이 도착하면 상품을 하차시키는 역할을 한다. *SSET*은 판매처를 나타내는 *S1, ..., Sn*, 스케줄에 따라 판매처로 차량을 보내주는 *SROUTER*, 각 판매처에 주문 내용을 보내주는 *SINIT* 등의 원소 모델로 구성된다. *CONNECT*는 창고들과 판매처들 사이의 도로망을 모델링하는 역할을 한다. *CONNECT*는 도로망을 표현하고 차량을 창고와 판매처로 구분해서 보내주는 *ROAD*, 각 판매처에 주문 내용을 초기화시키는 *ORDER* 등의 원소 모델로 구성된다.

본 논문에서는 위에서 제시된 여러 원소모델들에 대해서 각각 DEVS 형식론으로 모델링하였다. 그러나 지면 관계상 창고 모델에 대해서 예를 들어 설명하겠다. 창고의 원소 모델 AM_{ware}는 <그림 8>과 같이 정의된다.

외부 전이 함수는 외부로부터의 입력에 대해 모델 상태의 변화를 정의한다. 창고에 차량이 입력으로 들어오면 우선 그 차량의 스케줄이 해당 창고에 도착할 순서인지 *CrtDest()* 함수로써 검사한다. 검사 결과, 그 차량이 해당 창고에 도착할 순서라면, 차량의 도착 시각을 기록하고 상품들을 적재한다. 그런 다음 *CalcNextTravelTime()* 함수를 호출하여 다음 도착지까지의 운행 시간을 계산하고 여기에 상품을 적재하는 시간 *tload*와 현재 시각을 더하여 그 차량의 다음 스케줄 시각을 계산하여 기록한다. 스케줄이 끝난 차량을 *queue*에 넣는다. 그리고 만약 현재 상태가 *idle* 상태일 경우 *busy* 상태로 변

AM_{ware} = < X, S, Y, δ_{ext}, δ_{int}, λ, ta >

X = {input}
Y = {output, report, done}
S = {phase(={busy, idle}), queue, name}

δ_{ext}(s, e, x)
```
begin
   car = x.GetValue();
   tc = GetTime();
   if (car.CrtDest() == s.name) then
      car.CrtArrival(tc);
      car.load();
      tgo = CalcNextTravelTime(car);
      car.NextSchedule(tc + tload + tgo);
      s.queue.Add(car);
      if (s.phase == idle) s.phase = busy;
   endif
end
```

δ_{int}(s)
```
begin
   if (s.phase == busy) then
      if (s.queue.Length() > 0) then
         for each car in s.queue
            if (car.NextScheule() == GetTime())
               x.queue.Remove(car);
         endfor
      endif
      if (s.queue.Length() == 0)
         s.phase = idle;
   endif
end
```

λ(s)
```
begin
   if (s.phase == busy) then
      tc = GetTime();
      for each car in s.queue
         if (car.NextSchedule() == ta) then
            send(car, "output");
            send(car, "report");
         endif
      endfor
end
```

ta(s)
```
begin
   if (s.phase == busy) then
      ta = ∞;
      tc = GetTime();
      for each car in s.queue
         if (ta > car.NextSchedule() - tc)
            ta = car.NextSchedule() - tc;
      endfor
      return ta;
   else
      return ∞;
   endif
end
```

<그림 8> 창고 원소 모델 AM_{ware}

환한다.

내부 전이 함수는 외부 전이 함수가 발생된 뒤 변화된 상태에 해당하는 시각 전진 함수 만큼 시간이 경과했을 경우 모델 상태의 변화를 정의한다. *phase*가 *idle*인 경우, 내부 전이 함수는 어떤 동작도 하지 않는다. *phase*가 *busy*이면, *queue*에서 현재 시각으로 스케쥴된 차량들을 제거한다. 그런 다음, *queue*의 길이가 0이면 *phase*는 *idle*이 된다.

