화물운송 영역에서의 부분순서 계획법과 그래프 계획법에 대한 실험적 비교

이상기* · 정용규** · 김인철***

Empirical Comparisons between Partial-Order Planning and Graph Planning in Freight Transportation Domain

Sang-Gi Lee*, Young-Gyu Jeong**, In-Cheol Kim***

요 약

본 논문에서는 응용범위가 넓고 비교적 복잡도가 높은 화물운송 계획문제들을 대상으로 몇 가지 실험을 통해 대표적인 인공지능 계획방식인 부분순서 계획법과 그래프 계획법의 성능을 비교 분석하였다. 또 동시에 이러한 실험을 통해 DVO 및 LPVO와 같은 대표적인 제어전략들을 중심으로 이들이 그래프 계획법의 성능에 미치는 효과를 비교 분석하여 보았다. 본 연구의 실험을 통해서는 부분순서 계획법에 비해 그래프 계획법이 메모리 사용량이나 CPU 계산시간 면에서 월등히 우수한 성능을 보여주었으며 비교적 복잡도가 큰 계획문제에서도 좋은 결과를 보여주었다. 하지만 도출된 해 계획의 질적인 면에서는 부분순서 계획법이 대부분 최적의 해를 찾아낸 것에 반해 그래프 계획법은 사용된 제어전략과 최적화 방법에 따라 해 계획의 질이 크게 달라질 수 있음을 보였다. 한편 그래프 계획법에서는 부속목표 선택 전략인 DVO는 그 효과를 뚜렷이 보이지 못한 반면 동작 선택 전략인 LPVO는 도출된 해 계획의 질적인 면이나 계산속도 면에서 모두 뛰어난 효과를 보여주었다.

Keywords: 화물운송계획, 부분순서 계획법, 그래프 계획법

1. 서론

일반적으로 인공지능(artificial intelligence)의 계획(planning)분야는 작업(task)이나 목표(goal)를 효과적으로 달성하기 위한 일련의 동작(actions)들을 자동으로 구하는 방법에 관해 주로 연구한다. 계획시스템의 대표적인 응용분야로는 작업공정 계획과 스케쥴링, 로보틱스, 물류 계획, 자동운항, 무인우주선 작업제어, 천체망원경 제어, 전쟁 및 전투 모의게임, 지능형 에이전트 등이 있으며 계획문제(planning problem)로 모델링 가능한 분야라면 어디에나 적용 가능하므로 응용분야가 매우 광범위하다. 대표적인 계획수립 방법으로는 선형 계획법(linear planning), 부분순서 계획법(partial-order planning), 계층적 계획법(hierarchical planning), 그래프 계획법(graph planning), SAT기반계획법(SAT-based planning)등이 있으며 또한 분산환경을 가정하는 분산계획법(distributed planning)과 멀티에이전트 계획법(multi-agent planning) 등이 있다. 최근에는 계획수립(plan generation)뿐만 아니라 계획실행(plan execution)과 재계획(replanning)에 관한 관심이 높아져 소위 적응형 계획(adaptive planning), 반응형

* 경기대학교 전자계산학과 석사과정
** 한국복합물류주식회사 개발부
*** 경기대학교 전자계산학과 조교수

계획(reactive planning)에 관한 연구가 활발하다. 본 논문에서는 응용범위가 넓고 비교적 복잡도가 높은 화물운송 계획문제들을 대상으로 몇 가지 실험을 통해 대표적인 인공지능 계획방식인 부분순서 계획법과 그래프 계획법의 성능을 비교 분석하고자 한다. 또 동시에 이러한 실험을 통해 부속목표선택 전략의 하나인 DVO 와 동작선택 전략의 하나인 LPVO 등과 같은 그래프 계획법의 대표적인 제어전략들을 중심으로 이들이 그래프 계획법의 성능에 미치는 효과를 비교 분석하고자 한다. 제 2절에서는 부분순서 계획법과 그래프 계획법에 관해 간략히 소개하고, 제 3절에서는 시행하고자 하는 실험의 목표와 방법을 중심으로 실험계획을 설명하고, 제 4절에서는 실제 화물운송 계획문제들을 풀어본 결과들을 소개하고 이들을 토대로 두 계획법간의 비교와 제어전략들간의 비교를 수행한다. 끝으로 제 5절에서는 결론과 향후 연구에 대해 논의한다.

