한국지능정보시스템학회
02 춘계학술대회 논문집, pp. 323~332

최적화 에이전트를 위한 사례기반의 자동 모형화

장용식*·이재규**

요 약

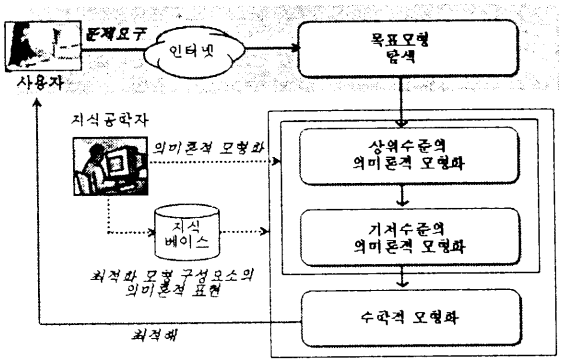
전자상거래와 같은 분산컴퓨팅환경에서는, 의사결정을 위해 협동적 문제해결과 사례기반의 자동 모형화가 더욱 중요시 되고 있다. 왜냐하면, 문제요구는 다양하고 이에 대응하기 위해 모든 모형을 준비한다는 것은 실제로 어려우며, 모형의 저장 및 관리관점에서도 비효율적이기 때문이다. 이에 따라, 최적화 에이전트 기반의 자동 모형화에 의한 문제해결을 위한 연구의 필요성이 인식되고 있다. 본 연구에서는 최적화 모형에 대한 지식이 부족한 사용자 수준의 XML 표현과 같은 문제요구를 이해하고, 최적화 모형 사례로부터 목표모형을 탐색하는 최적화 에이전트를 위한 사례기반 자동 모형화의 프레임웍을 제시한다. 이를 위해, 자동 모형화 지식의 표현과 목표모형 탐색을 위한 전방향 추론절차를 제시한다. 최적화 에이전트는 모형화 노력을 줄이기 위해서, 민감도분석을 통해 성능이 평가된 탐색 알고리즘을 사용한다.

Key words : 전자상거래, 분산컴퓨팅, 의사결정, 최적화 에이전트, 모형화, 사례기반, 최적화 모형

1. 서론

전자상거래와 같은 분산컴퓨팅 환경에서, 사용자는 의사결정을 위한 많은 문제에 부딪치게 되고 이에 대한 해를 필요로 한다. 이러한 문제는 매우 다양하고 이에 대응하기 위해 모든 모형을 준비한다는 것은 실제로 어렵고 모형의 저장 및 관리관점에서도 비효율적이기 때문에 협동적 문제해결과 사례기반의 자동 모형화가 더욱 중요시 되고 있다. 최적화 에이전트는 최적화 문제를 이해하고 해를 찾아주는 에이전트이다. 최적화 에이전트시스템은 스케쥴링, 협상, 협동을 위한 의사결정지원에 많이 이용되고 있기는 하나[6,15,19], 다양한 문제에 대응하기 위한 자동 모형화에 대한 연구는 거의 없다. 이에 따라, 최적화 에이전트 기반의 자동 모형화에 의한 문제해결 방법에 대한 연구의 필요성이 인식되고 있다. [그림 1]에서 보듯이 이전에는 주로 지식공학자들의 모형화를 지원하기 위한 모형화 언어와 모형화지식에 대한 연구가 많이 진행되어 왔다. 지식공학자는 지식베이스에 최적화 모형의 구성요소를 의미론적 표현으로 관리하고, 문제를 풀기 위해 수학적 모형에 대응하는 기저수준의 의미론적 모형을 생성하거나, 최적화 Solver가 이해할 수 없는 상위수준의 의미를 포함하는 모형화를 통하여 문제를 풀었다. 그러나, 분산환경에서 문제해결에 대한 사용자들의 요구에 대응하고 지원하기 위하여, 사용자 수준의 문제요구로부터 목표모형을 탐색하는 과정이 중요하게 인식되고 있다

[그림 1] 모형화계층 개념도



본 연구에서는 선형계획법(LP)과 정수계획법(IP)과 같은 최적화 모형에 대한 지식이 부족한 사용자 수준의 XML 표현과 같은 문제의 이해 및 최적화 모형사례 기반의 자동모형화 지식을 표현하고, 전방향 추론의 탐색에 기반한 최적화 에이전트의 프레임웍을 제시한다. 본 연구의 구성은 다음과 같다. 제 2장에서 사례기반의 모형화에 관련된 연구와 본 논문에서 사용되는 모형관리도구를 소개하고, 제 3장에서는 최적화 에이전트시스템에 대하여 기술한다. 제 4장에서 자동모형화 지식을 표현하고, 제 5장에서 목표모형 탐색전략과 탐색구조에 대하여 설명하며, 마지막 장에서 결론을 제시한다.

2. 문헌 연구

2.1 사례기반의 최적화 모형의 모형화

모형을 만드는 사람과 의사결정자 사이에

* 한국과학기술원 테크노경영대학원 경영공학
** 한국과학기술원 테크노경영대학원

의견의 격차를 줄이기 위하여 최적화를 위한 지식지원의 모형화가 시도되었으며[2,3,4]. 최적화 모형의 모형화에 대한 사례기반 추론의 접근이 연구되어 왔다. 유추에 의한 추론(analogical reasoning))[4,5,7,20]에 기반을 둔 유추에 의한 모형화(modeling by analogy)가 Liang 등에 의해 연구되었는데[16,18] 이 방법은 개념, 구조, 기능적 유사점의 특징을 매핑(mapping)하면서 새로운 모형을 생성한다. 그러나 모형 구조의 수정이 요구될 때는 비효율적이다. Liang이 지적했듯이, 이러한 접근은 특징 매핑이 이론적으로 NP-complete하기 때문에 규칙기반의 접근보다 상대적으로 경직되어 있고 계산비용이 매우 비싸다. 그리고 구조적으로 매핑이 힘든 큰 문제의 경우 효율적으로 생성될 수 없다[17]. 새로운 문제를 저장된 모형에 매핑하는 대신, Binbasioglu는 프로세스 유추(process analogy) 방법을 제안하였다[1]. 이 방법은 추상적인 여러 수준에서 동일시 되는 재사용 가능한 모형의 부품으로 구성된 기초모형으로부터 모형의 구조를 수정하지 않고 모형을 설계한다. 그러나 많은 경우에 저장된 기본 모형들이 문제를 풀 때 구조 변환없이 재사용 가능한 일은 드문 일이다. 우리는 사용자 수준의 의미론적인 문제요구를 바탕으로 기본모형 또는 모형사례로부터 목표모형을 탐색하는 기본 틀을 제안한다. 다양한 의사결정 형태가 나타나는 전자상거래 및 분산된 환경에서는 이러한 접근에 대한 에이전트 기반의 자동화된 처리 요구가 더욱 증대되고 있다.

