협상전략생성 Q-Table

"테스트 전이기 때문에 알고리즘 수정 가능성 있음"

요약

[테스트 개요]

- **목적**: LLM이 협상 카드 정보만으로 초기 Q-Table을 유의미하게 생성할 수 있는지 평가
- **예상 결과물**: 시나리오별 상태-행동 매핑이 포함된 초기 Q-Table
- **테스트 방식**: 카드 조건을 입력값으로 LLM에게 Q값 생성을 요청하고 직관적 합리성 및 다양성 평가
- **예상 기간**: 약 2~3일 내 1차 Q-Table 생성 및 간단 검토 가능

요구사항

- LLM Agent를 통해 학습 없이 도입 가능한 알고리즘
- 통제 가능한 알고리즘
- 조건에 따라 동적으로 협상 카드를 선택하는 알고리즘
- 결과 데이터를 학습하여 알고리즘 성능이 개선되는 방식

배경

기업 간의 협상 과정은 서로 번갈아가며 조건을 제시하는 일종의 **순차적인 상호작용 게임**으로 볼 수 있다. 협상 참여자들은 각자의 턴마다 특정 **행동**(예: 협상 카드 제시, 가격 입력 등)을 선택하며, 이 과정은 서로의 전략에 따라 복잡한 상호작용 구조를 형성한다.

협상에서 좋은 결과를 얻기 위해서는 각 상태에서 가능한 행동 중 최적의 행동을 전략적으로 결정할 필요가 있다. 그러나 행동의 조합과 상대방의 반응까지 고려하면, 현실의 협상 문제는 매우 복잡한 **의사결정 문제**가 된다.

문제정의

본 프로젝트에서 다루고자 하는 협상 문제는 다음과 같이 명확하게 정의할 수 있다.

- 협상 참가자들은 각자의 턴(turn)에 다음과 같은 행동을 수행한다.
  - **협상 카드 선택** (협상 전략에 따라 특정 조건을 상대에게 제시)
  - **가격 입력** (상대방 제안에 대한 가격 제시로 응답)
- 협상의 목표는 주어진 조건 하에서 최적의 결과(최대의 가격 절감, 최상의 거래 조건 등)를 얻는 것이다.
- 이러한 행동의 선택은 상대방의 반응과 상호작용하면서 협상이 종료될 때까지 지속적으로 반복된다.

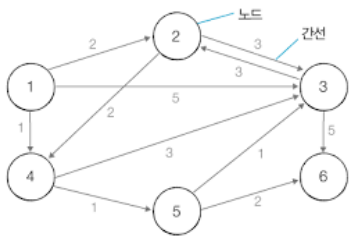
즉, 협상 과정은 **"턴 기반의 행동 선택 문제(turn-based action selection problem)"** 로 정의할 수 있다.

문제 추상화

복잡한 현실 협상 문제를 효과적으로 다루기 위해 다음과 같이 문제를 추상화할 수 있다.

- **상태(State)**: 각 턴의 특정한 상황 (예: 현재 선택된 협상 카드, 상대방이 제시한 가격의 조건 등)을 **노드(Node)** 로 표현한다.
- **행동(Action)**: 상태 간의 변화를 일으키는 행동(협상 카드 선택, 가격 입력 등)을 노드 간의 **간선(Edge)** 으로 표현한다.
- 이러한 방식으로, 협상 과정은 하나의 큰 그래프(graph) 형태로 표현될 수 있으며, 각 노드는 특정 상태를 나타내고 간선은 가능한 행동을 나타낸다.

즉, 본 협상 문제는 "각 상태가 노드이며, 각 행동이 노드 간을 연결하는 간선인 그래프 구조"로 명확하게 추상화할 수 있다.



문제 해결방안

그래프로 추상화된 협상 문제는 결국 **최적 경로 탐색 문제(optimal path finding problem)**로 환원될 수 있다.

- 각 노드(상태)에서 다음 노드로 이동하는 경로(행동 선택)에 따라 협상 결과가 달라진다.
- 따라서 목표는 이러한 그래프 구조에서 "최적의 결과를 얻을 수 있는 경로를 찾아내는 것"이다.