2. 부분순서계획법과 그래프계획법

2.1 부분순서계획법

부분순서 계획법(partial-order planning)은 최소결정 전략(least commitment strategy)을 사용하는 계획 방식으로서, 계획을 이루는 동작(action)들에 대한 부분순서(partial ordering)와 동작들에 포함된 변수들의 바인딩(binding)에 대한 부분제약(partial constraint)을 허용하고 계획수립 과정 동안 꼭 필요한 제약들만을 점진적으로 부가해 나간다[4, 11]. 부분순서 계획시스템(partial-order planning system)들은 선형 계획시스템(linear planning system)들이나 완전순서 계획시스템(total-order planning system)들에 비해 계획을 이루는 동작들의 순서나 변수들의 바인딩에 관해 성급한 임의 결정들을 미리 내릴 필요가 없으므로 잘못된 결정으로 인한 후진(backtracking) 발생을 감소시킬 수 있다. 또한 계획수립 과정동안 미완성 계획을 반드시 한 쪽 방향으로만 선형적으로 확장하는 선형 계획시스템이나 완전 순서 계획시스템들과는 달리 임의의 위치에서 임의의 방향으로 계획 확장이 가능하므로 전통적인 부속목표 상호작용 문제(subgoal interaction problem)를 피해 성공적인 계획을 수립하기가 용이하다는 장점이 있다.

부분순서 계획법에서는 계획수립 과정동안 한 미완성 계획을 하나의 (S, O, B, G, L, T)로 표현한다[2]. 이때 S는 이 계획에 포함된 스텝[2](step)들의 집합을, O는 스텝들의 순서 제약(ordering constraint) 집합을, B는 변수들의 바인딩 제약(binding constraint) 집합을 나타낸다. G는 현재 미달성 목표(open goal) 집합을 나타내며 이는 계획내의 각 스텝의 전제조건들 중에서 아직 이 조건을 달성해주는 인과링크(causal link)를 하나도 갖지 못한 것들을 의미한다. L은 계획 내에 존재하는 인과링크들의 집합을 나타낸다. 하나의 인과링크는 하나의 목표(goal) 또는 부속목표(subgoal)를 의미하는 계획내의 한 스텝의 한 전제조건을 어떤 다른 스텝이 이것을 달성해주는지를 기록하는 역할을 수행한다. 스텝 sj의 한 전제조건인 p를 스텝 si가 적용효과로 달성해주면, 이때 이 인과링크를 si $\xrightarrow{p}$ sj 로 표시한다. T는 계획 내에 존재하는 위협(threat)들의 집합이다. 한편 계획내의 또 다른 스텝 sk가 si와 sj 사이에 수행되면서 ¬p로 바운드될 수 있는 한 리터럴 q를 적용효과로 생성하거나 삭제할 가능성이 있는 경우, 스텝 sk는 이미 달성해놓은 인과링크 si $\xrightarrow{p}$ sj 에 대한 하나의 위협(threat)이 된다.

부분순서 계획법에서는 [표 1]과 같이 시작 스텝과 종료 스텝으로만 이루어진 하나의 미완성 계획으로부터 출발해 미달성 목표와 위협이 모두 없는 하나의 완전한 계획을 얻을 때까지 다음의 두 가지 처리과정을 비결정적 순서(non-deterministic order)에 따라 계속해서 반복함으로써 미완성 계획을 확장해간다. 첫 번째 처리과정은 미달성 목표를 하나 선택해 이것을 성취할 수 있도록 스텝을 새로이 삽입하거나 혹은 계획내의 기존 스텝을 찾아 인과링크를 하나 생성하는 것이고, 두 번째 처리과정은 계획 내에 존재하는 위협들 중에 하나를 택해 별도의 순서제약이나 바인딩 제약을 부가함으로써 이것을 제거하는 것이다. 따라서 기존의 선형 계획법이나 완전순서 계획법에 의한 계획수립 과정은 월드상태공간(world-state space)상의 한 탐색(search)으로 볼 수 있지만, 반면에 부분순서 계획법에 의한 계획수립 과정은 계획상태공간(plan-state space) 상의 한 탐색으로 볼 수 있다.

2.2 그래프 계획법

Blum과 Furst[3]에 의해 제안된 그래프 계획법은 그래프 확장(graph expansion)과 해 도출(solution extraction)이라고 하는 두 단계를 거듭하면서 진행되어진다. 그래프 확장 단계에서는 하나의 계획 그래프(planning graph)를 계획이 존재하기 위한 필요조건(necessary condition)을 만족할 때까지 시간의 흐름에 따라 전향적으로 확장한다. 그후, 해 도출 단계에서는 계획 그래프 위에서 후향연결 탐색(backward-chaining search)을 시행하며 주어진 문제의 해가 되는 한 계획을 찾아간다. 만약 해를 찾지 못하면 계획 그래프를 더 확장함으로써 위의 주기(cycle)을 반복한다.