2.2 최적화 모형관리 도구: UNIK-OPT

본 연구에서는 프레임(frame) 표현으로 LP와 IP를 위한 최적화 모형과 규칙기반의 시스템을 지원하는 UNIK-OPT[9,12]를 사용한다. UNIK-OPT는 LP 또는 IP 모형을 제약조건, BOT(Blocks Of Terms)와 같은 객체로 표현한다. BOT란 summation sign, 변수 그리고 상수를 공유하는 항들의 집합을 의미한다. Multi-depot vehicle routing problem(M-VRP)[10]의 예를 중심으로 의미론적인 모형의 표현에 의한 문제해결 과정을 간단히 소개한다. 이 모형에 사용되는 표기는 다음과 같다: n: 배송될 매장의 수; m: 창고(depot)의 수; v: 차량의 수; i, j, h: 매장과 창고의 색인, {1,2, ... n, n+1, ... , n+m}; k: 차량의 색인, {1, 2, ... , v }; d_{ij}: i와 j간 이동거리; t_{ij}: i와 j간 이동소요시간; q_i: 매장 i의 수요량, p_k: 차량 k의 용량; l_k: 차량 k의 최대 운송가능거리; t_k: 차량 k의 최대 운송가능시간; s_i: 매장 i에서의 서비스(하역)시간; u_k: 차량 k의 최대 방문가능한 하나의 창고와 매장들의 수; X_{ijk}: 1(i, j가 차량 k의 경로상에 있는 경우), 또는 0(기타 경우); T_i: 방문시간 변수. 수학적인 모형은 아래와 같다.

$$\text{Min} \sum_{i=1}^{n+m}\sum_{j=1}^{n+m}\sum_{k=1}^{v} d_{ij} X_{ijk} \quad (1)$$

제약조건:

$$\sum_{i=1}^{n+m}\sum_{k=1}^{v} X_{ijk} = 1 \quad \text{for } j=1, 2, \ldots, n \quad (2)$$

$$\sum_{j=1}^{n+m}\sum_{k=1}^{v} X_{ijk} = 1 \quad \text{for } i=1, 2, \ldots, n \quad (3)$$

$$\sum_{i=1}^{n+m} X_{ihk} - \sum_{j=1}^{n+m} X_{hjk} = 0$$
$$\text{for } h=1,2,..,n,n+1,\ldots n+m, \quad k=1,2,..,v \quad (4)$$

$$\sum_{i=1}^{n+m} q_i \sum_{j=1}^{n+m} X_{ijk} \le p_k \quad \text{for } k= 1, 2, \ldots, v \quad (5)$$

$$\sum_{i=1}^{n+m}\sum_{j=1}^{n+m} t_{ij}X_{ijk} \le t_k \quad \text{for } k= 1, 2, \ldots, v \quad (6)$$

$$\sum_{i=n+1}^{n+m}\sum_{j=1}^{n} X_{ijk} \le 1 \quad \text{for } k= 1, 2, \ldots, v \quad (7)$$

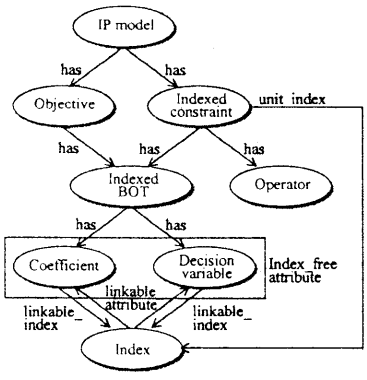
$$\sum_{j=n+1}^{n+m}\sum_{i=1}^{n} X_{ijk} \le 1 \quad \text{for } k= 1, 2, \ldots, v \quad (8)$$

$$Y_i - Y_j + (m+n)X_{ijk} \le n+m-1 \text{ for } 1\le i\ne j\le n,\ 1\le k\le v \quad (9)$$

$$X_{ijk} = 0 \text{ or } 1 \text{ for all } i, j, k \quad (10)$$

(1) ~ (10)식에 의미론적인 이름을 표현하면 다음과 같다: (1)식: total_traveling_distance_BOT; (2)식: drop_in_constraint; (3)식: drop_out_constraint; (4)식: route_continuity_constraint; (5)식: vehicle_capacity_constraint; (6)식: maximum_traveling_time_constraint; (7)식: vehicle_start_constraint; (8)식: vehicle_arrival_constraint; (9)식: subtour_breaking_constraint; (10)식: visit_variable.

[그림 2] 모형화 지식베이스 구조



UNIK-OPT는 LP 또는 IP에 대하여 [그림 2]와 같은 모형화 지식구조[8]에 근거하여 모형화하며, 프레임 형태로 저장한다. 수학적인 표현의 M-VRP를 UNIK-OPT로 모형화하면 다음과 같은 의미론적인 프레임 구조로 표현된다.

[그림 3] M-VRP의 UNIK-OPT 표현

```
{{M-VRP
  IS-A : IP_MODEL
  DIRECTION : min
  OBJECTIVE : total_traveling_distance_BOT      ; (1)
  CONSTRAINT : drop_in_constraint               ; (2)
    drop_out_constraint                         ; (3)
    route_continuity_constraint                 ; (4)
    vehicle_capacity_constraint                 ; (5)
    maximum_traveling_time_constraint           ; (6)
    vehicle_start_constraint                    ; (7)
    vehicle_arrival_constraint                  ; (8)
    subtour_breaking_constraint                 ; (9) }}
{{drop_in_constraint                                    ; (2)
  IS-A : CONSTRAINT
  OPERATOR : EQ
  LHS : (+drop_in_BOT)
  RHS : (+one_BOT)
  UNIT_INDEX: (in_index LE n)                           }}
  .....
{{visit_variable                                        ;
(10)
  IS-A : VARIABLE
  SYMBOL : X
  LINKED_INDEX : out_index    in_index
  TYPE:
binary                                                  }}
.....
```

M-VRP에 IF-THEN 제약조건을 추가해 보자.

$$\text{if } X_{ijk} \geq 1 \text{ then } T_i + t_{ij} \leq T_j \quad \text{for } i= 1,2,\ldots,n,\ j=1,2,\ldots,n,\ k=1,2,..,v \qquad (11)$$

[그림 4] IF-THEN 연산자의 상위수준 표현

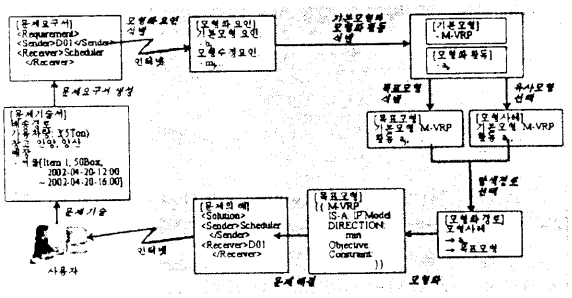
```
{{M-VRP
  IS-A : IP_MODEL
  DIRECTION : min
  OBJECTIVE : total_traveling_distance_BOT
  CONSTRAINT :
    .....
    IF-THEN(visit_constraint,    compatibility_requirement_
between_ routes_and_schedule_constraint)        ; (11-1) }}
  .....
{{visit_constraint
(11-2)
  IS-A : CONSTRAINT
  OPERATOR : GT
  LHS : (+visit_BOT)
  RHS : (+zero_BOT)
  UNIT_INDEX: (out_index  1  n)  (in_index  1  n)
        (vehicle_index  1  v)                       }}
  .....
```

이러한 표현은 IP Solver가 이해할 수 없기 때문에 상위수준의 표현(High-level representation)이라고 한다. UNIK-OPT는 이러한 상위수준 표현능력을 가지고 있으며, 자동으로 IP Solver가 이해할 수 있는 수학적인 모형에 대응하는 기저수준 표현으로 변환할 수 있다. [그림 4]는 UNIK-OPT에서 IF-THEN 연산자의 상위수준 표현을 나타낸 것이다.