출력 함수는 외부 전이 함수에 따라 변화된 상태에 해당하는 모델의 출력을 나타낸다. *phase*가 *idle*이면 출력 함수는 어떤 동작도 하지 않는다. *phase*가 *busy*이면 *queue*에서 현재 시각으로 스케쥴된 차량들을 출력 포트 *output*으로 출력한다. 또 결과 분석을 하기 위해 출력 포트 *report*로 시뮬레이션 결과를 출력한다.

시각 전진 함수는 모델에 외부 입력이 없을 때 어떤 상태에 얼마나 머무를 수 있는지를 나타내는 함수이다. *phase*가 *idle*이면 무한대, *busy*이면 *queue*에 있는 차량 중에서 가장 빠른 시각으로 스케쥴된 차량의 스케쥴 시각과 현재 시각의 차이를 리턴한다.

모델링된 시스템을 시뮬레이션하기 위해 DEVSim++를 이용했다. DEVSim++는 DEVS 형식론을 C++ 언어로 표현한 것이다. 구현된 시뮬레이터는 정해진 스케쥴을 가진 차량을 메시지로 전달하여 창고에 도착하면 정해진 상품들을 정해진 양만큼 적재하고, 판매처에 도착하면 정해진 상품들을 정해진 양만큼 하차한다. 그리고 도착시간, 출발시간 등을 분석하고, 최종적으로 원하는 시간 내에 배달되었는지 검증한다.

5. 시뮬레이션

제안된 알고리즘과 물류 시뮬레이터를 검증하고 성능을 측정하기 위하여 여러 가상 데이터에 대해 실험하였다. 그러나 지면 관계상, 본 논문에서는 간단한 예제에 대한 시뮬레이션 결과를 보이겠다. 실험에 사용된 물류망은 3개의 창고와 3개의 판매처로 구성된 시스템이다. 3종류의 상품들이 존재하며 각 상품들은 각기 다른 창고에서 취급하며 충분한 양의 재고가 있는 것으로 가정하였다. 차량 1대의 최대 적재량은 150으로, 상품의 무게는 종류에 상관없이 개당 1로 설정하였다. 그리고 시뮬레이션 시작시간은 오전 9시로 설정하였고 배달 요구시간은 모두 정오로 설정했다. 그리고 각 창고와 판매처에서의 적재 또는 하차시간은 5분으로 설정했다.

시뮬레이션을 위해서는 각각의 판매처와 창고 모델들은 <그림 5>의 *SSET*와 *WSET* 결합모델의 하위 모델로 구성된다. 표4는 시뮬레이션에서 사용된 각 판매처와 창고 사이의 운행시간이다. 이것을 이용하여 창고 모델이나 판매처 모델에 차량이 외부사건으로 입력되면 출력시간을 계산한다.

<표 4> 운행시간

	W1	W2	W3	S1	S2	S3
W1	0	34	26	53	74	33
W2	34	0	23	76	31	45
W3	26	23	0	26	42	56
S1	53	76	26	0	12	45
S2	74	31	42	12	0	22
S3	33	45	64	45	22	0

<표 5>는 시뮬레이션이 시작될 때 각 판매처에서 주문한 상품의 명세이다. 앞서 언급한대로 각 상품은 서로 다른 창고에서 취급하는 것으로 하였다. 그래서 상품 1은 창고 1에서, 상품 2는 창고 2에서, 상품 3은 창고 3에서만 취급하며 재고는 충분한 것으로 가정하였다.