계획 그래프는 명제 노드(proposition node)와 동작 노드(action node)등 두 가지 유형의 노드를 포함하며

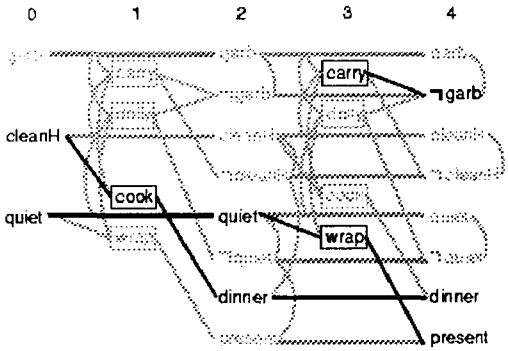
2) 부분순서 계획법에서는 동작(action)과 스텝(step)을 같은 의미로 사용한다.

[표 1] 부분순서 계획 알고리즘

Algorithm POP(S,O,B,G,L,T)

1. **Termiation :** If G and T are empty, report success and stop.
2. **Declobbering :** A step s_k threatens a causal link $s_i \xrightarrow{p} s_j$ when it occurs between s_i and s_j, and it adds or deletes p. If there exists a threat $t \in T$ such that t is a threat between a step s_k and a causal link $s_i \xrightarrow{p} s_j \in L$, then:
   - ° Remove the threat using promotion, demotion or separation.
     Promotion : $O' = O \cup \{s_k < s_i\}$, $B' = B$
     Demotion : $O' = O \cup \{s_j < s_k\}$, $B' = B$
     Separation : $O' = O \cup \{s_i < s_k\} \cup \{s_k < s_j\}$, $B' = B \cup \delta$, where $\delta \in \{\alpha \mid \alpha =$ noncodesinate(s) $\cup$ codesinate(P-s), where $s \subseteq P \wedge s \neq \{\ \}\}$.
   - ° Recursive invocation : **POP**(S',O',B',G,L,T-{t})
3. **Goal selection :** Let p be a proposition in G, and let s_{need} be the step for which p is a precondition.
4. **Operator selection :** Let s_{add} be an existing step, or some new step, that adds p before s_{need}. If no such step exists or can be added then backtrack. Let $L' = L \cup \{s_{add} \xrightarrow{p} s_{need}\}$, $S' = S \cup \{s_{add}\}$, $O' = O \cup \{s_{add} < s_{need}\}$, and $B' = B \cup$ the set of variable bindings to make s_{add} add p. $G' = (G - \{p\}) \cup$ preconditions of s_{add}, if new.
5. **Threat identification :** Let $T' = \{t \mid$ for every step s_k that is a positive or negative threat to a causal link $s_i \xrightarrow{p} s_j \in L'$, $t = (s_k, s_i \xrightarrow{p} s_j)\}$.
6. **Recursive invocation :** POP(S',O',B',G',L',T').

이들을 단계별로 펼쳐 놓는다. 짝수 번호의 레벨들은 명제 노드들을 포함하며, 특히 제로 번째 레벨은 계획 문제의 초기상태에서 참인 명제들로만 구성된다. 홀수 번호 레벨들은 동작 노드들을 포함하며, 각 노드는 이전 레벨에 자신의 전제조건을 나타내는 명제들이 모두 존재하는 하나의 동작을 나타낸다. 한 명제 노드 레벨에서는 다음 레벨에 존재하는 각 동작들의 전제조건을 나타내는 링크(link)가 연결되고, 다시 이 동작 노드 레벨에서는 각 동작들의 효과를 나타내도록 다음 명제 노드 레벨로의 링크들이 연결된다. 그래프 계획법은 각 동작들간의 혹은 각 명제들간의 상호 배타적 관계(mutual exclusion relation, mutex)를 판단하고 이것을 이용한다. 상호 배타적 관계(mutex)는 다음과 같이 정의할 수 있다.



[그림 1] 계획 그래프의 한 예

<표 2> 저녁식사데이트 문제의 정의

Initial Condition:(and (garbage)(cleanHands)(quiet))
Goal: (and (dinner) (present) (not (garbage)))
Action:
cook: precondition (cleanHands)
: effect (dinner)
wrap: precondition (quiet)
: effect (present))
carry: precondition
: effect(and(not(garbage)(not (cleanHands)
dolly: precondition
: effect (and (not (garbage)) (not (quiet)))

- 다음의 조건들 중의 하나를 만족하면 i번째 레벨의 두 동작은 상호 배타적 관계에 있다
  - 불일치 효과(inconsistent effects): 한 동작의 효과가 다른 동작의 효과의 부정(negation)일 경우
  - 간섭(interference): 한 동작이 다른 동작의 전제조건을 삭제할 때
  - 경쟁적 요구(competing needs): 동작들이 i-1번째 레벨에서 상호 배타적 관계에 있는 전제조건들을 가질 때
- i번째 레벨의 두 명제들이 서로 부정관계(negation)에 있거나 그 명제들을 달성해주는 i-1번째 동작들이 모두 둘씩 상호 배타적 관계에 있을 경우 (inconsistent support)

[그림 1]는 [표 2]에 정의된 저녁식사 데이트 문제(dinner-date problem)에 대한 계획 그래프를 보여준다. 계획 그래프의 첫 번째 레벨에서 carry동작과 garbage 유지동작이 불일치 효과로 인해 서로 상호 배타적 관계에 있고, 또 dolly와 wrap이 간섭에 의해 역시 상호 배타적 관계에 있다는 것을 알 수 있다. 명제 노드 레벨인 두 번째 레벨에서는 ¬quiet와 present가 상호 배타적 관계에 있다.