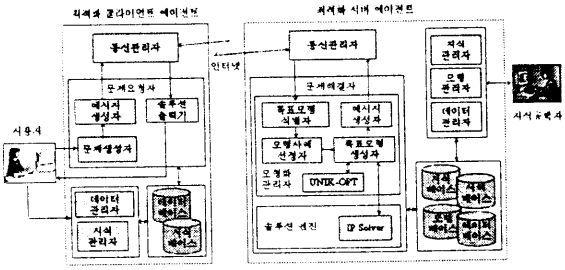
3. 최적화 에이전트시스템

최적화 에이전트시스템은 문제요구서를 만들어 해답을 요구하는 최적화 클라이언트 에이전트와 문제를 인식하고 최적화 모형을 찾아 문제를 풀어 응답하는 최적화 서버 에이전트로 구성된 멀티에이전트시스템이다. 문제를 푸는 과정은 다음과 같다([그림 5] 참조). (1) 최적화 클라이언트 에이전트는 문제상황을 사용자로부터 받거나 스스로 인식하여 문제요구서를 생성하고, 최적화 서버 에이전트에게 문제해결을 요청한다. (2) 최적화 서버 에이전트는 문제요구서로부터 모형화요인을 추출하여 목표모형을 식별한다. (3) 목표모형의 모형화를 위해 유사모형사례를 선정한다. (4) 목표모형화 탐색경로를 선정한다. (5) 모형사례로부터 목표모형으로 모형화한다. (6) 문제의 해를 최적화 클라이언트 에이전트에게 보내준다.

[그림 5] 문제해결 절차



[그림 6] 최적화 에이전트시스템 구조



[그림 6]은 최적화 에이전트시스템의 기본적인 구조를 나타낸 것이다. 각 에이전트는 에이전트간 통신을 담당하는 통신관리자가 있다. 최적화 클라이언트 에이전트의 문제요청자는 사용자로부터 문제요구를 받아들이는 문제생성자, XML 형태의 문제요구 메시지를 생성하여 서버 에이전트에 송신하는 메시지생성자, 최적화 서버 에이전트로 부터의 문제에 대한 해를 수신하여 사용자에게 보여주는 솔루션 출력기로 구성된다. 그리고, 문제생성을 위해 필요한 지식과 데이터를 관리하는 지식관리자와 데이터관리자가 있다. 한편, 최적화 서버 에이전트는 지식관리자, 모형관리자, 데이터관리자, 문제해결자가 있다. 모형관리자는 모형의 의미론적인 표현을 프레임형태로 저장관리하며, 지식관리자는 모형화 활동 및 규칙에 관한 지식을 관리한다. 데이터관리자는 모형생성에 필요한 데이터를 관리

한다. 문제해결자는 모형생성자와 솔루션 엔진으로 구성되어 있다. 모형생성자는 최적화 에이전트로부터 수신한 문제요구로부터 목표모형을 식별하는 목표모형식별자, 유사한 모형사례를 추출하는 모형사례선정자, 모형사례로부터 목표모형을 생성하는 목표모형생성자, 문제의 해를 XML 형태의 메시지로 변환하여 다시 최적화 클라이언트 에이전트에게 돌려주는 메시지생성자, 그리고 의미론적인 모형을 솔루션 엔진의 IP Solver가 이해할 수 있는 수학적인 모형으로 변환하는 UNIK-OPT가 있다. 본 연구에서는 IP solver로 LINDO를 이용하였다.

4. 자동모형화 지식표현

4.1 기본모형의 개념

기본모형(Base model)이란 유사한 문제를 기술하는 모형사례들의 공통 구성요소로 이루어지고, 그 자체로서 타당한 모형을 의미한다. 예를 들면, M-VRP 모형이 기본모형이며, 배달시간 제약조건을 가진 M-VRP 모형은 유사모형사례들에 해당한다.

4.2 모형화요인

모형화요인(Modeling factor)은 기본모형을 식별하는 기본모형요인(Base model factor)과, 목표모형화에 영향을 미치는 모형수정요인(Model modification factor)으로 구성되어 있다. 모형의 수정은 기본모형에 따라 다르기 때문에 모형수정요인은 기본모형요인에 의존적이다. 기본모형요인으로는 배송계획요인, 생산계획요인 등 최적화 모형 분야별로 분류 가능하다. <표 1>은 기본모형요인중 하나인 배송계획에 대한 대안들을 나타낸 예이고, <표 2>는 기본모형요인의 값이 multi_depot_vehicle_routing(b_1)일 때의 모형수정요인과 예시적인 대표적 대안에 대한 설명이다.

<표 1> 기본모형요인의 대안

요인	대안	기본모형
배송계획	*multi_depot_vehicle_routing(b_1)*	M-VRP 모형
	multi_depot_pickup_and_delivery(b_2)	M-PDP 모형
	transportation(b_3)	Transportation 모형

<표 2> 모형수정요인과 대안

요인	대안	제약
차량	*maximum_traveling_time_not_required(m_1)*	차량의 대운송가능시간에 대한 제약조건 삭제
	Maximum_traveling_distance(m_2)	차량의 최대운송가능거리
	Maximum_number_of_visiting_points(m_3)	차량의 방문 가능한 최대매장수.
	fixed_cost_of_utilizing_vehicles(m_4)	차량 운행을 위한 고정비용.
매장 또는 창고	*earliest_delivery_time_window(m_5)*	차량의 방문이 허용된 가장 빠른 시간.
	latest_delivery_time_window(m_6)	차량의 방문이 허용된 가장 늦은 시간.
	service_time(m_7)	매장에서의 서비스 (하역) 시간
벌칙	*penalty_of_tardy_starting_service_time(m_8)*	차량이 방문해야 할 가장 늦은 시간 경과후에 도착할 때의 벌칙
	penalty_of_tardy_routing_duration(m_9)	차량이 경로상에서 최대 허용 운송시간을 경과할 때의 벌칙

4.3 최적화 모형수정 언어

본 연구에서는 UNIK-OPT에서 모형관리를 위해 설계된 최적화모형 수정언어(OMML, Optimization Model Modification Language)[13]를 소개한다. OMML은 [그림 2]의 모형화 지식베이스 구조에서와 같이 모형을 구성하는 목적함수, 최대화 및 최소화, 제약조건, BOT, 연산자, 색인, 의사결정변수, 계수 등의 객체단위로 모형을 수정하는 명령어로서, INSERT, REMOVE, 그리고 UPDATE형이 있다. UPDATE는 REMOVE와 INSERT의 실행으로 해결이 가능하기 때문에 <표 3>에서는 주로 많이 사용될 OMML을 예시하기로 한다.

<표 3> OMML의 예

l_1	INSERT_BOT_OM(sign BOT_name)
l_2	REMOVE_BOT_OM(BOT_name)
l_3	INSERT_CONSTRAINT_M(Constraint_name)
l_4	REMOVE_CONSTRAINT_M(Constraint_name)
l_5	INSERT_IF_THEN_CONSTRAINT_M (Premise_constraint_name, Consequence_constraint_name)
l_6	REMOVE_IF_THEN_CONSTRAINT_M (Premise_constraint_name, Consequence_constraint_name)
l_7	INSERT_BOT_LC(Constraint_name, Sign, BOT_name)
l_8	REMOVE_BOT_LC(Constraint_name, BOT_name)
l_9	INSERT_BOT_RC(Constraint_name, Sign, BOT_name)
l_{10}	REMOVE_BOT_RC(Constraint_name, BOT_name)
l_{11}	INSERT_INDEX_C(Constraint_name, Index_name, Lower_bound, Upper_bound)
l_{12}	REMOVE_INDEX_C(Constraint_name, Index_name)
l_{13}	INSERT_INDEX_B(BOT_name, Index_name, Lower_bound, Upper_bound)
l_{14}	REMOVE_INDEX_B(BOT_name, Index_name)

비고 : M(model), I(index), OM(objective function of model), LC(left hand side of constraint), RC(right hand side of constraint), C(constraint), B(BOT)

INSERT_A_B는 A를 B에 추가하며, REMOVE_A_B는 B로부터 A를 삭제하는 기능을 가지고 있다. INSERT시는 관련된 하위객체가 존재하지 않으면 자동으로 추가한다(자동추가 원리). 추가되어야 할 객체가 이미 존재하면 중복되어 추가되지는 않는다(중복추가금지 원리). 한편, REMOVE시 관련된 하위객체를 다 삭제하게 되는데(삭제원리), 다른 객체가 사용하고 있으면 삭제

하지 않는다(삭제금지 원리).