이러한 최적 경로를 찾기 위해 다양한 길찾기 알고리즘(pathfinding algorithms)을 적용할 수 있다. 이 중에서 협상 문제에 적용하기 위해서는 다음과 같은 특성이 요구된다.

- **통제 가능성(Controllability)**: 시스템 운영자가 경로 결정의 이유를 명확히 이해하고 조정 가능해야 한다.
- **검증된 성능(Validated Performance)**: 복잡한 상태 간 행동 선택에 대해 안정적이며 이미 성능이 입증된 방법론을 적용해야 한다.
- **학습 기반 개선 가능성(Improvability)**: 시간이 지남에 따라 데이터가 축적되면 경로 선택의 품질이 개선될 수 있어야 한다.

위 조건을 가장 잘 충족하는 알고리즘이 **Q-Table 기반의 Q-Learning 알고리즘**이다.

- Q-Learning은 상태-행동 매핑을 통해 각 상태에서 행동을 선택했을 때의 기대 보상을 명확히 제시하며 통제 가능성이 높다.
- 또한, Q-Learning은 이미 다양한 분야에서 성공적으로 적용되어 성능이 입증된 방법론이다.
- 축적된 협상 데이터를 기반으로 지속적으로 Q값을 업데이트하여 장기적으로 전략의 개선이 가능하다.

결론적으로 본 협상 문제는 **"그래프 기반의 의사결정 문제"로 추상화될 수 있으며, 이를 해결하는 데 가장 적합한 알고리즘으로 Q-Learning (Q-Table 기반)이 채택되었다.**

Q-Table

- 특정 상태에서 가능한 행동들을 수행 했을 때 기대되는 보상을 계산하여 다음 행동을 정하는 형태의 알고리즘

Q -Table 예시

	행동	**a1**	**a2**	**a3**	**...**	**an**
상태						
s1		2	4	1		2
s2		3	5	5		2
s3		1	3	4		6
...						
sn		1	2	6	7	1

Q-Table Update Rule

$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$

기호	의미
$Q(s,a)$	현재 상태 s에서 행동 a를 했을 때 기대되는 누적 보상
r	현재 행동에 따른 즉시 보상
s'	다음 상태
a'	다음 상태에서 가능한 행동들
γ	Discount factor (미래 보상에 대한 중요도)
α	Learning rate (얼마나 빨리 Q 값을 업데이트할지)

- 처음에는 전부 0으로 시작
- 환경을 탐험하면서 보상을 수집
- 그 보상과 미래에 받을 최대 보상을 더하는 형식으로 테이블을 업데이트

Policy Derivation

- Q-Table이 지정되면 단순히 현재 상태에서 가장 보상이 높은 행동을 선택

Q-Table 현재 상황에 맞게 활용

먼저 해당 알고리즘에서 사용되는 상태, 행동, 보상을 정의한다.

- 상태 : 각 조건에서의 협상카드
- 행동 : 특정 협상카드로 이동
- 보상 : 내부 목표가와 상대방의 제안한 가격의 비율

Q-Table 예시

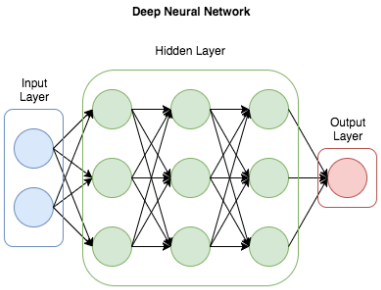
- 해당 Q-Table은 예시 Q-Table로 협상카드별로 시나리오와 상대방의 제안한 가격의 구간을 두는 조건을 추가하여 상태의 정의한 테이블
- 협상카드 : 16
- 시나리오 : 4(A,B,C,D)
- 가격 구간 : 3