해 도출 단계에서는 n 개의 목표 각각을 차례로 고려함으로써 하나의 계획을 찾아간다. i번째 레벨의 그러한 각 리터럴(literal)에 대해 그래프 계획법은 그 부속목표를 달성해주는 i-1번째 레벨의 한 동작 a를 선택한다. 그리고 이 선택을 하나의 후진점(backtrack point)으로 삼는다. 만약 주어진 한 부속목표를 달성해줄 수 있는 동작이 하나 이상 존재하면 완전성(completeness)을 확보하기 위해 그들 모두를 고려해 보아야 한다. 동작 a가 이 레벨에서 현재까지 선택된 다른 모든 동작들과 상호 배타적 관계가 아니면 다음 새로운 부속목표를 고려하고, 만약 그렇지 않으면 다른 동작을 선택해보되 이미 선택된 다른 동작들과 상호 배타적 관계에 있지 않은 동작이 더 이상 존재하지 않을 경우는 이전의 선택으로 후진(backtrack)한다. i-1 레벨에서 일치성이 있는 동작들을 모두 찾아내고 나면 i-2 레벨에서 이 동작들의 전제조건들을 위한 한 계획을 찾으려고 탐색을 계속한다. 이러한 재귀호출(recursion)의 끝은 제로 레벨이며, 요구되는 명제들이 이 레벨에 모두 존재하면 그래프 계획법은 하나의 해를 찾은 것이다. 만약 그렇지 못하고 가능한 모든 선택들에 대한 후진이 실패로 끝날 경우는 다시 그래프 확장 단계로 들어가 동작 레벨과 명제 레벨을 한 레벨씩 더 확장한 다음에 해 도출을 시도해본다. [그림 1]은 저녁식사 데이트 문제에 대해 4번째 레벨까지 그래프가 확장된 뒤 해 도출 과정을 통해 가능한 4가지 계획중의 하나를 찾는 것을 보여주고 있다.

3. 실험계획

3.1 실험목표

본 논문에서는 응용범위가 넓고 비교적 복잡도가 높은 화물운송 영역에서 대표적인 인공지능 계획방법인 부분순서 계획법과 그래프 계획법간의 성능을 서로 비교 분석하기 위해 몇 가지 실험을 하고자 한다. 또한 이러한 실험을 통해 그래프 계획법의 성능에 큰 영향을 줄 수 있는 제어전략들의 효과에 대해서도 서로 비교를 하고자 한다. 먼저 첫 번째 목표인 부분순서 계획법과 그래프 계획법간의 성능 비교를 위해서는 임의로 생성한 복잡도가 다른 몇 개의 화물운송 계획 문제들에 대해 부분순서 계획기와 그래프 계획기를 이용하여 풀어보고, 그 중에서 성공적으로 해를 구한 문제의 수, 해 계획(solution plan)의 길이, 소요된 CPU 시간 등을 측정하여 서로 비교해본다. 또 두 번째 목표인 그래프 계획법에서 부속목표 선택 순서와 동작 선택 순서에 관한 제어전략들의 효과를 비교 분석하기 위해서 가장 보편적으로 사용되는 부속목표 선택 전략의 하나인 DVO와 역시 가장 널리 사용되는 동작 선택 전략의 하나인 LPVO를

대상으로 실험한다. DVO 전략은 현재 미달성 부속목표들 중에서 그것을 달성해줄 수 있는 동작들이 가장 적은 부속목표를 가장 우선적으로 선택해 성취해 가는 전략이다. 반면에 LPVO 전략은 하나의 부속목표를 달성해줄 수 있는 동작들이 여러 개 존재 할 때 이들 중에서 가장 적은 수의 전제조건을 가진 동작을 우선적으로 선택해 주어진 부속목표를 성취해 보려는 전략이다. 두 전략 모두 각각 부속목표 선택 순서와 동작 선택 순서에 관한 최소결정 전략(least-committment strategy)의 하나이다. 부분순서 계획법에서 두 전략에 관한 효율성 연구는 이미 Joslin과 Pollack의 선행 연구[7]를 통해 그 효과가 입증된 바 있어 본 연구에서는 관련 실험을 제외한다. 본 연구에서는 (1)두 전략 모두 사용하지 않은 경우, (2) DVO 전략만 사용한 경우, (3) LPVO 전략만 사용한 경우, (4) 두 전략 모두 사용한 경우 각각에 대해 그래프 계획기를 이용해 문제들을 풀어보고, 역시 성공적으로 해를 구한 문제의 수, 해 계획의 길이, 소요된 CPU 시간 등을 측정하여 서로 비교해본다.