4.4 모형화활동

모형화활동(action for model formulation)은 선택된 초기모형이 다른 모형으로 수정되는 정보를 가지고 있다. 다음은 M-VRP 기본모형의 경우, 모형수정요인에 의해 인식되는 모형화활동의 내용과 수학적인 표현을 나타낸 것이다.

•a_1:Remove_maximum_traveling_time_constraint

모형으로부터 (6)식을 제거한다. m_1에 의해 인식된다.

•a_2:Insert_maximum_traveling_distance_constraint

모형에 (12)식을 추가한다. m_2에 의해 인식된다.

$$\sum_{i=1}^{n+m}\sum_{j=1}^{n+m} d_{ij}X_{ijk} \le l_k \quad \text{for } k=1,2,\ldots,v \qquad (12)$$

•a_3:Insert_maximum_number_of_visiting_points_constraint

모형에 (13)식을 추가한다. m_3에 의해 인식된다.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ijk} \le u_k \quad \text{for } k=1,2,\ldots,v \qquad (13)$$

•a_4:Insert_fixed_cost_of_utilizing_vehicle_BOT

목적함수에 (14)식을 추가한다. m_4에 의해 인식된다.

$$\sum_{i=1}^{n+m}\sum_{j=1}^{n}\sum_{k=1}^{v} f_k X_{ijk} \qquad (14)$$

•a_5:Insert_earliest_delivery_time_window_constraints

모형에 (15) ~ (17)식을 추가한다. m_5에 의해 인식된다. a_{10}이 선행될 때 의미가 있다.

$$et_i^{start} \le T_{ik}^{start} \text{ for } i=n+1,\ldots,n+m,\ k=1,2,\ldots,v \qquad (15)$$

$$et_i \le T_i \text{ for } i=1,2,\ldots,n \qquad (16)$$

$$et_i^{arrival} \le T_{ik}^{arrival} \text{ for } i=n+1,\ldots,n+m,\ k=1,2,\ldots,v \qquad (17)$$

•a_6:Insert_latest_delivery_time_window_constraints

모형에 (18) ~ (20)식을 추가한다. m_6에 의해 인식된다. a_{10}이 선행될 때 의미가 있다.

$$T_{ik}^{start} \le lt_i^{start} \text{ for } i=n+1,\ldots,n+m,\ k=1,2,\ldots,v \qquad (18)$$

$$T_i \le lt_i \text{ for } i=1,2,\ldots,n \qquad (19)$$

$$T_{ik}^{arrival} \le lt_i^{arrival} \text{ for } i=n+1,\ldots,n+m,\ k=1,2,\ldots,v \qquad (20)$$

•a_7: Insert_service_time_coefficient

모형에 (24)식 결론부의 왼쪽부분(LHS)에 (21)식을 추가한다. m_7에 의해 인식된다. a_{10}이 선행될 때 의미가 있다.

$$s_i \qquad (21)$$

•a_8:Inert_penalty_of_tardy_starting_service_time_BOT

목적함수에 (22)식을 추가한다. m_8에 의해 인식된다. a_6이 선행될 때 의미가 있다.

$$\sum_{i=1}^{n+m} a(T_i - lt_i) \qquad (22)$$

a는 상수를 의미한다.

•a_9:Insert_penalty_of_tardy_routing_duration_BOT

목적함수에 (23)식을 추가한다. m_9에 의해 인식된다. a_1이 동시에 존재하면 충돌이 발생한다.

$$\sum_{i=n+1}^{n+m}\sum_{k=1}^{v} b(T_{ik}^{arrival} - T_{ik}^{start}) \qquad (23)$$

b는 상수를 의미한다.

•a_{10}:Insert_compatibility_requirement_between_routes_and_schedules_constraint

모형에 (26)식을 추가한다. a_5, a_6, 또는 a_7에 의해 인식된다.

$$\text{if } X_{ijk} \ge 1 \text{ then } T_i + t_{ij} \le T_j \quad \text{for } i=1,2,\ldots,n,n+1,\ldots,n+m,\ j=1,2,\ldots,n,n+1,\ldots,n+m,\ k=1,2,\ldots,v \qquad (24)$$

모형수정대안에 의해 인식되는 모형화활동들이 타당한 목표모형을 만들기 위해서는 두가지 관계가 존재한다. 하나는 선행관계이고, 다른 하나는 충돌관계이다. 선행관계는 모형화활동에 의해 모형이 수정되는 순서가 존재한다는 것으로 다음과 같이 표현된다.

선행관계집합

$$= \{(a_i \rightarrow a_j) | \forall i,j,\ i \ne j,\ a_i: \text{선행활동},\ a_j: \text{후행활동}\}$$

$$= \{(a_5 \rightarrow a_{10})\ (a_6 \rightarrow a_{10})\ (a_7 \rightarrow a_{10})\ (a_8 \rightarrow a_6)\} \qquad (25)$$

충돌관계는 두 활동이 동시에 존재할 때 모형화 과정에서 충돌이 생기는 경우로서, 다음과 같이 표현된다.

충돌관계집합

$$= \{(a_i, a_j) | \forall i,j,\ i \ne j\}$$

$$= \{(a_1, a_9)\} \qquad (26)$$

각 모형화활동을 OMML형태의 의미론적 표현으로 기술하면 <표 4>와 같다.

<표 4> 각 모형수정 활동(a_i)의 OMML 표현

a_1	REMOVE_INDEX_C("*maximum_traveling_time_constraint*", "*vehicle_index*") REMOVE_CONSTRAINT_M("*maximum_traveling_time_constraint*")
a_2	INSERT_CONSTRAINT_M("*maximum_traveling_distance_constraint*") INSERT_INDEX_C("*maximum_traveling_distance_constraint*", "*vehicle_index*", *1, v*)
a_3	INSERT_CONSTRAINT_M("*maximum_number_of_visiting_points_constraint*") INSERT_INDEX_C("*maximum_number_of_visiting_points_constraint*", "*vehicle_index*", *1, v*)
a_4	INSERT_BOT_OM("+", "*fixed_cost_of_utilizing_vehicle_BOT*") INSERT_INDEX_B("*fixed_cost_of_utilizing_vehicle_BOT*", "*flow_out_index*" *1 n+m)* INSERT_INDEX_B("*fixed_cost_of_utilizing_vehicle_BOT*", "*flow_in_index*" *1 n)* INSERT_INDEX_B("fixed_cost_of_utilizing_vehicle_BOT", "vehicle_index" 1 v)
a_5	INSERT_CONSTRAINT_M("*earliest_start_delivery_time_constraint*") INSERT_INDEX_C("*earliest_start_delivery_time_constraint*", "*flow_ou_indext*", *n+1, n+m*) INSERT_INDEX_C("*earliest_start_delivery_time_constraint*", "*vehicle_index*", *1, v*) INSERT_CONSTRAINT_M("*earliest_delivery_time_constraint*") INSERT_INDEX_C("*earliest_delivery_time_constraint*", "*flow_out_index*", *1, n*) INSERT_CONSTRAINT_M("*earliest_arrival_delivery_time_constraint*")

