웹 기반 시스템에서 효과적 모델관리를 위한 메타지식

김철수*

Meta Knowledge for Effective Model Management in Web-based System

Chulsoo Kim*

Abstract

Diverse requirements of users on web-based model management force a system agent to develop user-adaptive ability building a model in reality and providing an adequate solution method of the model. The relationship between models is important knowledge for the agent to effectively build a new model, to adaptively adjust an existing model under a problem, and to efficiently connect the new model into an adequate solution method. Since the generating process of the inter-model relationship is more difficult than the building a new model, however, the process mostly depends on the knowledge of operation research experts. Without the adequate scheme of the inter-model relationship, the burden of the management for the agent increases rapidly, and the quality of the services may worsen. This study shows that meta-knowledge generated from relationships between models is important for the user to build a model in reality and to acquire the solver appropriate to the model.

The relationship that consists of common and exclusive objects between models can be represented by frames. The system under development to implement the idea includes user-adaptive ability, which identifies a model through forward chaining method and searches the solver appropriate to the model by using the meta knowledge. We illustrate the meta knowledge with an applied delivery system in supply chain management.

Key Word: web-based system, model management, system agent, meta knowledge

* 인하대학교 경영학부 조교수

1. 서 론

전자상거래 시장의 전체 매출액의 80% 이상이 기업과 기업간의 거래에서 이루어지게 되는 21세기에서는 한 기업이 가치사슬(Value Chain) 관계인 다른 기업과의 다양한 정보교류를 통해서 얻게되는 부가가치는 한계를 짐작하기 어려울 정도로 클 것이다. 기업간 전자상거래 확산에 따른 다양하고 복잡한 문제를 해결하는 모델에 대한 수요도 증대되는 반면에 모델표현 및 모델관리에 초점을 맞추었던 기존의 모델관리(Model Management) 분야도 웹 환경에서의 모델관리라는 관점으로 확대·발전시켜야 한다. 이러한 점에서 시급히 해결되어야할 몇 가지를 정리해 보자. 첫째로 기업과 기업들이 연결된 가치사슬에서는 의사결정에 사용되는 대부분의 모델베이스, 데이터베이스, 지식베이스 등은 유사성을 갖게 되는데, 예를 들면 공급체인(Supply Chain)을 이루는 기업에서 배달을 역할로 하는 기업들의 의사결정에 사용되는 문제들은 네트워크 특성을 갖는 것이 대다수이다. 유사성을 갖지만 모델들이 많기 때문에 효율적인 방법으로 모델베이스를 관리하지 못하면 시스템 저장능력이 충분하더라도 모델 요구가 있을 시에 필요한 모델을 적시에 찾기 힘들며 유사한 모델들을 약간 수정을 통해서 사용할 수 있는데도 불구하고 처음부터 새로 구축하는 비 낭비적인 프로세스를 갖는 경우가 태반이다. 둘째는 모델을 구축하는 작업도 중요하지만 그 모델에 맞게 해법을 추출하는 것은 더욱더 중요하다. 각 모델마다의 해법이 인덱싱이 되어 있더라도 그 해법이 시스템에 존재하지 않는 경우는 다른 시스템에서 해법을 제공받든지 유사한 해법이 존재하는지를 규명해서 추출해 내야한다. 이러한 해법을 추출하는 두 가지 문제를 해결하기 위해서는 다른 시스템으로부터 모델 제공을 요청하는 메시지를 생성해야하며 모델들 간의 유사성을 고려한 상하관계성을 나타내는 메타지식을 마련해야 한다.

본 연구에서는 네트워크 흐름문제들을 모델베이스로 갖고 있는 모델관리시스템을 공급체인관리(SCM: Supply Chain Management) 분야의 배달시스템에 적용하는데 있어서 중요한 과제는 사용자가 표현한 배달문제가 여러 부분문제로 이루어지고 그 부분문제들에 적합한 해법을 자동으로 추출해주기 위해서는 모델간의 관계성 지식이 시스템에 내장되어 있어서 이를 이용해서 해법 추출에 이용하기 위한 해법제공 지능에이전트 개발을 위한 이론적인 근거를 마련하는 것이 본 연구의 목적이다.

위의 목적을 달성하기 위해서 본 연구에서는 네트워크 흐름문제를 중심으로 다음과 같은 이론을 연구하고자 한다.

- 모델의 의미론적 표현
- 모델 정의를 위한 추론방법론
- 모델간 상하관계성 생성 및 지식표현
- 해법추출 규칙베이스

본 연구에서의 전반적인 지식표현은 UNIK-OPT의 표현방식을 사용했으며, 모든 문제는 프레임 형태인 문제 도메인 지식(Problem Domain Knowledges)으로 저장된다 [Lee et al. 1995]. 논문의 구성을 보면 2장에서는 모델관리 분야에서 지식기반시스템의 역할과 UNIK 하에서 개발된 지식기반시스템들의 주요기능을 설명한다. 3장에서는 네트워크 모델에 대한 문제도메인 지식에 의한 의미론적인 표현을 설명한다. 그리고 네트워크 흐름문제와 해법과의 관계를 설명하면서 네트워크 모델의 정의를 위한 추론방법과 그 정의과정을 4장에서 제시한다. 이 장에서는 해

법추출에 관한 두 가지 과정을 설명하며 모델들 간의 유사성을 고려한 상하관계성을 생성한다. 5장에서는 지식기반시스템에서 지능에이전트와 메시지를 주고받는 하부 에이전트들의 기능에 대해서 설명하고 관련된 지식들을 제안한다. 6장에서는 앞장까지 설명한 이론을 배달시스템의 사례를 대상으로 적용하며 배달문제에 관한 모델정의와 해법추출을 도와주는 지식의 우수성을 보인다.

2. UNIK 기반 모델관리시스템

모델관리 분야에서 최적화 모델과 지식기반시스템을 연계한 연구가 많은데, 그 중에서 본 논문은 모델 표현의 기반이 되고 있는 UNIK (UNIfied Knowledge) 하에서 개발된 지식기반 시스템들에 관해서 설명한다.