송화물과 이들이 운송되어야 할 몇 개의 도시와 지역, 그리고 운송수단으로 몇 대의 트럭과 비행기가 이용되며 이들과 관련한 기본 동작들로는 특정 운송화물을 싣거나 내리거나 특정지역으로 이동하는 등 총 8 개의 기본 동작들만을 포함한다. 각 개별 계획문제는 그 문제에 등장하는 사물들과 더불어 문제의 초기상태(initial state)와 목표상태(goal state)의 쌍으로 주어진다.

본 연구의 실험에는 [표 5]에 나타나 있는 것과 같이 총 7 개의 서로 다른 화물운송 계획 문제들을 사용하였다. 실험에 사용된 문제는 [표 5]에서 보듯 운송화물의 수, 운송도시 및 지역의 수, 비행기의 수, 트럭의 수, 최종목표의 리터럴 개수 각각에서 복잡도를 점차 증가시켜 만든 임의의 문제들이다. 따라서 비록 실험에 사용된 문제의 개수는 그다지 많지 않으나 각 문제들이 복잡도 면에서 차별화된 문제집단들을 각각 대표한다는 점에서 비교적 쉬운 유사한 문제들만을 대상으로 한 실험보다 훨씬 의미가 크다고 판단한다.

<표 3>PDDL 형태의 한 화물운송 영역에 대한 정의

```
(define (problem prob-2) (:domain transportation)
  (:objects package1 seoul-truck pusan-truck airplane1
      pusan-po seoul-po pusan-airport seoul-airport seoul pusan)
  (:init (OBJ package1)(TRUCK seoul-truck)(TRUCK pusan-truck)
      (AIRPLANE airplane1)(LOCATION pusan-po)
      (LOCATION seoul-po)(LOCATION pusan-airport)
      (LOCATION seoul-airport)(AIRPORT pusan-airport)
      (AIRPORT seoul-airport)(CITY seoul)(CITY pusan)
      (IN-CITY seoul-po seoul)(IN-CITY seoul-airport seoul)
      (IN-CITY pusan-po pusan)(IN-CITY pusan-airport pusan)
      (at package1 seoul-po)(at airplane1 seoul-airport)
      (at pusan-truck pusan-po)(at seoul-truck seoul-po))
  (:goal (and (at package1 pusan-airport))))
```

3.2 실험방법

본 연구를 위한 실험은 2개의 233MHz 펜티엄II 프로세서와 256MB의 주기억장치를 사용하는 Windows NT Server 4.0 워크스테이션에서 Allegro Common LISP 3.0을 이용하여 이루어졌다. 실험을 위한 화물운송 영역지식과 각 문제의 정의는 각각 [표 3]와 [표 4]에 나타나 있는 것처럼 PDDL 형태로 표현하여 입력하였다. PDDL은 인공지능계획시스템 국제학술회의인 AIPS에서 연례적으로 개최하는 계획시스템 경진대회를 위해 제정한 표준 계획문제 기술 언어로서 현재 교육 및 연구를 위한 표준으로 이용되고 있다. 본 연구의 실험을 위한 화물운송 영역은 몇 개의 운

부분순서 계획법을 위한 실험에는 대표적인 부분순서 계획기로 널리 보급되어 있는 UCPOP[12]을 이용한다. UCPOP은 미국 워싱턴 주립대학에서 Common LISP으로 개발된 것으로 전체한정사(universal quantifier), 조건부효과(conditional effect), 부정(negation)등 다양한 동작표현 양식을 지원하고 있으며 공개된 부분순서 계획기 중 비교적 수행효율이 높다. 부분순서 계획기를 이용하여 화물운송 계획 문제를 풀 경우에는, 성공적으로 해를 구한 문제의 수, 계획수립과정 동안 생성된 부분계획의 수, 실제로 검사한 부분계획의 수, 결과적으로 얻어진 해 계획의 길이, 총 소요된 CPU 시간 등을 측정하여 비교한다.