	INSERT_INDEX_C(*"earliest_arrival_delivery_time_constraint", "flow_out_index", n+1, n+m*) INSERT_INDEX_C(*"earliest_arrival_delivery_time_constraint", "vehicle_index", 1, v*)
a_6	INSERT_CONSTRAINT_M(*"latest_start_delivery_time_constraint"*) INSERT_INDEX_C(*"latest_start_delivery_time_constraint", "flow_out_index", n+1, n+m*) INSERT_INDEX_C(*"latest_start_delivery_time_constraint", "vehicle_index", 1, v*) INSERT_CONSTRAINT_M(*"latest_delivery_time_constraint"*) INSERT_INDEX_C(*"latest_delivery_time_constraint", "flow_out_index", 1, n*) INSERT_CONSTRAINT_M(*"latest_arrival_delivery_time_constraint"*) INSERT_INDEX_C(*"latest_arrival_delivery_time_constraint", "flow_out_index", n+1, n+m*) INSERT_INDEX_C("latest_arrival_delivery_time_constraint", "vehicle_index", 1, v)
a_7	INSERT_BOT_LC(*"service_time_BOT", "+", "compatibility_requirement_between_routes_and_schedule_constraint"*)
a_8	INSERT_BOT_OM("+", *"penalty_of_tardy_starting_service_time_BOT"*) INSERT_INDEX_B(*"penalty_of_tardy_starting_service_time_BOT", "flow_out_index" n n+m*)
a_9	INSERT_BOT_OM("+", *"penalty_of_tardy_routing_duration_ BOT"*) INSERT_INDEX_B(*"penalty_of_tardy_routing_duration_BOT", "flow_out_index" 1 n+m*) INSERT_INDEX_B("penalty_of_tardy_routing_duration_BOT", "vehicle_index" 1 v)
a_{10}	INSERT_IF_THEN_CONSTRAINT_M(*"visit_constraint", "compatibility_requirement_between_routes_and_ schedule_constraint"*) INSERT_INDEX_C(*"visit_constraint", flow_ out_index", 1, n+m*) INSERT_INDEX_C(*"visit_constraint", "flow_in_index", 1, n+m*) INSERT_INDEX_C(*"visit_constraint", vehicle_index", 1, n+m*) INSERT_INDEX_C(*"compatibility_requirement_between_routes_and_schedule_constraint", "flow_out_index", 1, n+m*) INSERT_INDEX_C(*"compatibility_requirement_between_routes_and_schedule_constraint", "flow_in_index", 1, n+m*) INSERT_INDEX_C(*"compatibility_requirement_between_routes_and*_schedule_constraint", "vehicle_index", 1, n+m)

4.5. 모형사례의 표현

모형은 기본모형 또는 하나 이상의 모형수정 활동으로 구성되는데, **M**(기본모형|모형화활동)으로 표기한다. 프레임 구조로 표현하면, [그림 7]과 같으며, 이 모형은 **M**(M-VRP|a_1,a_2,a_{10},a_6)을 의미한다.

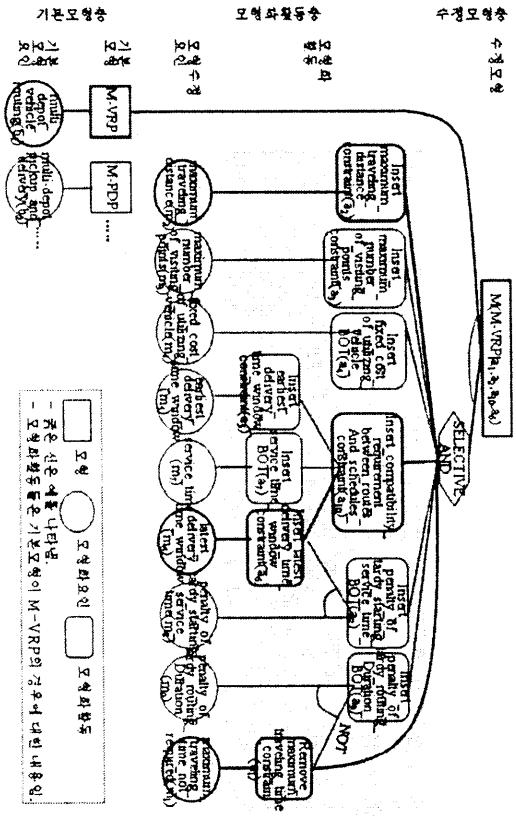
[그림 7] **M**(M-VRP|a_1,a_2,a_{10},a_6) 모형의 표현

{{ **Case_name:** *M-VRP_a1_a2_a10_a6*
Base_model: *M-VRP*
Modification_actions:
Remove_maximum_traveling_time_constraint
Insert_maximum_traveling_distance_constraint
Insert_compatibility_requirement_between_routes_and_schedules_constraint
Insert_latest_delivery_time_window_constraints }}

모형은 모정화활동에 의해 수정될 수 있다. **A**를 모형화활동을 이용하여 모형을 수정하는 모형화연산자라 하면 **M**(기본모형|a_i)**A**(a_j)는 i≠j의 경우 **M**(기본모형|a_i, a_j)가 되고, i=j의 경우 OMML의 중복추가금지원리에 의하여 **M**(기본모형|a_i)이 된다. 다. 한편, **M**(기본모형|a_i,a_j)**A**(~a_j)은 **M**(기본모형|a_i)이 된다.

4.6 모형화 규칙

[그림 8] 목표모형화를 위한 AND/OR 그래프관계



[그림 7]의 AND/OR 그래프는 목표모형을 탐색하는 과정에서 모형화요인, 기본모형, 모형화활동 사이의 관계를 보여준다. AND/OR 그래프는 기본모형요인의 값에 의해 기본모형이 선택되는 기본모형층, 모형수정요인 또는 다른 모형화활동에 의해 모형화활동이 정해지는 모형화활동층, 기본모형으로부터 모형화활동에 의해 모형이 변형되는 수정모형층의 3가지로 구분된다. 굵은 선은 하나의 예를 나타낸 것이다. 기본모형요인의 값이 b_1이고 모형수정요인의 값이 m_1, m_2, m_6이면 기본모형은 M-VRP, 모형화활동은 a_1, a_2, a_6, a_{10}이 각각 선택되어 M-VRP_a1_a2_a6_a10라는 최종 목표모형이 생

성되게 된다. 모형화 규칙은 모형화활동을 인식하기 위한 지식이다. [그림 8]에서와 같이, 기본모형이 정해지면 모형수정요인의 값에 따라 모형화활동이 식별된다. 이 AND/OR 그래프는 다음과 같은 IF-THEN 구문으로 표현 가능하다: Rule 1 ~ 7은 Rule i: IF(m_i) THEN A(a_i); Rule 8: IF($m_8 \cap a_6$) THEN A(a_8); Rule 9: IF($m_9 \cap \sim a_1$) THEN A(a_9); Rule 10: IF($a_5 \cup a_6 \cup a_7$) THEN A(a_{10}). 이 규칙과 모형화활동간 선행관계((25)식)와 충돌관계((26)식))를 제외하면, 주어진 모형수정요인은 M-VRP 기본모형을 327개 모형으로 변형 가능하다.