<표 5> 주문내용

	상품1	상품2	상품3
S1	130	200	90
S2	20	60	30
S3	100	40	70

<표 6>은 <표 5>에서 설정된 주문에 따라 제안 알고리즘에서 배차한 후 스케쥴을 생성한 결과이

<표 6> 생성된 스케쥴

차량	운송 스케쥴				
1	(W2,2,20)	(W1,1,130)	(S1,1,-130)(S1,2,-20)		
2	(W2,2,150)	(S1,2,-150)			
3	(W1,1,20)	(W3,3,90)	(W2,2,40)	(S2,1,-20)(S2,2,-10)	(S1,2,-30)(S1,3,-90)
4	(W1,1,70)	(W3,3,30)	(W2,2,50)	(S2,2,-50)(S2,3,-30)	(S3,1,-70)
5	(W2,2,40)	(W3,3,70)	(W1,1,30)	(S3,1,-30)(S3,2,-40)(S3,3,-70)	

다. 모두 5대의 차량이 운송에 사용되었으며 경로가 짧은 차량은 두 곳만 방문하면 되지만 경로가 긴 차량의 경우는 5곳을 방문하게 된다. 이것은 제안된 배차 알고리즘이 차량의 방문지 수는 고려하지 않고 배달량만 고려하여 차량을 배차하기 때문이다.

생성된 스케쥴을 이용하여 시뮬레이션을 수행하였다. 표 7은 시뮬레이션 결과로서 각 차량의 방문지 도착시간 및 출발시간을 나타낸 것이다. 결과를 살펴보면 판매처마다 배달 완료 시간이 조금씩 다르지만 가장 배달이 늦은 경우라도 실험에서 가정한 도착 마감시간인 정오보다 이르게 배달이 완료되는 것을 알 수 있다.

위 실험은 Pentium-II 400, 128 MB RAM, Windows 98 상에서 수행하였다. 시간은 배차 알고리즘부분에서 5~6초, 시뮬레이션 부분에서 10초 정도 소요되었다.

<표 7> 시뮬레이션 결과

차량	각 단계별 도착/출발시간				
1	9:00/9:05	9:39/9:44	10:37/10:42		
2	9:00/9:05	10:21/10:26			
3	9:00/9:05	9:31/9:36	9:59/10:04	10:35/10:40	10:52/10:57
4	9:00/9:05	9:31/9:36	9:59/10:04	10:35/10:40	11:02/11:07
5	9:00/9:05	9:28/9:33	9:59/10:04	10:37/10:42	

위 실험에서 배달요구시간을 정오가 아닌 오전 10시와 오전 11시로 설정하여 실험한 결과, 스케쥴은 <표 6>과 동일하게 생성되어 <표 7>과 동일한 시뮬레이션 결과를 나타냈다. 즉 판매처의 배달요구시간을 만족하지 못하는 결과가 나타났다.

6. 결론

본 논문의 목표는 자연 생태계의 진화이론을 이용한 탐색과정인 유전자 알고리즘을 이용한 배차 알고리즘의 개발과 이산 사건 시스템을 계층적이고 모듈화된 형태로 기술하는 수학적인 언어인 DEVS 형식론을 이용하여 물류 시뮬레이터를 구현하는 것이다. 본 논문의 대상 시스템은 다수의 차량을 이용하여 다수의 창고에서 다종의 상품들을 다수의 판매처로 운송하는 시스템이다. 각각의 판매처에서 주문하면, 상자 채우기 문제를 해결하는 방법인 BF 방법과 TSP(Traveling Salesman Problem)를 적용하여 스케쥴을 생성한 후 시뮬레이션하였다. 각각의 창고에서는 판매처에서 요구한 상품들을 적재하여 판매처에서는 원하는 상품들을 하차하고 정해진 시간 내에 배달되는지를 검증하였다. 모델링된 시스템을 시뮬레이션하기 위해 DEVSim++를 이용하였다. DEVSim++는 DEVS 형식론을 C++ 언어로 표현한 것이다. 여러 가상의 데이터로 배차 알고리즘을 적용한 후 시뮬레이션한 결과 적절히 동작하는 것을 알 수 있었다. 현재 계층적인 물류 시스템에 제안된 알고리즘을 적용하기 위한 연구가 진행중이다.

