의존성 구조 학습을 통한 masking 효과 축소

한 경 식, 이 수 원
숭실대학교 컴퓨터학부

Decreasing the Masking Effect by Learning Dependence Structures

Kyoungsik Han, Soowon Lee
School of Computing, Soong Sil University

요 약

설명 기반 학습은 시스템 성능향상에 필요한 탐색 제어 지식을 학습하는 방법으로 많이 이용되고 있다 EBL은 과거의 문제풀이 과정을 일반화하여 학습한 다음 이와 유사한 상황이 발생한 경우, 문제풀이를 거치지 않고 학습된 해답을 신속하게 제시하여 성능을 향상시킨다. 그러나 새로운 문제 해결이 과거 문제 풀이 해답에 의존할 경우, 그에 대한 해답을 신속히 구할 수는 있지만 해답의 질은 학습 결과에 의존하지 않을 때보다 오히려 못할 수 있다 이러한 현상을 masking 효과라고 한다. 본 논문에서는 의존성 구조를 학습, 이용하여 이러한 masking 효과를 축소하고자 한다. 의존성 구조는 현 상태에서 선택된 연산자가 이후의 문제 풀이에 끼치는 영향을 포함하는 구조로서, 이후 유사한 상황에 대해 선택될 연산자의 적합성 및 효율성을 평가하는 기준으로 사용될 수 있다는 점에서 masking 효과를 축소할 수 있다

1. 서 론

설명 기반 학습(Explanation-Based Learning)[1, 2]은 시스템 성능향상에 필요한 탐색 제어 지식(search control knowledge)을 학습하는 표준적인 방법으로 Soar[3], Prodigy[4]등의 여러 시스템에서 사용되고 있다 EBL은 과거의 문제풀이 결과를 일반화하여 학습한 다음 이와 유사한 상황이 발생할 경우, 문제 풀이를 거치지 않고 학습된 해답을 신속하게 제시하여 성능을 향상시킨다 EBL을 사용하는 시스템은 주어진 문제에 대하여 어떻게 탐색을 수행하는가에 대한 지식을 습득하게 되고, 탐색시 제어에 따른 결과를 분석할 수 있게 되어 그로부터 얻은 결론은 다음의 유사한 상황에 유용하게 이용된다 특히, EBL을 통해 생성된 제어 지식은 연역적 추론 과정을 통한 것이므로 항상 정확성(correctness)이 보장되고 학습을 위해 오직 하나의 시례만이 필요하기 때문에 문제 해결기(problem solver) 또는 계획기(planner)등과 같은 성능 시스템(performance system)에서 많이 선호되는 방법이다.

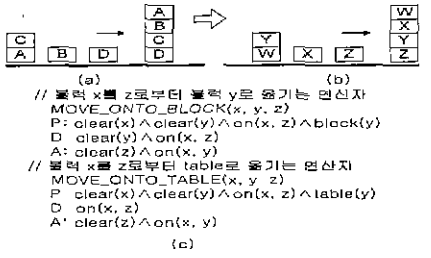
그러나 EBL에서 새로운 문제의 해결과정이 과거 문제 풀이 해답에 의존할 경우, 그에 대한 해답을 신속히 구할 수는 있지만 새로 구한 해답의 질(solution quality)은 학습 결과에 의존하지 않을 때보다 오히려 못할 수 있다 그 이유는 학습된 지식을 사용함으로써, 그렇지 않을 경우 찾아 낼 수 있을 지 모르는 보다 나은 대안을 탐색할 기회를 시스템 스스로 막아 버리기 때문이다. 이러한 현상을 masking 효과라고 한다[5, 6] masking 효과는 일반적으로 유용성 문제(utility problem)[7]의 구별된다. 유용성 문제는 학습 이후 시스템의 효율성의 저하(degradation of efficiency)를 뜻하고 masking 효과는 학습 이후 해답의 질적 저하(degradation of quality)를 뜻한다

본 논문에서는 이러한 masking 효과를 축소하기 위해 의존성 구조(dependence structure)를 학습하는 방법을 제시한다. 의존성 구조는 현 상태에서 선택된 연산자가 이후의 문제 풀이에 끼치는 영향을 포함하는 구조로서 이후 유사상황에 대해서는 선택될 연산자의 적합성 및 효율성을 평가하는 기준으로 사용된다 본 논문의 2절은 masking 효과의 사례를 살펴보고 3절에서는 이러한 masking 효과 축소 방안인 의존성 구조를 정의하고 이를 학습하는 방법을 설명한다 4절은 구현 및 실험에 대해 설명하고 마지막 5절에서는 본 논문의 결론 및 향후 연구에 대해 논한다.