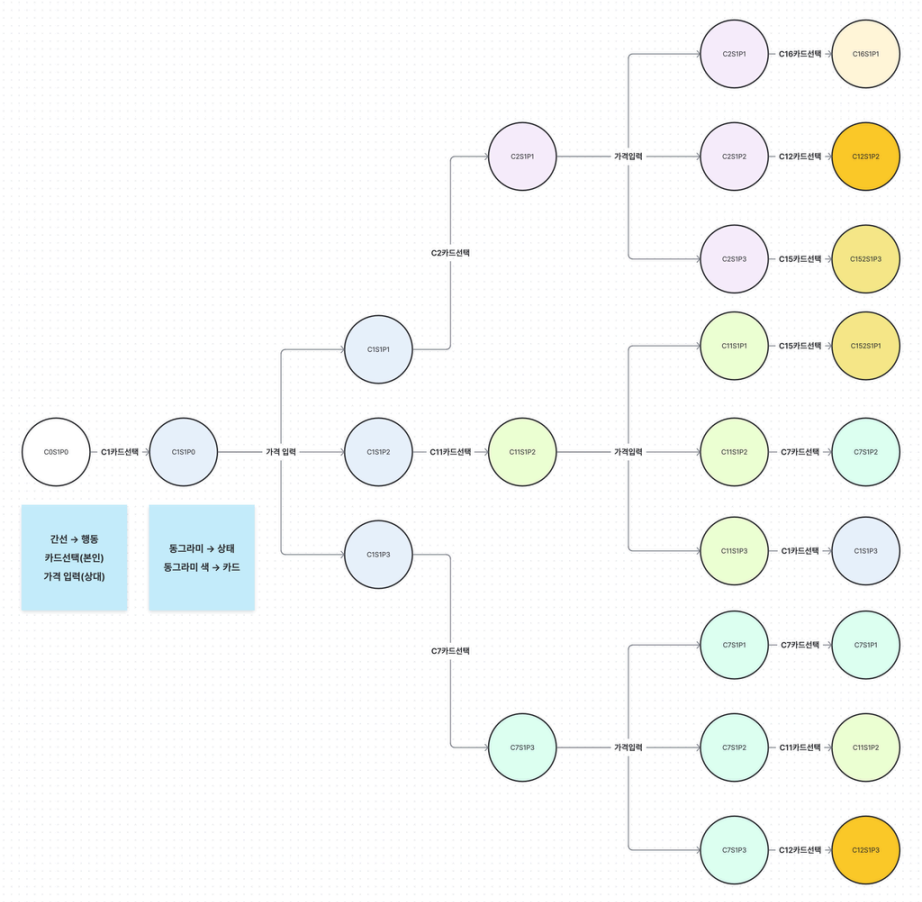
→ 총 상태 수 : 16 * 4 * 3 = 192

- 용어정리
  - →Cn : 카드n으로 이동
  - CnSnPn : n번카드 조건에서 n번 시나리오이면서 상대방이 n번 구간에 속하는 가격을 제시
- 참고
  - 초기 상태

- C0SnP0 : 0번카드 조건에서 n번 시나리오이면서 상대방에게 가격을 받지 않은 상태
- 보상함수
  - Q(s,a) \leftarrow Q(s,a) + \alpha \left[0 + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]
  - 이 때는 현재 보상이 0(상대방에게 가격을 못받았기 때문)
  - 하지만 미래 기대 보상에 대해 학습되기 때문에 여전히 유의미함

	행동	→C1	→C2	→C3	...	→Cn
상태						
C0S1P0		10	3	2		1
C1S1P1		2	4	1		2
C1S1P2		3	5	6		2
C1S1P3		1	3	4		6
...						
CnSnPn		1	2	6	7	1





Untitled whiteboard (2)

Oops, you've found a dead link.

- Go back to the previous page
- Go to the Home Page

- 위의 그림은 Q-Table의 각 Sell에 Q값이 정해진 이후에 이동 가능한 경우의 수를 간략히 트리 구조로 나타낸 예시
- 위의 그림에서 보면 Q-Table의 정의되지 않은 상태가 존재함 (C1S1P0)
  - 이러한 이유는 Q-Table에서는 우리가 행동(카드선택)하는 경우만 정의되기 때문
  - 상대방의 행동(가격입력)에 따라 지금 현재의 상태(맵)가 이동함
- 그림설명
  - 초기상태(C0S1P0) → Q-Table을 조회 → C1 선택 → 상태변경(C1S1P0)
  - 현재상태(C1S1P0)에서 상대방의 가격 입력 → 가격 구간에 따른 상태변경(C1S1P1)(P1 선택 가정)
  - 현재상태(C1S1P1) → Q-Table을 조회 → C2 선택 → 산태변경 (C2S1P1)
  - 현재상태(C2S1P1)에서 상대방의 가격 입력 → 가격 구간에 따른 상태변경(C2S1P2)(P2 선택 가정)
  - 현재상태(C2S1P2) → Q-Table을 조회 → C12 선택 → 산태변경 (C12S1P2)