한편, 그래프 계획법을 위한 실험에는 역시 워싱턴 대학에서 개발되어 널리 보급되어진 대표적인 그래프 계획기인 SGP를 이용한다. 공개된 그래프 계획기는 SGP외에도 Graphplan, IPP, STAN 등이 있지만 이

[표 4] PDDL형태의 한 화물운송 계획문제

```
(define (problem prob-2) (:domain transportation)
  (:objects package1 seoul-truck pusan-truck airplane1
      pusan-po seoul-po pusan-airport seoul-airport seoul pusan)
  (:init (OBJ package1)(TRUCK seoul-truck)(TRUCK pusan-truck)
      (AIRPLANE airplane1)(LOCATION pusan-po)
      (LOCATION seoul-po)(LOCATION pusan-airport)
      (LOCATION seoul-airport)(AIRPORT pusan-airport)
      (AIRPORT seoul-airport)(CITY seoul)(CITY pusan)
      (IN-CITY seoul-po seoul)(IN-CITY seoul-airport seoul)
      (IN-CITY pusan-po pusan)(IN-CITY pusan-airport pusan)
      (at package1 seoul-po)(at airplane1 seoul-airport)
      (at pusan-truck pusan-po)(at seoul-truck seoul-po))
  (:goal (and (at package1 pusan-airport))))
```

[표 5] 실험문제들의 구성

problem	packages	cities	locations	airplanes	trucks	goals
prob-1	1	2	4 (2)	1	2 (1)	1
prob-2	1	2	4 (2)	1	2 (1)	1
prob-3	1	2	4 (2)	1	2 (1)	1
prob-4	8	3	6 (2)	2	3 (1)	1
prob-5	8	3	6 (2)	2	3 (1)	2
prob-6	3	3	9 (3)	2	3 (1)	3
prob-7	4	3	9 (3)	2	3 (1)	4

들은 모두 C 언어로 구현되었을 뿐 아니라 나름의 독특한 최적화기법들을 내포하고 있다. 따라서 UCPOP과의 합리적인 비교를 위해서는 역시 Common LISP으로 구현되어 있으며 비교적 손쉽게 부속목표 선택 전략인 DVO와 동작 선택 전략인 LPVO를 구현하기 용이한 SGP를 선택하였다. 그래프 계획법에서 DVO 전략과 LPVO 전략의 효과를 비교하기 위해서 (1)두 전략 모두 사용하지 않은 경우(NONE), (2) DVO 전략만 사용한 경우(DVO), (3) LPVO 전략만 사용한 경우(LPVO), (4) 두 전략 모두 사용한 경우(DVO+LPVO) 각각에 대해 그래프 계획기 SGP를 이용해 문제들을 풀어보고, 성공적으로 해를 구한 문제의 수, 계획그래프의 최종 레벨과 해 계획내의 동작수, 소요된 CPU 시간 등을 측정하여 서로 비교해본다.

4. 실험결과 및 분석

4.1 실험결과

[표 6]은 7 개의 화물운송 계획문제를 부분순서 계획기인 UCPOP으로 푼 결과를 보여준다. 각 문제를 풀 때 총 100만개의 부분계획 노드 생성을 탐색한계(search limit)로 삼아 실험한 결과 이들 중 단지 2 개의 문제만을 성공적으로 풀어내고 나머지 5 개의 문제는 모두 실패로 끝났다. 표에서는 성공적으로 문제를 푼 경우에 대해 계획수립을 위한 탐색과정 중에 생성된 총 부분계획 노드의 수와 이들 중 실제 검사된 부분계획 노드의 수, 해 계획의 길이 즉 해 계획에 포함된 동작들의 수, 총 소요된 CPU 계산시간 등을 보여준다. 한편 [표 7]과 [표 8]은 동일한 화물운송계획문제들을 그래프 계획기인 SGP로 푼 결과를 보여준다. [표 7]은 서로 다른 4 가지 제어전략의 조

<표 6>부분순서계획기의 실험결과

problem	created partial plans	explored partial plans	actions in the solution plan	CPU Time(sec)
prob-1	206	89	3	0.156
prob-2	438870	176926	6	488.251
prob-3	1067184(fail)	449100(fail)	fail	1178.021(fail)
prob-4	fail	fail	fail	fail
prob-5	fail	fail	fail	fail
prob-6	fail	fail	fail	fail
prob-7	fail	fail	fail	fail

[표 7] 그래프계획기의 실험결과: Solution Plan

problem	none		dvo		lpvo		dvo+lpvo	
	levels	actions	levels	actions	levels	actions	levels	actions
prob-1	3	3	3	3	3	3	3	3
prob-2	6	9	6	9	6	6	6	6
prob-3	9	18	9	18	9	9	9	9
prob-4	9	18	9	18	9	10	9	10
prob-5	9	29	9	31	9	14	9	14
prob-6	9	41	9	41	9	25	9	25
prob-7	9	48	9	48	9	25	9	25

합인 NONE, DVO, LPVO, DVO+LPVO를 각각 적용한결과 구해진 최종 계획그래프의 레벨과 해 계획에 포함된 동작들의 수를 보여준다. 또 [표 8]은 4 가지 제어전략의 조합을 적용한 결과. 해 계획을 구할 때까지 소요된 총 CPU 계산시간을 보여준다.