2. masking 효과 사례

masking 효과가 발생하는 이유는 다음과 같다. 첫째, 저품질의 해를 유도할 수 있는 상황에도 적용될 정도로 학습된 제어지식의 선행조건이 과잉일반화(overgeneral)[8]될 수 있기 때문이다. 이에 대해 Tambe는 approximation과 filter의 개념을 도입하여 학습된 계획이 사용되는 순간 그 계획이 부적절한 경우에 대한 경고를 함으로써 시스템이 보다 나은 해를 구하도록 했다[6].

둘째, EBL을 사용하는 시스템은 다른 연산자간의 상호관계(의존성, 효율성)에 대한 고려 없이 과잉구체적인(overspecific) 행위를 학습하고 이를 무조건으로 선호하기 때문에 masking 효과가 발생할 수 있다 본 논문에서 제시하는 의존성 구조 학습 방법은 이 두가지 경우 모두에 대하여 masking 효과를 축소할 수 있으나, 본 논문에서는 과잉구체인 행위의 학습에 의해 발생하는 masking 효과 축소에 대하여 설명한다. 블록의 세계에서 학습된 계획의 과잉구체화로 인해 발생하는 masking 효과의 예는 다음과 같다. [그림 1](a)는 학습이 일어나는 상황의 초기 상태와 목표 상태를 나타내고 그림 [그림 1](b)는 [그림 1](a)에서 학습된 계획이 적용될 수 있는 초기 상태와 목표 상태를 나타낸다. [그림 1](c)는 연산자의 정의를 나타낸다.



[그림 1] : 블록의 세계

(a) : 과잉구체적으로 학습이 발생하는 상황

(b) : (a)의 과잉구체화로 masking 효과가 발생하는 상황

(c) : 연산자 정의

계획과 학습을 수행하는 성능 시스템이 [그림 1](a)에 대하여 다음과 같은 계획을 생성했다고 가정하자. 아래의 계획에서 보통체는 계획 생성 과정에서 선택은 되었으나 전조건의 미성취로 종속목표를 생성하는 연산자를 의미하고, 볼드체는 선택되어 현 상태에 적용되는 연산자를 뜻한다

MOVE_ONTO_BLOCK(A, B, TABLE),
MOVE_ONTO_BLOCK(C, B, A),
MOVE_ONTO_BLOCK(B, C, TABLE),
MOVE_ONTO_TABLE(C, TABLE, B),
MOVE_ONTO_BLOCK(C, D, TABLE),
MOVE_ONTO_BLOCK(B, C, TABLE),
MOVE_ONTO_BLOCK(A, B, TABLE)

위 계획에서 MOVE_ONTO_BLOCK(C, B, A)는 MOVE_ONTO_BLOCK(A, B, TABLE)가 생성한 종속 목표인 clear(A)를 성취한다. 그러나 MOVE_ONTO_BLOCK(C, B, A)는 이후에 선택될 연산자의 선행조건 clear(B)를 취소하였으며, 성능 시스템은 이를 다시 성취하기 위해 MOVE_ONTO_TABLE(C, TABLE, B)를 실행하고 있다. EBL은 위의 과정을 학습하여 clear(A)를 성취하기 위해 블록-C를 블록-B에 옮기는 연산자를 선호하도록 한다. [그림 1](b)는 [그림 1](a)의 유사한 상황으로 학습된 계획이 직접적으로 적용될 수 있다. [그림 1](b)에서는 [그림 1](a)에서 학습된 계획에 의해 clear(W)를 성취하기 위한 연산자 MOVE_ONTO_BLOCK(Y, X, W)를 선택하게 된다. 이 과정에서 학습된 계획의 과잉구체적인 특성에 의해 발생하는 masking 효과를 관찰할 수 있다. 시스템이 [그림 1](a)에 대한 학습된 계획을 갖지 않았다면 clear(W)를 위해 MOVE_ONTO_BLOCK(Y, Z, W)와 같은 보다 좋은 대안을 선택할 수 있어 보다 고품질의 계획을 생성 할 수 있었을 것이다. 그러나 학습된 계획은 clear(W)를 성취하기 위해 블록-Y를 블록-X로 옮기도록 과잉구체적으로 지정하여 보다 더 좋은 대안을 탐색할 수 있는 기회를 막아버려 저품질의 계획을 생성하게 한다.

이러한 masking 효과는 계획 생성 과정을 모니터하여 생성되는 의존성 구조를 학습하여 축소할 수 있다.