참고문헌

[1] Theodore P. Stank and Patrick A. Traichal, "Logistics Strategy, Organizational Design, And Performance In A Cross-Border Environment", *Transpn Res.-E(Logistics and Transpn Rev.)*, Vol. 34, No. 1, pp. 75-86, 1998

[2] Laura Meade and Joseph Sarkis, "Strategic Analysis of Logistics and Supply Chain Management Systems Using The Analytical Network Process", *Transpn Res.-E(Logistics and Transpn Rev.)*, Vol. 34, No. 3, pp. 201-215, 1998

[3] Bernard P. Zeigler, *Multifacetted Modelling and Discrete Event Simulation*, Academic Press, 1984.

[4] Tag Gon Kim, *DEVSim++ User's Manual: C++ Based Simulation with Hierarchical Modular DEVS Models*, Computer Engineering Lab., Dept. of Electrical Engineering, KAIST, 1994.

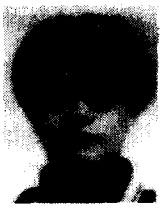
[5] 안명수, 박성봉, 김탁곤, "DEVSim++: 의미론에 기반한 이산사건 시스템의 객체지향 모델링 및 시뮬레이션 환경," 한국정보과학회논문지, 제21권, 제9호, pp. 1652-1664, 1994.

[6] David E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison Wesley, 1989.

[7] Ellis Horowitz and Sartaj Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, 1978.

[8] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Second Extended Edition, Springer-Verlag, 1994.

저자소개



조병헌 (e-mail : cbh@zeus.kookmin.ac.kr)
1997년 2월 국민대학교 전자공학과 (공학사)
1999년 8월 국민대학교 전자공학과 (공학석사)
1999년 9월 ~ 현재 국민대학교 전자공학과 박사과정
관심분야 : 컴퓨터구조, 시뮬레이션, 멀티미디어

정성훈
1984 ~ 1988 : 한양대 전자공학과 공학사
1989 ~ 1991 : 한국과학기술원 전기및 전자공학과 공학석사
1991 ~ 1995 : 한국과학기술원 전기및 전자공학과 공학박사
1995 ~ 1996 : 한국과학기술원 전기및 전자공학과 위촉연구원
1996 ~ 1998 : 한성대학교 정보전산학부 전임강사
1998 ~ 현재 : 한성대학교 정보전산학부 조교수

● 저자소개 ●

김기형
1990년 2월 : 한양대학교 전자통신공학 학사
1992년 2월 : 한국과학기술원 전기 및 전자공학과 석사
1996년 8월 : 한국과학기술원 전기 및 전자공학과 박사
1996년 8월~1997년 2월: 한국과학기술원 위촉연구원
1997년 3월~현재 : 영남대학교 컴퓨터공학과 전임강사
관심분야 : 네트워크기반컴퓨팅, 멀티미디어운영체제, 시스템모델링 및 성능평가, ATM네트워크관리

오하령 (e-mail : hroh@kmu.kookmin.ac.kr)
1983년 서울대학교 전기공학과(공학사)
1983년~1986년 삼성전자 종합연구소
1988년 한국과학기술원 전기전자과 컴퓨터공학전공(공학석사)
1992년 한국과학기술원 전기전자과 컴퓨터공학전공(공학박사)
1992년 ~ 1996년 국민대학교 공과대학 전자공학부 조교수
1996년 ~ 현재 국민대학교 공과대학 전자공학부 부교수
관심분야 : 컴퓨터구조, 운영체제, 분산 및 병렬처리, 멀티미디어

성영락 (e-mail : yeong@kmu.kookmin.ac.kr)
1989년 한양대학교 전자공학과, 공학사
1991년 한국과학기술원 전기 및 전자공학과, 공학석사
1995년 한국과학기술원 전기 및 전자공학과, 공학박사
1995년~1996년 한국과학기술원 위촉연구원,
1996년~1998년 국민대학교 전임강사
1998년~현재 국민대학교 조교수
관심분야 : 멀티미디어 시스템, 시스템 모델링 및 시뮬레이션, 병렬처리