4.2 비교분석

화물운송 계획문제들에 대한 실험결과를 토대로 부분순서 계획기와 그래프 계획기간의 성능을 서로 비교하면 다음과 같다. 먼저 전체 문제의 수와 실제로 성공적으로 풀어낸 문제의 수를 비교해보면 부분순서 계획기의 경우는 주어진 탐색한계 내에서 총 7개의 문제 중 비교적 쉬운 2개의 문제밖에 풀지 못하였고 반면에 그래프 계획기는 비교적 복잡도가 높은 문제들을 포함해 주어진 7 개의 문제를 모두 풀어냈다. 또 [표 6]에서 부분순서 계획기가 성공적으로 문제를 푼 경우를 살펴보면 문제의 복잡도가 조금 증가하면 생성된 부분계획의 수와 검사된 부분계획의 수가 보여주듯 탐색공간이 급격히 증가하게 되어 메모리 요구량이 폭증하고 계획수립에 필요한 CPU 계산시간도 따라 증가한다는 것을 알 수 있다. 하지만 [표 7]과 [표 8]에서 보듯이 그래프 계획기는 복잡도가 매우 높은 문제들의 경우에도 메모리 요구량이 그다지 많지 않고 4-8초 이내의 매우 짧은 CPU 계산시간 내에 풀어냈다. 따라서 그래프 계획기의 경우는 문제의 복잡도에 따라 쉽게 메모리 요구량이나 계산시간이 증가하지 않는 매우 안정적인 성능을 보여준다는 것을 알 수 있다. 한편 계획수립 결과 얻어진 해 계획(solution plan)의 질(quality)면에서 두 계획기를 비교해보면 이 경우 비교할 대상문제가 적기는 하지만 부분순서 계획기는 풀어낸 두 문제 모두 해 계획의 길이가 가장 짧은 최적의 해(optimal solution)를 구한 반면 그래프 계획기는 적용하는 제어전략에 따라 해 계획의 길이가 서로 다르다는 것을 알 수 있다.

[표 8] 그래프계획기의 실험결과: CPU Time(sec)

problem	none	dvo	lpvo	dvo+lpvo
prob-1	0.125	0.11	0.125	0.109
prob-2	0.281	0.297	0.297	0.296
prob-3	0.515	0.516	0.516	0.515
prob-4	8.531	8.484	8.485	8.391
prob-5	8.641	8.641	8.438	8.375
prob-6	4.391	4.453	4.156	4.234
prob-7	9.547	5.984	5.344	5.422

다음은 [표 7]과 [표 8]의 실험결과를 토대로 그래프 계획기에 적용된 제어전략들간의 성능을 서로 비교해본다. 먼저 최종 계획그래프의 레벨 수와 해 계획에 포함된 동작의 수는 해 계획의 병렬성(parallelism) 및 해 계획의 질(quality)과 밀접한 관련이 있다. 일반적으로 계획그래프의 레벨 수가 같은 경우 계획에 포함된 동작의 수가 많을수록 병렬성이 높아진다고 볼 수 있으나 반면에 계획에 불필요한 동작들을 포함할 가능성이 상대적으로 높아진다. 본 연구의 실험결과에서는 4 가지 제어전략의 조합 모두 각 문제에 대해 동일한 레벨 수를 보여주고 있다. 하지만 계획에 포함된 동작의 수에서는 큰 차이를 보이고 있다. LPVO와 DVO+LPVO의 제어전략을 사용한 경우가 NONE과 DVO의 경우보다 모든 문제에 대해 더 적거나 같은 동작수를 보였으며 본 실험의 경우 이들이 곧 각 문제의 최적 해였다. NONE이나 DVO의 제어전략을 사용하여 얻어진 해 계획들을 면밀히 검사해본 결과 대부분의 해 계획들이 모두 불필요한 동작들을 일부 포함하고 있다는 사실을 확인할 수 있었다. 또한 [표 8]에서 계획수립에 소요된 CPU 계산시간을 비교해보아도 역시 LPVO와 DVO+LPVO의 제어전략을 사용한 경우가 거의 모든 문제에서 NONE과 DVO의 경우보다 더 빠른 결과를 보이고 있다. 하지만 NONE과 DVO 혹은 LPVO와 DVO+LPVO 간에는 CPU 계산시간이나 해 계획의 질 면에서 서로 뚜렷한 차이를 보이고 있지 않다. 따라서 본 실험에서는 그래프 계획법의 부속목표 선택전략인 DVO는 예상과는 달리 별다른 효과를 나타내지 못한 반면 동작 선택 전략인 LPVO는 뚜렷한 성능 향상을 가져온다는 사실을 알 수 있다. 이러한 결과는 본 실험의 대상영역인 화물운송 계획문제의 특성에도 그 원인이 있으나 본래 그래프 계획법에서는 부속목표들을 어떤 순서로 달성해 가느냐하는 제어문제가 선형 계획법이나 부분순서 계획법에 비해 성능에 영향을 덜 미친다는데 주된 원인이 있다.