첫째는 UNIK-OPT(Knowledge-based Modeling System in Linear Programming and Integer Programming)가 있는데, 이 시스템은 모델 표현을 의미론적인 관점, 모델링 언어 관점, 수리적 표기 관점, 테이블 관점 등과 같이 네 가지 관점으로 기술하고 있다. 여기서 의미론적인 관점은 어튜리뷰트(Attribute), BOT(Blocks Of Terms) 그리고 제약식 간의 연관성은 제약네트워크(Constraint Network)를 통해서 가능하며 특정의 선형 및 정수 계획 모델을 구축하는 추론화 과정을 위해서 공통지식베이스(Common Modeling Knowledge Base)를 사용한다. 공통지식베이스는 모델링 구조 지식베이스(Modeling Structure Knowledge Base)와 도메인 지식베이스(Domain Knowledge Base)로 구성되는데 전자는 모델링 요소들(Modeling Constructs) 간의 연관성을 파악하기 위해서 사용되는 반면에, 후자는 비전문적 사용자와의 상호작용을 통해서 자신의 문제를 명확히 하는 과정을 도와주는데 사용한다. 이 도메인 지식베이스는 프레임 구조로 표현되며, 각 프레임 지식베이스 들은 NETWORK_MODEL, CONSTRAINT, BOT, ATTRIBUTE, INDEX 등으로 구성된다. 그리고 이 시스템은 지식획득과 유지를 위해서 사례기반 학습(Case-based Learning) 접근법을 사용하고 있다 [Lee and Kim, 1995].

둘째는, UNIK-IP(Knowledge-based System for Higher Level Representation of Integer Programming Models)가 있는데, 보통 정수계획모델은 수학적으로 표현된 모델의 관점에서 보면 변수가 이진 정수이거나 일반적인 양의 정수라는 것 외에는 선형계획 모델과 크게 다르지가 않다. 그러나 지식으로 표현된 구조 속에서 그러한 정수계획 모델의 내용을 해석해 내는 과정은 쉽지가 않으며 거기에다가 정수계획 모델은 변수나 제약조건 들간의 복잡한 논리적 관계를 나타내기 위한 의사결정 상황을 담고 있어서 이에 대한 논리적인 프로세스를 지식으로 담아서 사용자가 상위수준의 문제 표현이 가능하도록 이 시스템은 도와주고 있다.

셋째는, UNIK-RELAX(Intelligent System for Structural Identification and Relaxation in Integer Programming Model)가 있는데, 이 시스템에서 다루는 정수계획모델은 많은 응용분야를 갖고 있으며 응용성이 높을수록 비다항식 형태(NP-hard)가 두드러져서 푸는데 많은 어려움이 있고 이를 위해서 대부분의 문제를 휴리스틱 해법을 이용하게 된다. 이 시스템의 기본 개념은 모델의 부분 구조를 정의하고 목적식에 제약식을 완화하여 새로운 모델을 생성시켜서 해를 구하게

되는데 잘 알려진 라그랑지안 릴렉세이션 방법론(Lagrangian Relaxation Method)을 적용하여 의사결정자들의 현실적이지만 복잡한 문제에 대한 부담을 덜어주고 있다 [Kim and Lee, 1996].

이 시스템에서는 정수계획 모델을 일곱 가지의 내포구조(Embedded Structure)로 구분하는데 그것은 0-1배낭 구조(0-1 knapsack), 일반 상한한계 구조(generalized upper bound), 변수 상한한계 구조(variable upper bound), 하나 걸침나무 구조(one spanning tree), 동적계획 모델의 속성을 갖는 배낭 구조(knapsack with Dynamic Model), 일반적인 정수계획 구조 (General Integer Programming) 등이다. 사용자는 정수계획 모델시스템 UNIK-OPT를 이용해서 모델을 네 가지 객체를 갖는 의미론적인 관점으로 표현하고, 네 가지 객체인 변수 및 계수, BOT, 제약식, 목적함수 들은 특성치(Distinctiveness)를 갖는다. 표현된 정수계획 모델에서 내포구조를 정의하게 되는 추론과정이 필요한데 여기에는 정방향 추론방식, 역방향 추론방식과 경험적 추론방식 등을 이용하고 있다. 내포구조가 정의되면 라그랑지안 모델로 완화시켜주는 변환과정이 이루어지는데, 이 과정은 먼저 제약식 객체를 발견하고, 그 제약식에 관계된 라그랑지안 승수를 새로운 객체로 생성되며, 그 승수에 의해서 새로운 BOT를 생성한다. 그 후에 목적식에 승수와 변형된 제약식이 결합하여 더해지게 된다. 그리고 완화된 객체와 라그랑지안 모델에 이용되지 않는 항과 계수 객체들은 삭제된다.

끝으로, UNIK-PMA(Knowledge-assisted System for Unification of Linear Programming with a Rule-based System by the Post-model Analysis Approach)는 규칙기반 모델과 최적화 모델을 통합한 모델을 다루며, 최적화 모델의 목적함수와 규칙베이스에 표현된 목표치를 상호조정(Tradeoffs)할 수 있게 해준다. 이 통합 모델은 양쪽의 목적들을 포괄하는 일종의 다목적 의사결정(Multiple Objective Decision Making: MODM) 모델이며 여기서는 MODM(1, M) 모델로 공식화하였다. 통합모델을 풀기 위해서 사후모델분석 접근방법을 채택하였으며, 비지배(Non- dominated)된 해를 구해주기 위한 구조로 세 가지 형태의 상호조정을 제안했다. UNIK-PMA는 위와 같은 기능을 수행하기 위해서 다음과 같은 지식과 기능을 갖는다. 최적화 모델의 의미론적인 표현, 관련 규칙베이스 구축 및 추출, 규칙베이스 차원에서의 최적화 모델의 최적해 평가, 자동 모델 생성에 의한 비지배 상호조정 과정 제공과 규칙베이스 목적으로부터의 제약식 생성 등이다 [Lee and Song, 1995].

3. 네트워크 모델의 의미론적 표현

네트워크 문제의 도메인 지식(Domain Knowledge)은 UNIK-OPT에서 다룰 수 있도록 프레임 구조를 갖으며 설명을 위해서 네트워크 흐름 문제에서 자주 인용되는 단순최소비용흐름문제(Pure Minimum Cost Flow Problem)를 표현하면 아래와 같다. 여기서는 각 객체 중에 하나씩만 예를 들었으며, 전체표현은 <부록 1>을 참조한다.

```
{{ Pure_Minimum-Cost_Flow_Problem    모델객체의 예
   IS-A : NETWORK_MODEL
   DIRECTION : min
   OBJECTIVE : cost_flow_BOT
   CONSTRAINT : amount_of_source_constraint(flow=real,
                with_gain=one)
                amount_of_destination_constraint(flow=r
                eal, with_gain=one)
```

amount_of_transfer_constraint(0,
with_gain=one)
lower_bounded_flow_constraint
upper_bounded_constraint
DISTINCTIVENESS : Pure_Minimum-Cost_Flow_Structure
SOLVER : }}