5. 목표모형 탐색과 모형화

5.1 모형수정 활동의 상대적 거리

모형수정은 OMML 유형과 대상 객체에 따라 모형수정 노력이 다르다. [그림 2]의 모형화 지식베이스 구조에서 상위층에 존재하는 객체는 상대적으로 많은 노력이 든다. 노력이 많이 든다는 것은 원래 모형으로부터 유사도가 더욱 작아진다는 것을 의미하며, 반대로 상대적 거리가 커진다는 것을 의미한다. <표 3>의 각 OMML이 모형에 적용시 예상되는 노력에 해당하는 상대적 거리($r(l_i)$)는 다음과 같다. $r(l_1)$과 $r(l_2)$은 6, $r(l_3)$와 $r(l_4)$은 8, $r(l_5)$과 $r(l_6)$은 15, $r(l_7)$와 $r(l_8)$은 6, $r(l_9)$과 $r(l_{10})$는 6, $r(l_{11})$과 $r(l_{12})$는 2, $r(l_{13})$와 $r(l_{14})$은 2이며, 여기서 같은 객체에 대한 INSERT와 DELETE의 OMML은 같은 노력이 든다고 가정하였으며, 이 경우 $r(\sim l_i) \cong r(l_i)$이다. $r(a_i)$는 i번째 모형화활동 적용시 예상되는 노력으로서, 모형과의 상대적 거리, 즉,

$$r(a_i) = \sum_{j, l_j \in a_i} r(l_j) \quad \forall i \qquad (27)$$

가 된다. 한편, 역 모형수정활동의 상대적 거리는

$$r(\sim a_i) \cong r(a_i) \quad \forall i \qquad (28)$$

이다. $D_{A,B}(D_{B,A})$를 모형 A와 B 사이의 상대적 거리라고 하고, $\mathbf{G} = \{a_i\}$를 기본모형에서 목표모형으로 변환하는 모형화활동의 집합이라고 하자. 그러면,

$$D_{base,goal} = \sum_{a_i \in \mathbf{G}} r(a_i) \qquad (27)$$

가 된다. 마찬가지로 $\mathbf{S} = \{a_j\}$를 목표모형에 가장 유사한 모형을 기본모형으로 변환하는 모형화활동의 집합이라고 하면,

$$D_{most\ similar,goal} = \sum_{a_j \in S} r(\sim a_j) \qquad (27)$$

가 된다.

5.2 모형수정 노력

모형수정노력($Effort_T$)은 초기모형 추출을 위한 검색노력($Effort_S$)과 목표모형으로 수정하는 노력($Effort_M$)으로 구성된다.

$$Effort_T = Effort_S + Effort_M \qquad (29)$$

$Effort_S$의 경우는, 기본모형보다는 모형사례가 많기 때문에

$$Effort_{S,base} < Effort_{S,most\ similar} \qquad (30)$$

가 된다. $Effort_{S,base}$는 새로운 목표모형을 위한 기본모형 탐색노력이며, $Effort_{S,most\ similar}$는 목표모형과 가장 유사한 모형사례를 탐색해 내는 노력을 의미한다. 그리고 $Effort_{M,base}$는 기본모형으로부터의 수정노력, $Effort_{M,most\ similar}$는 가장 유사한 모형사례로 부터의 수정노력를 의미한다. $Effort_{M,base}$는 $D_{goal,base}$에 비례하고, $Effort_{M,most\ similar}$는 $D_{goal,most\ similar}$에 비례할 것이다. $Effort_{M,base}$와 $Effort_{M,most\ similar}$는 목표모형에 따라 다르다. $D_{goal,base}$가 작은 경우는 모형수정 노력이 적게 들 것이기 때문에 기본모형으로부터 목표모형으로 변환하는 것이 바람직하고, 이 거리가 큰 경우에는 가장 유사한 모형사례를 찾아내는 탐색시간이 걸리더라도 수정 노력이 작게드는 모형이 있다면 그 모형사례로부터 목표모형으로 변환하는 것이 총 노력을 줄일 수 있을 것이다. 따라서 $D_{goal,base}$의 변화에 따른 $Effort_T$의 성능평가를 통한 초기모형 선택전략 알고리즘을 제시한다.

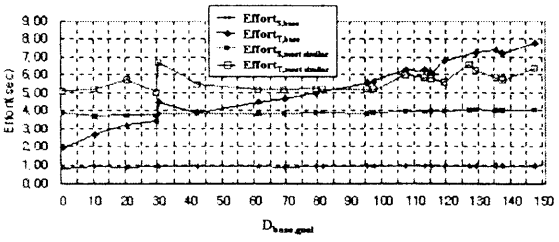
5.3 초기모형 선택을 위한 민감도 분석

민감도 분석을 위한 실험환경으로는 Pentium IV(256M RAM) 기종에 OS로는 Windows ME, DBMS로는 MS SQL Server, 그리고 시뮬레이션을 위한 프로그램으로는 JAVA를 이용하였다. 1개의 M-VRP 기본모형과 M-VRP 기반의 20개의 모형사례를 사례베이스(Case-base)에 저장하고, 하나씩 새로운 목표모형으로 가정하면서 나머지 모형과 비교하였다.

•실험1: 상대적 거리와 모형수정 효율성에 대한 민감도 분석

$D_{goal,base}$순으로 순차적으로 목표모형으로 두고, 기본모형 또는 가장 유사한 모형사례로 부터의 모형화에 걸리는 컴퓨터 실행시간을 측정 후, 효율성을 평가하기 위하여 시뮬레이션하였다. 각 $Effort_T$ 측정에는 20번 반복하여 평균치를 기록한 값이다.

[그림9] 상대적 거리와 모형수정 효율성에 대한 민감도 분석



분석: 검색노력 $Effort_{S,base}$는 $Effort_{S,most\ similar}$ 보다 작다. 모형사례가 많을수록 그 차이는 커지게 될 것이다. $Effort_{T,base}$는 $D_{goal,base}$의 값에 비례적으로 증가한다. $Effort_{T,most\ similar}$는 거의 일정하며, 모형사례가 많아질수록 이 값도 증가하게 된다. 단순회귀분석을 이용하여 두 직선의 교차점(D_T, Threshold distance)을 찾아 효율성을 평가해 보자. $Effort_{T,base}$에 대한 회귀선은

$$Effort_{T,base}(est) = \alpha_1 + \beta_1 D_{goal,base} \qquad (31)$$

이며, α_1 = 2.4023 그리고 β_1 = 0.0352로 추정된다. H_0: β_1 = 0, H_a: $\beta_1 \neq 0$에 대해 T = 22.9966 ≥ t(19, 0.025) = 2.0930이므로, 유의수준 0.05에서 기울기는 0이 아니라고 추정할 수 있다. 마찬가지로, $Effort_{T,base}$에 대한 회귀선은

$$Effort_{T,most\ similar}(est) = \alpha_2 + \beta_2 D_{goal,base} \quad (32)$$

이며, α_2 = 5.1758 그리고 β_2 = 0.0052로 추정된다. H_0: β_2 = 0, H_a: $\beta_2 \neq 0$,T = 2.2605 ≥ t(19, 0.025) = 2.0930이므로, 유의수준 0.05에서 기울기는 0이 아니라고 추정할 수 있다.

한편, 두 직선의 동일여부를 평가해 보면, H_0 : $\alpha_1 = \alpha_2$ 그리고 $\beta_1 = \beta_2$, H_a: $\alpha_1 \neq \alpha_2$ 또는/그리고 $\beta_1 \neq \beta_2$에 대하여, F = 61.1386 > F(2, 38; 0.05) = 3.3이며, H_0 : $\beta_1 - \beta_2 = 0$, H_a: : $\beta_1 - \beta_2 \neq 0$에 대하여 T = 10.8255 > t(38, 0.025) = 2.02이므로, 유의수준 0.05 두 직선은 서로 같지 않으며 기울기도 다르다고 할 수 있다. D_T 는 92.45로 추정된다.