3. 의존성 구조 학습을 통한 making 효과 축소

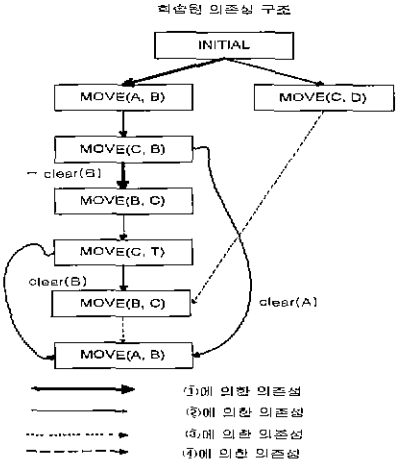
의존성 구조는 선택된 연산자의 이후의 계획 생성에 미치는 영향을 포함하는 구조이기 때문에, 이후 유사한 문제를 만날 경우 의존성 구조는 선택될 연산자의 적합성 및 효율성을 평가하는 도구로 사용되어 masking 효과의 축소를 가져올 수 있다. [그림 2]는 [그림 1](a)의 같은 계획을 수행한 경우의 의존성 구조를 나타내며 이와 같은 의존성 구조는 다음의 4가지 요소에 의해 구성된다. ([그림 2]에서 표기의 간결화를 위해 MOVE_ONTO_TABLE과 MOVE_ONTO_BLOCK을 MOVE로 표현하였다)

① 선택된 연산자가 이후에 선택되는 연산자의 전조건을 방해하는 경우의 의존성이다. 이 경우 현 계획 생성 과정은 현재 선택된 연산자가 다른 연산자의 실행의 방해하는 등 악영향을 끼치고 있음을 의미한다. 이러한 의존성은 이후의 새로운 문제를 풀 경우 제거되어야 한다. [그림 2]에서 MOVE(C, B)가 MOVE(B, C)의 전조건을 취소하므로써 생성되는 의존성이 이에 해당된다.

② 현재 선택된 연산자가 나중에 선택된 연산자의 전조건을 성취시키는 경우의 의존성이다. 이 경우 현재 선택된 연산자는 나중에 선택된 연산자의 전조건을 성취시키기 위해 먼저 선택 실행되었음을 의미한다. [그림 2]에서 MOVE(C, T)가 MOVE(B, C)의 전조건 clear(B)를 성취하므로써 발생하는 의존성이 이에 해당된다.

③ 새로운 종속목표를 생성하여 종속 연산자를 생성 하므로써 발생하는 의존성이다. 이 경우 종속 연산자가 생성, 선택되기 위해 상위 연산자가 먼저 선택되어 목표를 생성하였음을 의미한다. [그림 2]에서 MOVE(A, B)가 종속목표 clear(A)를 생성하여 MOVE(C, B)가 생성, 선택하게 하므로써 생성되는 의존성이 이에 해당된다.

④ 나중에 선택된 연산자가 현재 선택된 연산자의 전조건을 제거하는 경우 나중에 선택된 연산자는 현재 연산자를 보호하기 위해 현 연산자보다 이후에 선택되어야함에 따른 의존성이다. 이는 현재 계획 생성과정은 현 연산자를 먼저 선택함으로서 나중에 선택된 연산자의 방해를 피해 효율적으로 계획을 생성하였음을 의미하며 이 요소는 보호성에 의해 의존성 구조를 형성함을 뜻한다. [그림 2]에서 MOVE(B, C)의 전조건 clear(B)를 보호하기 위해 MOVE(A, B)보다 먼저 선택, 적용되게 하므로써 생성되는 의존성이 이에 해당된다.



[그림 2] : [그림 1]에 대한 의존성 구조

이렇게 생성된 의존성 구조는 이후 유사한 상황 대해 다음과 같이 이용될 수 있다. 첫째, 의존성 구조는 넓이 우선 탐색으로 연산자를 선택해 나간다. [그림 1](a)와 같이 학습하여 [그림 1](b)에 적용할 경우, 일반적인 EBL은 MOVE(W, X)를 선택하여 종속목표 clear(W)를 생성한다. 그러나 의존성 구조에서는 MOVE(W, X)와 MOVE(Y, Z)사이에 의존성이 존재하지 않기 때문에 이중 임의로 선택할 수 있다. 만약 MOVE(Y, Z)를 선택할 경우, 보다 목표 상태에 가까운 상태로 진화되어 고품질의 계획을 생성할 수 있다. 둘째, 만약 선택될 연산자가 이후에 선택되는 연산자의 전조건을 방해하는 경우, 이 연산자를 선택에서 제거하고 다른 연산자를 선택한다. 이 과정에서 선택될 수 있는 연산자는 제거된 연산자가 성취하는 목표를 성취하고 의존성 구조에서 지금까지 탐색되지 않은 연산자의 전조건을 방해하지 않아야 한다. 이와 같은 이유는 새로 선택된 연산자로 인해 학습된 의존성 구조가 깨지는 것을 방지하기 위함이다. [그림 1]을 예로 들면, 일반적인 EBL은 2장에서 설명한 것과 같이 clear(W)를 성취하기 위해 MOVE(Y, X)를 선택하여 저품질의 계획을 생성할 것이다. 그러나 의존성 구조는 이후 계획 생성에 미치는 영향을 포함하고 있기 때문에 이를 이용하여 비효율적인 선택을 제거하여 고품