{{ amount_of_source_constraint(flow=real, with_gain=one)
IS-A : CONSTRAINT 제약식 객체의 예
OPERATOR : EQ
LHS : (+flow_out_BOT) (-flow_in_BOT)
RHS : (+requirement_BOT)
UNIT_INDEX : source_index
DISTINCTIVENESS : amount_of_source_constraint }}

{{ cost_flow_BOTBOT 객체의 예
IS-A : BOT
ATTRIBUTE : cost_unit_flow
SUMMATION_INDEX : flow_out_index flow_in_index
DISTINCTIVENESS : cost_flow_BOT }}

{{ flow_variable 변수객체(ATTRIBUTE에 속함)의 예
IS-A : VARIABLE
SYMBOL :
LINKED_INDEX : flow_out_index flow_in_index
DISTINCTIVENESS : decision_variable}}

{{ cost_unit_flow 상수객체(ATTRIBUTE에 속함)의 예
IS-A : COEFFICIENT
SYMBOL :
LINKED_INDEX : flow_out_index flow_in_index
DISTINCTIVENESS : constant}}
{{ flow_out_index 인텍스 객체의 예
IS-A : INDEX
SYMBOL :
LINKED_ATTRIBUTE :
DISTINCTIVENESS : network _index }}

4. 네트워크 모델 정의와 해법추출

4.1. 네트워크 흐름문제와 해법

이 장에서는 경영과학분야에서 잘 알려진 네트워크 흐름문제와 그 문제를 푸는데 잘 알려진 휴리스틱 해법을 정리하였다.

위의 문제를 정의하기 위해서는 문제가 가지고 있는 고유한 특성을 나타낼 수 있어야하는데, 그것을 표현하면 아래와 같다.

- AMOUNT_OF_SOURCE_CONSTRAINT(FLOW=CONSTANT,TRANSFER=EXIST,WITH_GAIN =ONE)
- AMOUNT_OF_DESTINATION_CONSTRAINT (FLOW=CONSTANT,TRANSFER=EXIST,WITH_GAIN=ONE)
- AMOUNT_OF_TRANSFER_CONSTRAINT(FLOW=CONSTANT,TRANSFER=EXIST,WITH_GAIN =ONE)
- BOUNDED_CONSTRAINT(LOWER=REAL, UPPER=REAL)

위의 표현들은 노드의 흐름 방출량에 관한 제약, 노드의 흐름 유입량에 관한 제약, 중간 노드의 흐름량 보존에 관한 제약이다. 그리고 네 번째의 표현은 노드마다의 흐름양에 최대 및 최소 제약을 나타낸다. 위의 특성치는 앞에서 설명한 문제 도메인 지식 표현으로 다시 재 생성될 수 있다. "Conservation"이란 특성을 나타내는 표현을 문제 도메인 지식으로 표현하면 다음과 같다.

{{ conservation_constraint(flow=constant/real, transfer=exist/not_exist, with_gain=one/real)
IS-COMPOSED-OF :

amount_of_source_constraint (flow=constant/real, with_gain=one/real)
amount_of_destination_constraint (flow=constant/real, with_gain=one/real)
mount_of_transfer_constraint (0, with_gain=one/real)
DISTINCTIVENESS : conservation }}

다음과 같은 프레임으로 변환된다.

{{ amount_of_source_constraint (flow=constant/real, with_gain=one/real)
IS-A : CONSTRAINT
OPERATOR : EQ
LHS : (+flow_out_BOT) (-flow_in_BOT_[with_gain=one/real])
RHS : (+requirement_BOT [flow=constant/real])
UNIT_INDEX : source_index
DISTINCTIVENESS : amount_of_source_constraint }}
{{ amount_of_destination_constraint (flow=constant/real, with_gain=one/real)
IS-A : CONSTRAINT
OPERATOR : EQ
LHS : (+flow_out_BOT) (-flow_in_BOT_[with_gain=one/real])
RHS : (-requirement_BOT [flow=constant/real])
UNIT_INDEX : destination_index
DISTINCTIVENESS : amount_of_destination_constraint }}
{{ amount_of_transfer_constraint (transfer=exist/not_exist, with_gain=one/real)
IS-A : CONSTRAINT
OPERATOR : EQ
LHS : (+flow_out_BOT) (-flow_in_BOT_[with_gain=one/real])
RHS : (0)
UNIT_INDEX : transfer_index
DISTINCTIVENESS : amount_of_transfer }}
{{ flow_in_BOT_[with_gain=one/real]
IS-A : BOT
ATTRIBUTE : [with_gain=one/real] flow_variable
SUMMATION_INDEX : flow_in_index
DISTINCTIVENESS flow_in_BOT }}
{{ flow_out_BOT_[with_gain=one/real])
IS-A : BOT
ATTRIBUTE : [with_gain=one/real] flow_variable
SUMMATION_INDEX : flow_in_index
DISTINCTIVENESS flow_in_BOT }}
{{ requirement_BOT [flow=constant/real]
IS-A : BOT
ATTRIBUTE : flow=constant/real
SUMMATION_INDEX :
DISTINCTIVENESS : requirement_BOT }}

이러한 특성들은 네트워크 흐름문제들을 구분하는데 중요한 요소들이며, 이것을 가지고 상향식과 하향식 인식방법의 규칙들(Rules)을 생성한다.

4.2. 네트워크 모델 정의

이 절에서는 네트워크 모델의 정의과정을 설명하는데 그것은 상향식 방법(Bottom-up Approach), 하향식 방법(Top-down Approach), 그리고 사례기반 방법(Case-based Approach)이다.

상향식 방법은 INDEX, VARIABLE과 COEFFICIENT, BOT, CONSTRAINT, 그리고 MODEL 순으로 각 객체의 특성(Distinctiveness)을 정의해 가는 추론방법을 사용하는데 이 방법을 후방향 추

〔표 1〕 네트워크 흐름문제와 해법

네트워크 흐름모델	해법	입력자료	출력자료
할당문제	Hungarian	비용행렬	할당행렬
수송문제	Vogel	비용행렬, 수요량, 공급량	총수송비용, 수송계획
다단계 수송문제	Kilter	비용행렬, 수요량, 공급량	총수송비용, 수송계획
최단거리문제	Dijkstar	네트워크 구조, 시점, 종점	최단거리, 최단거리경로
최단걸침나무문제	Prim	네트워크 구조	최단걸침나무의 형태
최대흐름문제	Ford	네트워크 구조, 시점, 종점, 흐름량 상한, 흐름량 하한	최대흐름양, 흐름경로
단순최소비용 흐름문제	Kilter	네트워크 구조, 시점, 종점, 비용행렬, 흐름량	최대흐름양, 흐름경로
비보존 최소비용흐름문제	Chris	흐름변동계수, 네트워크 구조, 시점, 종점, 비용행렬, 흐름량	최대흐름양, 흐름경로