5.4 초기모형선택 전략

실험1을 바탕으로 효율적인 탐색 알고리즘을 위한 전략을 제시한다. $D_{goal,base} \leq D_T$(short distance)의 경우 $Effort_{T,base}$ 가 더 작으므로 기본모형을 초기모형으로 선택하고(초기모형 탐색시간 감소전략), $D_{goal,base} > D_T$(long distance)의 경우 가장 유사한 모형사례를 이용하여 목표모형으로 모형화하는 것이 $Effort_{T,most\ similar}$가 더욱 효율적이다(모형수정시간 감소전략). 이 경우, 총노력 $Effort_{T,most\ similar}$는 $Effort_{S,base}$ 만큼 더 소요될 것으로 예상되나 [그림 10]에서 볼 수 있듯이 그 값은 작으므로 크게 고려하지 않아도 된다.

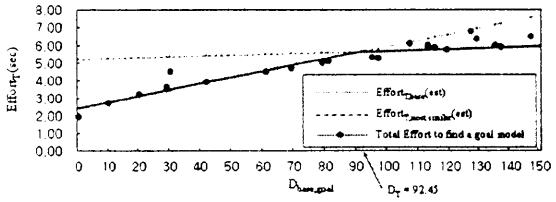
•예1: 초기모형 선택

[그림 6]에서 목표모형은 M-VRP_a1_a2_a10_a6 이므로 $D_{M\text{-}VRP_a1_a2_a10_a6,M\text{-}VRP} = r(a_1) + r(a_2) + r(a_{10}) + r(a_6) = 8 + 8 + 20 + 26 = 62$이다. 실험2 결과 $D_{M\text{-}VRP_a1_a2_a10_a6,M\text{-}VRP} = 62 \geq D_T = 92.45$이므로 M-VRP 모형을 초기모형으로 선택한다. 만일 $D_{M\text{-}VRP\_a1\_a2\_a10\_a6,M\text{-}VRP} < D_T$ 이라면, 가장 상대적 거리가 작은 모형사례를 선택한다. 예를 들어, $D_{M\text{-}VRP_a1_a2_a10_a6,\ M\text{-}VRP_a1_a2_a10_a7} = r(a6) + r(\sim a7) = 26 + 6 = 32$의 값이 모형사례들 중에서 가장 최소값이면 $M(M\text{-}VRP|a_1,a_2,a_{10},a_7)$ 모형을 선택한다.

•실험2: 초기모형 선택의 성능평가

D_T = 92로 두고, 초기모형선택 전략을 이용하여 목표모형을 탐색하는 과정에서의 효율성을 확인해 본다. 21개 모형을 각각 새로운 목표모형으로 하여 초기모형으로 선택되는 결과를 확인해 본다. 각 $Effort_T$ 측정에는 20번 반복하여 평균치를 기록한 값이다.

[그림 10] 초기모형 선택의 성능평가



분석: D_T 를 분기점으로 왼쪽영역에서는 기본모형을, 그리고 오른쪽 영역에서는 모형사례를 초기모형으로 선택하는 효율적인 검색현상을 보게 된다.

5.5 전방향 검색구조에 의한 목표모형 탐색

목표모형이 인식되고, 초기모형이 선택되면 목표모형화를 위한 경로 선택과 모형화 과정을 거치게 된다. 먼저 모형수정과 경로에 관련된 용어를 정의하면 다음과 같다.

•탐색경로(Search path): 기본모형 또는 모형사례로부터 목표모형을 탐색하는 경로로서, 하나 이상의 모형화활동으로 구성된다.

•변환성(Transformability): 기본모형 또는 모형사례로부터 타당한 목표모형으로 변환되는 경로가 하나 이상이 발견되는 경우를 의미한다.

•경로무관성(Path independence): 둘 이상의 탐색경로가 존재할 때, 어느 경로를 따르더라도 변환성이 있으면 경로무관하다고 하며, 그렇지 않을 때는 경로에 종속적(Path dependent)이라고 한다. 경로에 무관하는 것은 모형화활동의 순서는 관계없으며, 같은 목표모형으로 변환됨을 의미한다.

•예2: 목표모형 탐색경로

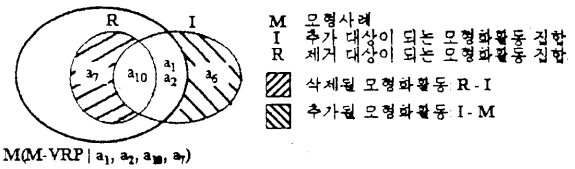
예1에서 목표모형은 M-VRP 기본모형에 $\mathbf{A}(a_1)$, $\mathbf{A}(a_2)$, $\mathbf{A}(a_6)$, $\mathbf{A}(a_{10})$의 4개 모형화활동에 의해 수정된다. 4개의 모형화활동의 순서에 따라 다수의 탐색경로가 발견되는데, 선행관계에 의해서 $\mathbf{A}(a_{10})$은 $\mathbf{A}(a_6)$에 선행되어야 하기 때문에 경로 종속적이다. 이 선행조건을 만족하는 경로들은 변환성이 있으며, 이 경로의 총 숫자는 $12개(3P_3+2_2P_2+_2P_2)$이다.

경로종속적이거나 탐색경로가 여러 개 있을 경우, 모형의 변환성이 보장되며 효율적인 경로를 찾는 것이 중요한데, 두 가지 관점에서 고려되어야 한다. 첫째, 모형화활동의 순서는 INSERT와 DELETE의 OMML 순서에 영향을 주고, 이 순서는 모형화에 영향을 미치기 때문에 두 활동간 항상 교환 가능한 것이 아니다. 모형화활동간 선행관계가 있을 경우에는 선행관계규칙을 따르면 된다. 둘째, [그림 11]에서 처럼 유사한 모형사례(**M**), 제거될 모형수정활동 집합(**R**), 그리고 새롭게 추가될 모형수정활동 집합(**I**)이 있는 경우에,

$$\mathbf{R} \cap \mathbf{I} \neq \phi \text{ 이면 } \mathbf{MA(I)A(R)} \neq \mathbf{MA(R)A(I)} \quad (33)$$

즉, 교환가능하지 않은 상황이 발생한다. 이 경우 **A(I-M)**와 **A(R-I)**는 경로무관하며, 목표모형의 모형화가 가능하다.

[그림 11] 목표모형화를 위한 모형화활동



목표모형화의 경로상에는 초기모형에 추가활동 및 삭제활동의 두 과정이 있으나, 각 활동간에 선행관계가 존재하지 않으면 경로에 무관하기 때문에 순서에는 관계가 없다. 목표모형을 탐색하는 경로선택 단계는 다음과 같다.

- •1단계: 삭제활동 선택(**R-I**)
- •2단계: 추가활동 선택(**I-M**)
- •3단계: 모형화활동간 충돌관계 확인
- •4단계: 모형화활동간 선행관계 확인

•예3: 탐색경로 발견

예2에서 목표모형을 **M**(M-VRP|a_1,a_2,a_{10},a_6), 초기모형을 **M**(M-VRP|a_1,a_2,a_{10},a_7)라고 가정해 보자. [그림 11]에서 **R**={a_7 a_{10}}, **I**={a_1 a_2 a_{10} a_7}이 된다. (35)식에 의하여 **R**과 **I**는 교환 가능하지 않다. **R-I** ={a_7}, **I-M**= {a_6} 이고, 이 들간에는 충돌 및 선행관계가 없기 때문에, 경로에 무관하며 변환성이 있다. 즉,

M(M-VRP|a_1,a_2,a_{10},a_7)**A**(a_6)**A**(~a_7)
= **M**(M-VRP| a_1,a_2,a_{10},a_7)**A**(~a_7)**A**(a_6) (34)

가 된다.