질의 계획을 유도할 수 있는 연산자(MOVE(Y, Z) 혹은 MOVE(Y, T))를 선택하게 한다. 의존성 구조 이용에서의 악영향을 끼치는 연산자 선택의 제거는 학습된 제어지식의 과잉구체적인 행위로 인해 발생하는 masking 효과를 제거할 수 있다 이와 같은 방법으로 의존성 구조를 이용할 경우 [그림 1](b)에 대한 계획으로 다음과 같은 최적의 계획을 생성할 수 있다

MOVE_ONTO_BLOCK(Y, Z, W),
MOVE_ONTO_BLOCK(X, Y, TABLE),
MOVE_ONTO_BLOCK(W, X, TABLE)

4. 구현 및 실험

본 논문의 의존성 구조는 Soar에서 구현되었다. Soar는 계획, 문제해결, 학습, 추론 등의 지능적 행위를 구현할 수 있도록 하는 통합 아키텍쳐이다 Soar는 EBL과 유사한 청킹(chunking)이라는 학습 메카니즘을 사용하며 청킹은 Soar의 문제 풀이 경험을 기록한 조건-결론문 형태의 청크(chunk)를 생성한다[9]

의존성 구조를 학습하여 masking 효과를 축소하기 위해서는 크게 두 가지 규칙 즉, 의존성 구조 생성 규칙, 의존성 구조 넓이 우선 순위 탐색 규칙이 필요하다. 의존성 구조 생성 규칙은 현 연산자가 지금까지 생성된 의존성 구조에 끼치는 4가지 영향에 따라 현 연산자에 대한 의존성 구조를 생성하게 하며 의존성 구조 넓이 우선 순위 탐색 규칙은 의존성 구조를 이용할 때에 의존성 구조를 넓이 우선 순위로 탐색하면서 비효율적인 선택을 제거하여 masking 효과가 축소되도록 한다

본 논문에서 제시하는 접근방법의 실험적 평가는 블록의 세계에서 수행되었다. 블록의 세계에서 3개 블록 및 4개 블록 각각 12개, 18개의 총 30개의 학습 문제를 임의로 추출하여 의존성 구조 및 일반적인 EBL 방법으로 학습시켰으며 3개 블록, 4개 블록, 5개 블록에서 각각 10개의 적용 가능한 테스트 문제를 추출하여 성능을 테스트하였다. [표 1]은 이에 대한 실험 결과를 나타낸다

[표 1]의 첫 번째 라인은 masking 효과의 빈도 수를 나타낸다 masking 효과는 학습 이후에 나타나는 해답의 질적 저하로서 여러 측정 방법이 있으나 본 논문에서는 생성된 계획의 길이로 평가하였다 masking 효과의 유무를 측정하기 위해 테스트 문제 30개의 최적의 계획 길이와 각각의 학습을 이용해 유추된 계획 길이를 비교하였다 일반적인 EBL을 사용할 경우 11개의 테스트 문제에서 masking 효과가 발생했지만 의존성 구조를 이용할 경우 단 1개의 문제에서 masking 효과가 발생하였다 여기서 masking 효과가 발생하는 이유는 두 연산자 사이에 명확한 의존성이 없음에도 불구하고 넓이 우선 순위로 의존성 구조를 탐색하였기 때문이다 두 번째 라인은 학습된 계획을 이용할 경우 각각의 decision cycle을 나타낸다 의존성 구조 학습은 의존성 구조를 이용할 때에 현재 선택의 효율성을 고려하여 사용하기 때문에 decision cycle이 감소하였다 세 번째 라인은 계획 알고리즘이 보호성을 사용할 때, 문제의 해결 유무를 나타낸다 일반적인 EBL 학습의 경우, 테스트 문제 6개에 대해 문제를 풀지 못하는 경우가 발생하였으나 의존성 구조 학습의 경우 모든 문제를 풀 수 있었다 네 번째 라인은 재계획(replanning)의 빈도 수를 나타낸다 의존성 구조의 경우는 새로운 문제를 접할 때 현 의존성 구조의 이용 여부를 검증하게된다 이러한 검증에서 현 의존성 구조가 새로운 문제에 과잉일반화인지를 검증하여 재계획 유무를 결정하게 된다 일반적인 EBL 학습의 경우는 이러한 검증없이 바로 학습된 결과를 이용하지만 거의 재계획을 하는 것과 같은 결과가 나타났음을 의미한다