론기법(Backward Chainning Method)으로 할 수 있도록 UNIK-BWD(UNIfied Knowledge-BackWarD)로 진행해 나간다. 이 방법은 시스템에 저장된 모델들의 수가 많을 때에 유용한 방법이다. 하향식 방법은 MODEL, CONSTRAINT, BOT, VARIABLE과 COEFFICIENT, INDEX 순으로 이어지는 추론방법으로 시스템에서는 전방향 추론기법(Forward Chainning Method)에 의해서 진행되며 UNIK-FWD(UNIfied Knowledge-ForWarD)와 연동하여 정의과정이 진행된다. 이 방법은 저장된 모델들의 수가 적고 객체 수가 많게되면 규칙의 층수(Number of Level)가 비례적으로 많아지므로 그 때에는 이 방법을 사용하는 것이 유리하다. 마지막 방법인 사례기반 인식방법은 하향식 방법이나 상향식 방법이 우선 수행되어 많은 종류의 네트워크 흐름문제들이 시스템에 저장된 상태에서 유사한 문제에 대해 정의과정이 필요할 때에 사용할 수 있다.

4.3. 모델들간의 상하관계 및 해법추출

해법을 추출하는 과정은 규칙베이스를 이용해서 해법을 직접 찾는 과정이 있으며, 해법이 시스템에 존재하지 않는 경우에 모델들간의 상하관계로 생성된 지식베이스를 이용하여 다른 차선의 해법을 찾는 과정으로 구분된다. 첫 번째가 직접적인 해법 추출과정이라 할 수 있는데, 앞 절에서 설명한 [표 1]의 내용을 담고있는 내용을 이용해서 적합한 해법을 추출한다 [Lee, 1995]. 다음은 네트워크 흐름문제에 따라 해법을 정의하는 규칙베이스의 형식이다.

```
(RULE Connection_of_Network_Problem_A_into_Solving_Method_B
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF      (IS DISTINCTIVENESS OF NETWORK_MODEL
        ('DISTINCTIVENESS 'Network_Problem_A))
THEN    (IS SOLVER 'Solving_Method_B))
```

두 번째의 간접적인 과정은 모델들 사이의 상하관계를 통해서 이루어진다. 상하관계는 네트워크 문제에서 토폴로직 특성치를 이용하는데, 토폴로직 특성치는 다수의 파라미터들을 갖는데 파라미터들이 갖는 집합에 따라 상하관계가 성립된다. 하위관계의 토폴로지 특성치의 파라미터가 갖는 집합은 상위관계의 파라미터가 갖는

집합을 부분집합으로 갖는다. 이러한 상하관계는 초기에는 경영계량 문제의 전문가가 제공하고 이것을 계속적으로 지식기반 시스템이 상하관계를 추가하거나 변경해나가는데 이 연구에서는 상하관계 자동생성과정을 돕는 조사에이전트(Investigator Agent)의 역할이 중요한데 여기서는 생략하기로 한다. 상하관계 표현구조를 보면 다음과 같다.

{{ A_model
IS-A-RELATIONSHIP-WITH : B_model
INHERITANCE :
inheritance_1
inheritance_2
inheritance_3 . . . . }}

위의 형식이 의미하는 내용은 B_model을 표현한 프레임 객체가 있을 때에 inheritance_1, inheritance_2, inheritance_3, … 등과 같은 상속값을 가지고 변형만 시켜주면 A_model을 구할 수가 있다. 예를 들어 설명하면, 앞에서 설명한 다단계 수송문제와 중간 경유지가 없는 수송문제(Transportation Problem)를 보면 다음과 같은 상하관계 형식으로 표현할 수가 있다.

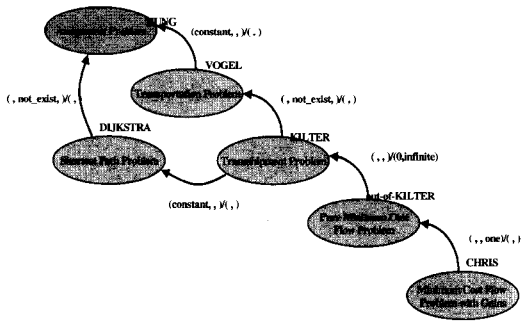
{{ Tranportation_problem
IS-A-RELATIONSHIP-WITH : Transshipment_problem
INHERITANCE :
conservation_constraint (flow, transfer=not_exist, with_gain) }}

네트워크 흐름 문제들 중에서 이러한 상하관계가 많이 나타나는데 몇 가지 모델에 대해서 위의 형식을 가지고 표현해 보자. 네트워크 흐름문제 중에서 할당문제(Assignment Problem), 수송문제(Transportation Problem), 다단계 수송문제, 단순 최소비용흐름문제, 비보존 최소비용흐름문제(Minimum Cost Flow Problem with Gains)에 대한 상하관계를 표현하면 다음과 같다.

{{ Assignment_problem
IS-A-RELATIONSHIP-WITH : Transportation_problem
INHERITANCE :
conservation_constraint (flow=one, transfer, with_gain) }}
{{ Transportation_problem
IS-A-RELATIONSHIP-WITH : Transshipment_problem
INHERITANCE :
conservation_constraint (flow, transfer=not_exist, with_gain) }}
{{ Transshipment_problem
IS-A-RELATIONSHIP-WITH : Pure_minimum_cost_flow_problem
INHERITANCE :
lower_bound (lower_limit_value=zero)
upper_bound (upper_limit_value=infinite) }}
{{ Pure_minimum_cost_flow_problem
IS-A-RELATIONSHIP-WITH:Minimum_cost_flow_with_gains_problem
INHERITANCE :
conservation_constraint (flow, transfer, with_gain =real) }}

네트워크 흐름문제의 상하관계를 나타내면 [그림 1]과 같다.

해법을 찾는 간접적인 과정은 사용자에 의해서 표현되어진 모델이 수송문제라 했을 때에 그것의 알고리즘은 VOGEL 방법을 사용하면 되는데 그 방법이 시스템에서 활용가능하지 않을 때에는 그 모델 아래 단계인 KILTER, out-of-KILTER, CHRIS 순으로 가능한 해법기를 찾아주어 해법기가 여러 웹사이트에 분산된 환경에서는 이러한 상하관계에 의한 해법연결과정이 중요한 의미를 지닌다.