5. 결 론

본 논문에서는 화물운송 계획문제들을 대상으로 몇 가지 실험을 통해 대표적인 인공지능 계획방식인 부분순서 계획법과 그래프 계획법의 성능을 비교 분석하였다. 또 동시에 이러한 실험을 통해 DVO 및 LPVO와 같은 대표적인 제어전략들을 중심으로 이들이 그래프 계획법의 성능에 미치는 효과를 비교 분석하여 보았다. 본 연구의 실험을 통해서는 부분순서 계획법에 비해 그래프 계획법이 메모리 사용량이나 CPU 계산시간 면에서 월등히 우수한 성능을 보여주었으며 비교적 복잡도가 큰 계획문제에서도 좋은 결과를 보여주었다. 하지만 도출된 해 계획의 질적인 면에서는 부분순서 계획법이 대부분 최적의 해를 찾아낸 것에 반해 그래프 계획법은 사용된 제어전략과 최적화 방법에 따라 해 계획의 질이 크게 달라질 수 있음을 보였다. 한편 그래프 계획법에서는 부속목표 선택 전략인 DVO는 그 효과를 뚜렷이 보이지 못한 반면 동작 선택 전략인 LPVO는 도출된 해 계획의 질적인 면이나 계산속도 면에서 모두 뛰어난 효과를 보여주었다. 한편 부분순서 계획법이 비록 성능 면에서는 그래프 계획법에 비해 뒤떨어지지만 전체 한정사나 조건부 효과 등 동작 표현에 다양성을 지원하기 쉽고 조건부 계획생성 기능과 하나의 틀 안에서 계획생성과 계획실행을 번갈아 수행할 수 있는 기능 등을 제공하기 용이한 구조를 가지고 있다. 반면에 명제논리와 제약만족기법에 기초한 그래프 계획법은 이러한 기능을 제공하기엔 구조적인 어려움이 있을 뿐 아니라 비교적 최근에 제안된 계획방법인 이유로 관련연구가 아직 부족한 실정이다. 향후 진행되어야 할 연구로는 보다 충분한 실험적 데이터의 보완과 실험적 결과를 뒷받침할만한 분석적 연구가 뒤따라야 할 것으로 판단한다.

참 고 문 헌

[1] F. Bacchus and P. van Run, "Dynamic variable ordering in CSPs", Proceedings of the 1995 conference on Principles and Practice of Constraint Programming, pp.258-275, 1995.

[2] A. Barrett and D. Weld, "Partial order planning: Evaluating possible efficiency gains", J.Artificial Intelligence, vol.67, no.1, pp.71-112, 1994.

[3] A. Blum and M. Furst, "Fast planning through planning graph analysis", J. Artificial Intelligence, vol.90, no.1-2, pp.281-300, 1997.

[4] D. Chapman, "Planning for conjunctive goals", J. Artificial Intelligence, vol.32, no.3, pp.333-377, 1987.

[5] J. Cheng and K. B. Irani, "Ordering problem subgoals", Proc. 11th Int. Joint Conf. AI, pp.931-936, August 1989.

[6] M. Ernst, T. Millstein, and D. Weld, "Automatic sat-compilation of planning problems", Proc. 15th Int. Joint Conf. AI, 1997.

[7] D. Joslin and M. Pollack, "Least-cost flaw repair: A plan refinement strategy for partial-order planning", Proc. 12th nat. Conf. AI, July 1994.

[8] R. Kambhampati, E. Lambrecht, and E. Parker, "Understanding and extending graphplan", Proc. 4th European Conf. on Planning, Sept 1997.

[9] S. Kambhampati, "Ebl and ddb for graphplan", Department of Computer Science and Engineering TR-99-008, Arizona State University, August 1998.

[10] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos, "Extending planning graphs to an ADL subset", Proc. 4th European Conf. on Planning, pp.273-285, Sept 1997.

[11] D. McAllester and D. Rosenblitt, "Systematic nonlinear planning", Proc. 9th Nat. Conf. AI, pp.634-639, July 1991.

[12] J.S. Penberthy and D. Weld, "UCPOP: A sound, complete, partial order planner for ADL", Proc. of KRR-92, Oct., 1992.