평가 \ 학습방법	의존성 구조 학습	일반적인 EBL 학습
masking 효과 빈도	1개	11개
decision cycles(평균)	10 cycles	14 8 cycles
fail 빈도	0개	6개
replanning빈도	5개	5개

[표 1] . 임의의 문제 30개에서의 실험 결과

5. 결론 및 향후 연구

본 논문에서는 masking 효과 축소를 위한 의존성 구조의 학습을 제시하였다 의존성 구조는 현 상태에서 선택된 연산자가 이후의 계획 생성에 끼치는 영향을 포함하는 구조로써 이후 유사상황에 대해서는 선택된 연산자의 적합성 및 효율성을 평가하는 기준으로 사용된다. 또한 의존성 구조에 포함되는 연산자와 목표를 이용하여 이 구조가 효과적으로 적용될 수 있는 상황을 판단할 수 있다. 본 논문에서는 과잉구체화로 인해 발생하는 masking 효과를 의존성 구조를 이용하여 축소할 수 있음을 보였으며, 과잉구체화에 의해 발생하는 masking 효과는 의존성 구조 탐색 시 과잉구체적으로 지정된 비효율적인 연산자를 제거함으로써 축소되었다.

향후 연구로는 첫째, 의존성 구조의 탐색에 관한 연구가 필요하다 현재 단계에서는 의존성 구조를 탐색하기 위한 방법으로 넓이 우선 순위 방법을 이용하고 있다. 그러나 masking 효과의 보다 효율적인 축소와 학습된 의존성 구조가 이후의 계획 생성에 보다 효과적으로 기여하기 위해서는 이에 적절한 탐색 방법이 필요하다. 둘째, 의존성 구조 선택에 관한 연구이다 시스템이 현 문제에 대해 이용할 수 있는 의존성 구조가 여러 개 있을 때, 어떤 의존성 구조를 선택, 적용하여 새로운 문제를 풀어나갈 것인가 하는 문제이다 의존성 구조 선택에 관한 연구는 현 문제에 대해 이용될 수 있는 의존성 구조가 얼마나 효율적인가를 측정하는 휴리스틱 함수에 대한 연구이며, 적용할 의존성 구조의 선택에 따라 생성되는 계획의 품질 및 시스템의 성능에 영향을 끼치기 때문에 이는 매우 중요한 문제이다

참고문헌

[1] G. F Dejong and R. J. Mooney Explanation-based learning. An alternative view. *Machine Learning*, 1(1), 1986

[2] T M. Mitchell, R Keller, and S. Kedar-Cabelli Explanation-based generalization A unifying view. *Machine Learning*. 1(1), 1986.

[3] P S. Rosenbloom, J. E Laird, and A. Newell. A preliminary analysis of the Soar architecture as a basis for general intelligence *Artificial Intelligence, 47(1-3)*: 289-325, 1991

[4] S. Minton, C. A Knoblock, D. R Kuokka, Y Gil, R. L. Joseph, , and J G Carbonell. Prodigy 2.0. The manual and tutorial Technical Report CMU-CS-89-146, School of Computer Science, Carnegie Mellon University, 1989

[5] Clark, P., and Holte., R Lazy partial evaluation: an integration of explanation-based generalization and partial evaluation *Proceedings of the International conference on Machine Learning*. 1992, pp 82-91

[6] Tambe, M and Rosenbloom, P., On the Masking Effect, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 526-533, Washington, DC, 1993

[7] S Minton *Learning Effective Search Control Knowledge: An Explanation-Based Approach* PhD thesis, Carnegie Mellon University, 1988. Available as technical report CMU-CS-88-133

[8] Laird, J E, Rosenbloom, P S, and Newell, A. Overgeneralization during knowledge compilation in Soar *Proceedings of the Workshop on Knowledge Compilation*, September, 1986, pp. 46-57

[9] Laird, J E., Rosenbloom, P.S., and Newell, A (1986). Chunking in soar The anatomy of a general learning mechanism *Machine Learning*, 1.11-46