〔그림 1〕 네트워크 흐름문제들간의 상하관계 예

5. 해법제공 지능 에이전트

5.1 주요 하부시스템

본 연구에서 개발하고자 하는 해법제공 지능 에이전트는 세 가지의 중요한 하부 에이전트와 실시간으로 정보를 교류하고 있다.

① 의미론적 모델 구축자(Semantic Model Builder) 사용자로부터 문제에 대한 정보를 입력받는다. 이 구축자는 UNIK-OPT에서 사용되는 모델구축 지식베이스를 이용하고 있으며, 모델의 지식구조도 동일하다. 단 네트워크의 상위수준의 표현을 UNIK-OPT에 작동 가능한 표현수준으로 변환시킨다.

② 모델구조 정의자(Model Structural Identifier) 네트워크 흐름문제를 토폴로직 제약식으로 표현하고 이미 저장된 규칙베이스를 추론하여 구축된 문제의 구조를 판단하여 모델을 정의한다.

③ 모델-해법기 연결자(Connector of a Model into a Heuristic)
모델의 구조가 모델구조 정의자에 의해서 정의되면 모델-해법기 연결자는 직접적 연결규칙과 간접적 연결규칙들을 가지고 추론을 통해서 적합한 해법기를 찾아 그 해법기의 입력양식에 맞게 데이터를 정리한다.

5.2 해법추출을 위한 규칙베이스

사용자가 대화식으로 문제에 관한 정보를 입력하면 의미론적 모델 구축자는 3장에서 표현한 방식으로 상위수준의 표현을 한다. 다시 이것을 UNIK-OPT에서 작동할 수 있는 기저수준의 표현으로 변환하게 된다 [Kim, 1997].

3장에서 설명한 UNIK-OPT 표현은 인식추론 과정에서 사실베이스(Factbase)가 되며, 이것을 가지고 규칙베이스에 따라 모델구조 식별자는 흐름문제를 인식하게 된다. 이 논문에서는 규칙베이스를 UNIK-BWD 형식으로 구축하였으며 그 규칙베이스를 설명하면 다음과 같다.

```
(RULE pure_minimum_cost_flow_structure_rule
[RULE_GROUP MODEL]
IF    (IS CONSTRAINT.DISTINCTIVENESS 'amount_of_source_constraint)
      (IS CONSTRAINT.DISTINCTIVENESS 'amount_of_destination_constraint)
      (IS CONSTRAINT.DISTINCTIVENESS 'amount_of_transfer_constraint)
      (IS CONSTRAINT.DISTINCTIVENESS 'lower_bounded_flow_constraint)
      (IS CONSTRAINT.DISTINCTIVENESS 'upper_bounded_flow_constraint)
THEN  (IS DISTINCTIVENESS 'pure_minimum_cost_flow_structure))

(RULE amount_of_source_constraint_rule
[RULE_GROUP CONSTRAINT]
IF    (IS CONSTRAINT.LHS '(+flow_out_BOT) (-flow_out_BOT))
      (IS CONSTRAINT.RHS 'requirement_BOT)
      (IS CONSTRAINT.UNIT_INDEX.DISTINCTIVENESS 'source_index)
THEN  (IS CONSTRAINT.DISTINCTIVENESS 'amount_of_source_constraint))
(RULE amount_of_destination_constraint_rule
[RULE_GROUP CONSTRAINT]
IF    (IS CONSTRAINT.LHS '(+flow_out_BOT) (-flow_out_BOT))
      (IS CONSTRAINT.RHS ' - requirement_BOT)
      (IS CONSTRAINT.UNIT_INDEX.DISTINCTIVENESS 'destination_index)
```

```
THEN  (IS CONSTRAINT.DISTINCTIVENESS 'amount_of_destination_constraint))
(RULE amount_of_transfer_constraint_rule
[RULE_GROUP CONSTRAINT]
IF    (IS CONSTRAINT.LHS '(+flow_out_BOT) (-flow_out_BOT))
      (IS CONSTRAINT.RHS '0)
      (IS CONSTRAINT.UNIT_INDEX.DISTINCTIVENESS 'transfer_index)
THEN  (IS CONSTRAINT.DISTINCTIVENESS 'amount_of_transfer_constraint))

(RULE lower_bounded_constraint_rule
[RULE_GROUP CONSTRAINT]
IF    (IS CONSTRAINT.OPERATOR 'GE)
      (IS CONSTRAINT.RHS 'lower_bounded_BOT)
THEN  (IS CONSTRAINT.DISTINCTIVENESS 'lower_bounded_constraint))
(RULE upper_bounded_constraint_rule
[RULE_GROUP CONSTRAINT]
IF    (IS CONSTRAINT.OPERATOR 'LE)
      (IS CONSTRAINT.RHS 'upper_bounded_BOT)
THEN  (IS CONSTRAINT.DISTINCTIVENESS 'upper_bounded_constraint))
```

모델구조 정의자에 의해서 모델의 구조가 정의되면 모델-해법기 연결자는 직접적인 과정과 간접적인 과정으로 해법기를 찾아서 해법기의 입력형식에 맞추어서 모델에 관한 자료를 넘겨준다. 두 과정은 UNIK-BWD를 사용한 후방향 추론기법을 활용한다. 아래 표현은 모델-해법기를 직접 연결하기 위한 규칙베이스이다.

```
(RULE Connection_into_Hungarian_Method
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF    (IS DISTINCTIVENESS OF NETWORK_MODEL
        ('DISTINCTIVENESS 'Assignment_Problem_ Structure))
      (AND (EXIST 'Hungarian_Method))
THEN  (IS SOLVER 'Hungarian_Method))
(RULE Connection_into_Vogel_Method
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF    (IS DISTINCTIVENESS OF NETWORK_MODEL
        ('DISTINCTIVENESS 'Transportation_Problem_Structure))
      (AND (EXIST 'Vogel_Method))
THEN  (IS SOLVER 'Vogel_Method))
(RULE Connection_into_Kilter_1_Method
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF    (IS DISTINCTIVENESS OF NETWORK_MODEL
        ('DISTINCTIVENESS 'Transshipment_Problem_Structure))
      (AND (EXIST 'Kilter_Method))
THEN  (IS SOLVER 'Kilter_Method))
(RULE Connection_into_Dijkstar_Method
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF    (IS DISTINCTIVENESS OF NETWORK_MODEL
        ('DISTINCTIVENESS 'Shortest_Path_Problem_Structure))
      (AND (EXIST 'Dijkstar_Method))
THEN  (IS SOLVER 'Dijkstar_Method))
(RULE Connection_into_Prim_Method
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF    (IS DISTINCTIVENESS OF NETWORK_MODEL
        ('DISTINCTIVENESS 'Minimum_SpanningTree_Problem_Structure))
      (AND (EXIST 'Prim_Method))
THEN  (IS SOLVER 'Prim_Method))
(RULE Connection_into_Ford_Method
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF    (IS DISTINCTIVENESS OF NETWORK_MODEL
        ('DISTINCTIVENESS 'Maximum_Flow_Problem_Structure))
      (AND (EXIST 'Ford_Method))
THEN  (IS SOLVER 'Ford_Method))
(RULE Connection_into_Kilter_2_Method
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF    (IS DISTINCTIVENESS OF NETWORK_MODEL
        ('DISTINCTIVENESS 'Pure_Minimum_Cost_Flow_Problem_
        Structure))
      (AND (EXIST 'Kilter_Method))
THEN  (IS SOLVER 'Kilter_Method))
(RULE Connection_into_Chris_Method
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF    (IS DISTINCTIVENESS OF NETWORK_MODEL
        ('DISTINCTIVENESS'Minimum_Cost_Flow_Problem_with_Gains_
        Problem_Structure))
      (AND (EXIST 'Chris_Method))
```

```
THEN    (IS SOLVER 'Chris_Method))
```