5.6 경로에 따른 모형의 수정

경로가 설정되면 모형화 작업에 들어가게 된다. UNIK-OPT는 의미론적인 초기모형으로부터 의미론적인 목표모형으로 변환하고, IP solver에 맞는 수학적인 최적화 모형으로 변환한다. 예를들어 모형의 수정과정을 보기로 한다.

•예4: 최적화모형의 모형화

예3의 결과에서 에서 경로 1을 **M**(M-VRP | a1 a2 a10 a7)**A**(a6)**A**(~a7), 경로 2를 **M**(M-VRP | a1 a2 a10 a7)**A**(~a7)**A**(a6)라고 하자. [그림 12]은 경로 1의 결과를 나타낸 것이다. 경로 1도 같은 목표모형에 도달하게 된다.

[그림 12] **M**(M-VRP|a_1 a_2 a_{10} a_7)**A**(a_6)**A**(~a_7)

```
{{M-VRP_a1_a2_a10_a6
  IS-A : IP_MODEL
  DIRECTION : min
  OBJECTIVE : total_traveling_distance_BOT
  CONSTRAINT : drop_in_constraint
    drop_out_constraint
    route_continuity_constraint
    vehicle_capacity_constraint
    vehicle_start_constraint
    vehicle_arrival_constraint
    subtour_breaking_constraint
    maximum_traveling_distance_constraint
```

(1) a_6에 의해 추가된 결과

```
IF_THEN("visit_constraint", "compatibility_
    requirement_between_routes_and_schedules_
    constraint")
  latest_delivery_time_window_constraint
  latest_start_delivery_time_window_constraint
  latest_arrival_delivery_time_window_constraint }}
  .....

{{latest_delivery_time_window_constraint
  IS-A : CONSTRAINT
  OPERATOR : LE
  LHS : depot_time_BOT
  RHS : latest_time_BOT
  UNIT_INDEX: (flow_out_index 1 n)          }}
  .....

{{ compatibility_requirement_between_routes_and_
schedules_constraint
  IS-A : CONSTRAINT
  OPERATOR : LE
  LHS : (+traveling_time_toward_visit_BOT)
  RHS : (+traveling_time_BOT)
  UNIT_INDEX: (flow_out_index 1 n+m)
       (flow_in_index 1 n+m)
       (vehicle_index 1 v)                  }}
  .....
```

(2) ~a7에 의해 service_time_BOT 가 삭제된 결과

6. 결론

본 연구는 분산환경에서 에이전트가 사례기반의 자동모형화에 의한 효율적인 문제해결을 제시하는 프레임웍을 제시하였다. 최적화 에이전트의 자동 모형화에 관한 지식을 표현하였으며, 민감도분석에 의해 목표모형으로 모형화되는 노력이 최소화되는 초기모형 선택 알고리즘, 전방향 추론에 기반을 둔 탐색경로를 제시하였다. 분산환경에서 사용자들의 의사결정 형태의 다양성과 신속성이 요구가 커질수록 문제해결을 위한 자동화된 에이전트의 연구는 더욱 필요하게 될 것이다. 프로토타입 및 실제 구현에 필요한 에이전트의 구조 및 기능을 함께 제시하면서, VRP 모형을 이용한 배송계획의 예를 중심으로 그 유용성을 보였다. 그러나, IP는 문제의 범위가 커지면 모형화 노력보다는 보다는 문제를 풀기 위한 시간이 많이 걸린다. 큰 문제 또는 시간제약적인 상황에서 해가 필요할 때는 휴리스틱 방법이나 또는 유전알고리즘(Genetic algorithm)을 이용하여 문제를 푸는 것이 효율적일 수 있다. 불가능해가 존재하는 경우 에이전트시스템간의 협력에 의해 문제를 해결하거나, 문제해결에 걸리는 시간과 부하를 고려하여 상호간 효율적 의사결정을 지원하기 위한 시간제약적인 협상 프로토콜[14] 기반의 접근, 또는 Extensible Rule Markup Language(XRML)[11]에

기반한 문제요구를 해결하려는 시도는 흥미있는 연구가 될 것이다.

참고문헌

[1] Binbasioglu, M., "Process-based Reconstructive Approach to Model Building", Decision Support Systems 12, 1994, pp.97-113.

[2] Dumas, Y., Desrosiers, J. and Soumis, F., "The pickup and delivery problem with time windows", European Journal of Operational Research, 54, 1991, pp.7-22.

[3] Elam, J. J., Henderson, J. C. and Miller, L. W., Model Management Systems: An Approach to Decision Support in Complex Organizations", Proceedings of the First International Conference on Information System, 1980 pp.98-110.

[4] Gentner, D., "Analogical inference and analogical access", Analogica, 1988, pp. 63-68. CA: Kaufmann.

[5] Ishikawa, T. and Terano, T., "Analogy by Abstraction: Case Retrieval and Adaptation for Inventive Design Expert Systems", Expert systems with applications, Vol. 10, No. ¾, 1996, pp.351-356.

[6] Kalakota,R., Stallaert,J. and Whinston, A.B., "Implementing Real-Time Supply Chain Optimization Systems", in Global Supply Chain and Technology Management, H.L.Lee and S.M.Ng Ed. Production and Operations Management Society, 1998. pp.60-75.

[7] Kedar-Cabelli, S.T., "Toward a computational model of purpose-directed analogy". Analogica, 1988, pp.89-108. CA: Kaufmann.

[8] Kim, C.S., "Communication Network Design and Analysis Expert Systems Based on Higher Level Representation", Expert Systems With Application, Vol. 13, No. 2, 1997, pp.127-143.

[9] Kuhn, Y. and Lee,J.K., "Logical Representation of Integer Programming Models", Decision Support Systems 18, 1996, pp.227-251.

[10] Kulkarni, R.V. and Bhave, P.R., "Interger Programming Formulations of Vehicle Routing Problems", European Journal of Operational Research , Vol 34, pp 403-404

[11] Lee, J.K., "Artificial Intelligence in Electronic Commerce", Plenary Speech at Pacific Rim International Conference on Artificial Intelligence, Melbourine, www.icec.net, August 2000.

[12] Lee, J.K. and Kim, M.Y., "Knowledge-Assisted Optimization Model Formulation: UNIK-OPT", Decision Support System 13, 1995.

[13] Lee,J.K. and Lee,B.Y., "Integrated Management System for Optimization and Heuristic Models", Working paper, 2001, KAIST.

[14] Lee, K.J., Chang,Y.S. and Lee,J.K., "Time-Bounded Negotiation Framework for Electronic Commerce Agents", Decision Support Systems, vol. 28, number 4, June 2000, pp.319-331.

[15] Lee,K.J. et al., "Negotiation and Decision Making of Virtual Manufacturing Agent under Time-bounded Environment", International Conference on Electronic Commerce, 2001.

[16] Liang,T.P., "Modeling by analogy: A Case-based approach to automated linear program formulation", IEEE, 1991, pp.276-283.

[17] Liang,T.P., "Analogical reasoning and case-based learning in model management systems", Decision support systems, 10, 1993, pp.137-160.

[18]Liang,T.P. and Konsynski,B.R., "Modeling by Analogy: Use of analogical reasoning in model management systems", Decision support system, 9, 1993, pp.113-125.

[19] Sadeh,N., Hildum,D. and Kjenstad,D, "Intelligent e-Supply Chain Decision Support", in Proc. International Conference on Electronic Commerce, 2000, pp.124-132.

[20] Winston, W.L., Operations Research, 3rd ed. Duxbury Press, 1994.