LLM - Agent

서론

- Q-Table은 머신러닝 기법이기 때문에 기본적으로 LLM이 필요하지 않다
- 이러한 장점은 이전에 논의 되었던 *LLM 오류에 관한 불안을 해소할 수 있다
  (LLM 오류 : 어떠한 경우든 의도치 않게 LLM을 사용할 수 없게되는 경우)
- 하지만 이러한 장점은 반대로 학습을 해야 기능을 한다는 점이 존재한다
- 그러나 본 프로젝트에서는 학습할 데이터 및 시간이 부족하여 지금 당장의 콜드 스타트 문제를 해결해야 한다
- 따라서 본 알고리즘에서는 이를 해소하기 위해 LLM 기반의 Agent를 활용하여 콜드 스타트 문제를 해결하고자 한다

협상 전략

- 협상 전략이란 협상 카드의 순서 조합이다
- Q-Table에서의 협상 카드 순서 조합이란 *Q값이 정해진 Table을 의미한다
  (Q값 : 특정 상태에서 어떠한 행동을 했을 때 기대되는 보상)
- 따라서 본 알고리즘에서 협상 전략이란 Q값이 정해진 Q-Table을 의미한다
- 서론에서 언급 한 것 처럼 기본 Q-Table을 머신러닝 기법으로 학습이 되어야한지만, 현재로썬 불가능하다
- 따라서 본 알고리즘에서는 LLM Agent가 Q-Table 즉 협상 전략을 생성하는 것을 목표로 한다.

테스트

1. 테스트 목적

본 테스트의 목적은 **학습을 수행하지 않은 상태에서**, **LLM이 협상 카드 및 조건 정보를 바탕으로 Q-Table을 직접 생성할 수 있는지**를 평가하는 것이다.

2. 테스트 배경 및 필요성

- 기존 Q-Learning은 초기 Q값이 전부 0이기 때문에 **콜드 스타트 문제**가 존재함
- 협상 시스템에서 학습할 데이터나 시간이 충분치 않은 경우, **초기 Q-Table 생성이 성능에 결정적인 영향을 미침**
- 따라서 본 테스트에서는 **LLM이 사전 학습 없이 Q-Table을 생성해 전략적 출발점**을 제공할 수 있는지 실험하고자 한다

3. Q-Table 구성 조건

- **행동(Action)**: 카드 선택 → `→ Cn`
- **상태(State)**: 카드 조건 × 시나리오 × 가격 구간
  → 예: `C1S2P3`
- **총 상태 수**: 16장 카드 × 4개 시나리오 × 3개 가격 구간 = **192**
- **Q값 초기 조건**: LLM이 생성한 값을 기준으로 초기화됨

4. 테스트 입력

- 협상 카드의 설명 또는 메타 정보
- 시나리오 별 특징 (예: A = 공급사 강세, B = 구매자 우위 등)
- 가격 구간의 정의 (예: P1 = 제시가가 목표가의 90% 이상 등)

5. 테스트 목표

항목	설명
Q-Table 생성 가능성	LLM이 상태-행동 매핑에 대해 초기 Q값을 유의미하게 생성할 수 있는가
전략 적합성	생성된 Q-Table이 직관적이거나 합리적인 협상 흐름을 유도하는가

다양성	다양한 카드 조건/시나리오/가격 구간 조합에 대해 균형 있게 Q값이 분포되는가

6. 테스트 방법

1. **LLM Prompt Template 구성**
   - 카드 설명, 시나리오 설명, 가격 구간 설명을 입력
   - 예시: “다음 조건을 바탕으로 상태-행동 Q-Table을 생성하세요…”
2. **LLM 응답으로부터 Q-Table 추출**
   - Q값은 [0, 10] 등의 스케일 내에서 정규화된 수치로 요청
3. **도출된 Q-Table을 평가**
   - 내부검토