아래 지식 표현은 모델과 해법기 연결에서 해법기가 존재하지 않는 경우에 모델들간의 상하 관계를 고려해서 하위 모델의 해법기를 이용하는 규칙베이스를 나타낸다.

```
(RULE Hungarian_Method_not_Existed
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF      (IS DISTINCTIVENESS OF NETWORK_MODEL
            ('DISTINCTIVENESS 'Assignment_Problem_Structure))
        (AND (NOT_EXIST 'Hungarian_Method))
THEN    (Connection_into_Vogel_Method))
(RULE Connection_into_Vogel_Method_not_Existed
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF      (IS DISTINCTIVENESS OF NETWORK_MODEL
            ('DISTINCTIVENESS 'Transportation_Problem_Structure))
        (AND (NOT_EXIST 'Vogel_Method))
THEN    (Connection_into_Kilter_1_Method))
(RULE Connection_into_Kilter_1_Method_not_Existed
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF      (IS DISTINCTIVENESS OF NETWORK_MODEL
            ('DISTINCTIVENESS 'Transshipment_Problem_Structure))
        (AND (NOT_EXIST 'Kilter_Method))
THEN    (Connection_into_Dijkstar_Method))
(RULE Connection_into_Dijkstar_Method_not_Existed
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF      (IS DISTINCTIVENESS OF NETWORK_MODEL
            ('DISTINCTIVENESS 'Shortest_Path_Problem_Structure))
        (AND (NOT_EXIST 'Dijkstar_Method))
THEN    (Connection_into_Prim_Method))
(RULE Connection_into_Prim_Method_not_Existed
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF      (IS DISTINCTIVENESS OF NETWORK_MODEL
            ('DISTINCTIVENESS 'Minimum_SpanningTree_Problem_Structure))
        (AND (NOT_EXIST 'Prim_Method))
THEN    (Connection_into_Ford_Method))
(RULE Connection_into_Ford_Method_not_Existed
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF      (IS DISTINCTIVENESS OF NETWORK_MODEL
            ('DISTINCTIVENESS 'Maximum_Flow_Problem_Structure))
        (AND (NOT_EXIST 'Ford_Method))
THEN    (Connection_into_Kilter_2_Method))
(RULE Connection_into_Kilter_2_Method_not_Existed
[RULE_GROUP NETWORK_FLOW_PROBLEM]
IF      (IS DISTINCTIVENESS OF NETWORK_MODEL
            ('DISTINCTIVENESS' Pure_Minimum_Cost_Flow_Problem_
             Structure))
        (AND (NOT_EXIST 'Kilter_Method))
THEN    (Connection_into_Chris_Method))
```

6. 프로토타입 개발 및 활용 예

앞에서 설명한 이론을 가지고 공급체인관리에서 중요한 부분인 배달시스템에 적용해 보고자 한다. 여기서는 배달시스템에서 자주 일어나는 예제를 가지고 5장에서 설명한 의미론적 모델 구축자(SMB), 모델구조 정의자(MSI), 모델-해법기 연결자(CMH) 등의 기능을 설명하기로 한다.

배달시스템에서 고객들은 컴퓨터를 구성하는 부품들을 구입하고자 공급회사에게 필요한 부품에 대해서 공급여부를 묻게 되는데 공급회사는 일단 창고에 그 부품이 있는지 아니면 생산을 해야 되는지를 점검한 후에 공급 가능한 부품들에 대해서는 고객이 요구하는 배달예정시간까지 배달해 줄 수 있는지를 배달회사들에게 묻는다. 배달시스템에서는 고객과 공급회사, 공급회사와 배달회사 사이에서는 기업간 인터넷 기반에서 메시지를 주고받는 것을 가정하여 공급여부, 생산여부, 배달여부 등을 위한 메시지를 전송해야 하는데 이를 위한 협의 프로토콜(Negotiation Protocol)에 따른 메시지 프로토콜과 내용이 결정되어야 한다. 배달시스템에서는 객체(기업) 간에 통신되는 메시지를 해석하여 문제를 모델링

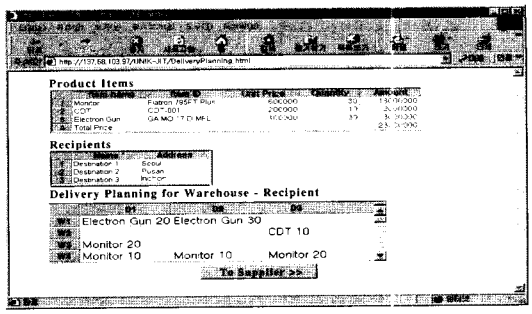
하는 이슈와 문제해결로부터 나온 결과를 다시 메시지로 변환시키는 이슈는 중요하고 복잡한 논리가 필요하여 여기서는 언급하지 않기로 한다 [Kim et al., 2000].

고객으로부터 들어온 메시지로부터 배달회사는 메시지 해석기(Message Interpreter)를 통해서 내용을 해석하고 필요한 내용을 SMB에게 넘겨준다.

SMB는 전달받은 내용을 가지고 배달계획을 수립하기 위해서 일반적인 세 가지 형태의 문제로 구분하는데 즉, 수송문제, 운송차량 선정, 차량경로 선정 등인데 이러한 문제에서 처음에 수송문제에 관한 내용을 정리해서 다종제품(Multi-commodity) 수송문제를 추출해 낸다. 그리고 SMB는 구축된 수송모델을 가지고 문제에 대한 인식과정과 문제-해법 연결과정을 진행할 수 있도록 MSI에게 모델 정보를 넘겨준다.

여기서 MSI는 UNIK-BWD와 연동하여 문제의 모델을 정의하고, 이 과정에서 토폴로직 특성치가 존재하면 그 특성치의 형태와 파라미터에 따로 문제 정의를 하고 그 특성치로 표현이 안된 문제에 대해서는 5장에서 설명한 규칙베이스를 추론을 하게된다. 보통 많이 사용하는 상향식 추론방법은 ATTRIBUTE, BOT, CONSTRAINT, NETWORK_MODEL의 순서로 프레임의 DISTINCTIVENESS를 진행해 나가는 추론기법이다 [Kim et al. 1996]. 위의 문제에서는 Transportation_Structure로 정의되어 이 내용을 NETWORK_MODEL 프레임의 DISTINCTIVENESS 슬롯에 기록한다.

문제가 정의되면 CMH는 NETWORK_MODEL 프레임의 DISTINCTIVENESS의 슬롯값인 Transportation_Structure를 가지고 5장에서 제시한 규칙베이스를 가지고 해법을 찾는다. 추론결과는 Vogel-Method가 결정되는데 이것이 배달시스템내의 서버에 존재하는지를 점검하여 존재하면 SOLVER 슬롯에



〔그림 2〕 시스템을 통해서 생성하는 배달계획 결과 예

그 값을 채운다. 존재하지 않는 경우에는 Kilter-Method, Dijkstar-Method 순서로 차선의 해법을 찾아서 NETWORK_MODEL 프레임의 SOLVER 슬롯에 그 내용을 기록한다. 위의 문제에서는 Vogel-Method가 시스템에 존재하여 그 값을 기록하였다. CNH는 배달시스템의 문제를 Vogel-Method의 입력형태에 맞게 데이터를 넘겨준다. 문제 해결기를 통해서 얻게되는 해는 다음의 문제인 운송차량 선정, 차량경로 선정 문제 등의 입력데이터로 들어가서 [그림 2]와 같은 결과를 출력하는데 내용은 배달해야할 제품내역, 배달지, 그리고 창고로부터 배달지에 배달 제품 및 수량들이다.

배달시스템에서는 메시지 생성기(Message Generator)가 수립된 배달계획 내용을 가지고 공급시스템에 넘겨줄 메시지를 생성하고 그 메시지를 전송한다.

7. 결론 및 향후연구

본 연구는 전자상거래 물류시스템에서 부딪히게 되는 수배송 문제를 해결한다든지 통신망을 구축하여 네트워크를 분석한다든지 하는 경우에 이를 해결하기 위해서는 문제마다의 모델이 필요하게 된다. 지식기반시스템은 표현된 문제를

정의하고 그에 적합한 해법기를 제공하게 되는데 이를 위해서는 다수의 모델과 해법기가 시스템 내에 보관되어 있어야 한다. 다수의 모델을 중복 없이 저장하고 효율적으로 관리하기 위해서는 모델들간의 관계를 표시하는 메타 지식베이스가 필요하다. 이는 해법기의 효율적인 제공에도 중요한 지식이기도 하다.

본 연구에서는 네트워크 모델을 인식하는 과정을 세 가지 접근방식을 제안하였고, 추론과정을 거쳐서 모델과 해법기를 연결시키는 지식베이스와 해법기의 효율적 관리를 위한 모델간 관계를 표현하는 메타지식베이스도 제안하였다. 여기서 제시한 이론적인 근거를 가지고 배달시스템의 프로토타입을 개발하였고, 그 시스템을 이용해서 기업과 기업사이에서 발생하는 다양한 상거래에서 특히 배달계획을 수립하는 것을 보였으며 이 시스템이 기업간 전자상거래에서 활용가치를 더욱더 높일 것으로 판단된다.

이러한 지식기반 시스템을 웹기반으로 확장하기 위해서는 인텔리전트 에이전트를 구축하는 것이 필수적이다. 기업간 배달시스템에서는 공급시스템, 고객시스템, 그리고 재고시스템 각각에 에이전트가 있어서 다른 시스템의 에이전트들과 메시지를 주고받는 통신 제어기(Communication Controller)와 메시지를 해석, 문제 해결, 그리고 메시지 생성을 담당하는 문제해법기(Problem Solver)를 갖게 된다.

웹기반 하에서는 수많은 해법이 한 사이트에만 있지 않고 분산되어 있는 경우가 많은데, 이때에는 지식기반 해법추출시스템이 필요한 해법을 추출하기 위해서 다른 웹기반 MMS를 갖는 에이전트에 메시지를 보내어 해법을 제공받을 수 있다. 이렇게 되기 위해서는 먼저 본 논문에서 연구된 해법추출시스템의 이론적 근거가 MMS를 확장시키는데 중요한 역할을 하게 된다.

참 고 문 헌

[1] Bhargava, H. K., and R. Krishnan, Computer-aided Model Construction, *Decision Support Systems* 9(1), (1993).

[2] Bisschop, J. J. and C. A. C. Kuip, Hierarchical Sets in Mathematical Programming Modeling Languages, *Computational Optimization and Applications*, Vol 1, No 4, (1994).

[3] Binbasioglu, M., and M. Jarke, Domain Specific DSS Tools for Knowledge-based Model Building, *Decision Support Systems* 2(3), (1986).

[4] Dincvas, M., H. Simonis, and P. Van Hentenryck, Solving Large Combinatorial Problem in Logic Programming, J. *of Logic Programming*, 8, (1990).

[5] Fourer, R., Modeling Languages versus Matrix Generators for Linear Programming, *ACM Transactions on Mathematical Software* 9, (1983).

[6] Geoffrion, A. M., The SML Language for Structured Modeling, *Operations Research*, 40(1) (1992).

[7] Glover, F., D. Klingman and N. V. Phillips, Netform Modeling and Applications, *Interface* 20(4), (1990).

[8] Kendrick, D. A., A Graphical Interface for Production and Transportation System Modeling: PTS, *Computer Science in Economics and Management*, 4, (1991).

[9] Klein, Michel and Lief B. Methlie, Expert Systems: A Decision Support Approach with Applications in Management and Finance, Addison-Wesley Publishing Co., (1990).

[10] Kim, Chulsoo, and Kyu J. Lim, An

Agent-based Model Management Framework for Providing a Solution Method, *Proceedings of Informs-Korms,* (2000).

[11] Kim, Chulsoo and Jae K. Lee, Automatic Structural Identification and Relaxation for Integer Programming, *Decision Support Systems* 18 (3&4), (1996).

[12] Lee, H. On Knowledge-based Modeler for Network Analysis, Korean Management *Science Review* 12(3) (1995).

[13] Lee, Jae K. and M. Y. Kim, Knowledge-Assisted Optimization Model Formulation: UNIK- OPT, *Decision Support Systems* 13, (1995).

[14] Lee, Jae K. and Y. U. Song, Unification of Linear Programming with a Rule-based System by the Post-model Analysis Approach, *Management Science 41(5)* (1995).

[15] Ma, P., F. H. Murphy, and E. A. Stohr, A Graphics Interface for Linear Programming, *Communications of the ACM* 32(8), (1989).

[16] Steiger, D., and R. Sharda and B. Leclaire, Graphical Interfaces for Network Modeling: A Model Management System Perspective, *ORSA Journal on Computing* 5, (1993).

[17] Stohr, E., and M. R. Tanniru, A Database for Operations Research Models, *Int. J. Policy Anal. Infor. Syst.* 4, (1980).

[18] Tsai, Yao-Chuan, Model Integration using SML, *Decision Support Systems* 22, (1998).

[19] Yeom, K and Jae K. Lee, Higher Level Representation of Integer Programming Models, *Decision Support Systems* 18(3&4) (1996).

〔부록〕 문제 도메인 지식 표현

{{ Pure_Minimum-Cost_Flow_Problem
IS-A : NETWORK_MODEL
DIRECTION : min
OBJECTIVE : cost_flow
CONSTRAINT : amount_of_source_constraint(flow=real, with_gain=one)
amount_of_destination_constraint(flow =real, with_gain=one)
amount_of_transfer_constraint(0, with_gain=one)
lower_bounded_flow_constraint
upper_bounded_constraint
DISTINCTIVENESS :
Pure_Minimum-Cost_Flow_Structure
SOLVER : }}

{{ amount_of_source_constraint(flow=real, with_gain=one)
IS-A : CONSTRAINT
OPERATOR : EQ
LHS : (+flow_out_BOT) (-flow_in_BOT)
RHS : (+requirement_BOT)
UNIT_INDEX : source_index
DISTINCTIVENESS : amount_of_source_constraint }}

{{ amount_of_destination_constraint(flow=real, with_gain=one)
IS-A : CONSTRAINT
OPERATOR : EQ
LHS : (+flow_out_BOT) (-flow_in_BOT)
RHS : (-requirement_BOT)
UNIT_INDEX : destination_index
DISTINCTIVENESS : amount_of_destination_constraint }}

{{ amount_of_transfer_constraint(0, with_gain=one)
IS-A : CONSTRAINT
OPERATOR : EQ
LHS : (+flow_out_BOT) (-flow_in_BOT)
RHS : (0)
UNIT_INDEX : transfer_index
DISTINCTIVENESS : transfer_amount_constraint }}

{{ lower_bounded_flow_constraint
IS-A : CONSTRAINT
OPERATOR : GE
LHS : (+flow_BOT)
RHS : (+lower_bounded_BOT)
UNIT_INDEX : flow_out_index flow_in_ondex
DISTINCTIVENESS : lower_bounded_flow_constraint }}

{{ upper_bounded_flow_constraint
IS-A : CONSTRAINT
OPERATOR : LE
LHS : (+flow_BOT)
RHS : (+upper_bounded_BOT)
UNIT_INDEX : flow_out_index flow_in_ondex
DISTINCTIVENESS : upper_bounded_flow_constraint }}

{{ cost_flow_BOT
IS-A : BOT
ATTRIBUTE : cost_unit_flow
SUMMATION_INDEX : flow_out_index flow_in_index
DISTINCTIVENESS : cost_flow_BOT }}

{{ flow_out_BOT
IS-A : BOT
ATTRIBUTE : one flow_variable
SUMMATION_INDEX : flow_out_index
DISTINCTIVENESS flow_out_BOT }}

{{ flow_in_BOT
IS-A : BOT
ATTRIBUTE : one flow_variable
SUMMATION_INDEX : flow_in_index
DISTINCTIVENESS : flow_in_BOT }}

{{ requirement_BOT
IS-A : BOT
ATTRIBUTE : requirement
SUMMATION_INDEX :
DISTINCTIVENESS : requirement_BOT }}

{{ flow_BOT
IS-A : BOT
ATTRIBUTE : one flow_variable
SUMMATION_INDEX :

DISTINCTIVENESS : flow_BOT }}
{{ lower_bounded_BOT
IS-A : BOT
ATTRIBUTE : lower_bounded
SUMMATION_INDEX :
DISTINCTIVENESS : lower_bounded_BOT }}
{{ upper_bounded_BOT
IS-A : BOT
ATTRIBUTE : upper_bounded
SUMMATION_INDEX :
DISTINCTIVENESS : upper_bounded_BOT }}
{{ flow_variable
IS-A : VARIABLE
SYMBOL :
LINKED_INDEX : flow_out_index flow_in_index
DISTINCTIVENESS : decision_variable}}
{{ cost_unit_flow
IS-A : COEFFICIENT
SYMBOL :
LINKED_INDEX : flow_out_index flow_in_index
DISTINCTIVENESS : constant}}
{{ requirement
IS-A : COEFFICIENT
SYMBOL :
LINKED_INDEX : flow_out_index
DISTINCTIVENESS : constant }}
{{ one
IS-A : COEFFICIENT
SYMBOL :
LINKED_INDEX :
DISTINCTIVENESS : constant }}
{{ lower_bounded
IS-A : COEFFICIENT
SYMBOL :
LINKED_INDEX : flow_out_index flow_in_index
DISTINCTIVENESS : constant }}
{{ upper_bounded
IS-A : COEFFICIENT
SYMBOL :
LINKED_INDEX : flow_out_index flow_in_index
DISTINCTIVENESS : constant }}
{{ flow_out_index
IS-A : INDEX
SYMBOL :
LINKED_ATTRIBUTE :
DISTINCTIVENESS : network _index }}
{{ flow_in_index
IS-A : INDEX
SYMBOL :
LINKED_ATTRIBUTE :
DISTINCTIVENESS : network _index }}
{{ source_index
IS-A : INDEX
SYMBOL :
LINKED_ATTRIBUTE :
DISTINCTIVENESS : source_index }}
{{ destination_index
IS-A : INDEX
SYMBOL :
LINKED_ATTRIBUTE :
DISTINCTIVENESS : destination_index }}
{{ transfer_index
IS-A : INDEX
SYMBOL :
LINKED_ATTRIBUTE :
DISTINCTIVENESS : transfer_index }}