게임으로 배우는 알고리즘

이 재명

어떤 이야기를 할까요?

- 게임 프로그래밍에서 자주 사용되는 대표적인 알고리즘 두 가지
  - 게임을 예로 들었지만, 게임 이외의 분야에서도 널리 쓰인다.

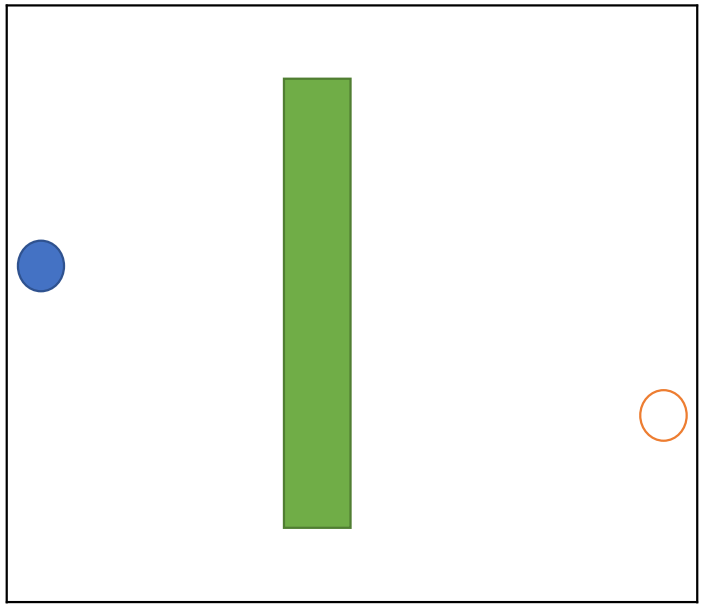
- StarCraft와 A*
  - 2D/3D 게임 프로그래밍에서의 길 찾기 문제

- AlphaGo와 MCTS(Monte Carlo Tree Search)
  - 게임 AI 프로그래밍에서의 최적의 수를 찾아내는 문제

StarCraft와 A*

A*?

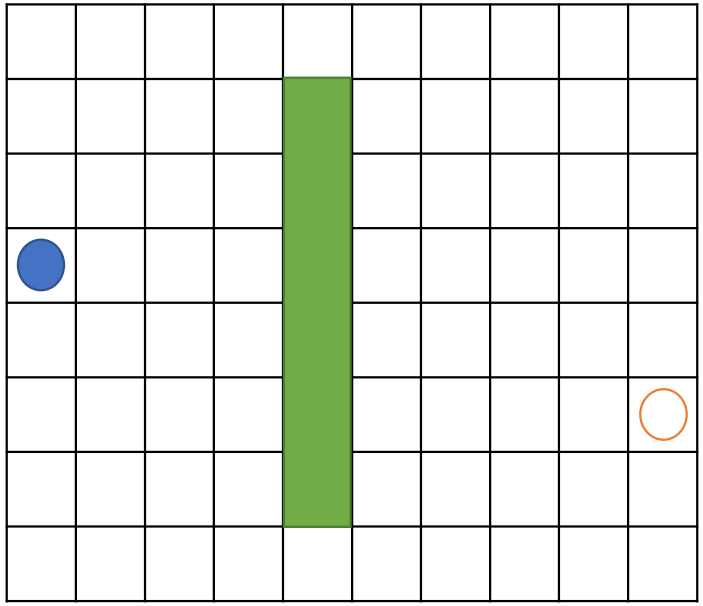
- 어떤 게임에서 다음과 같이 파란색 유닛이 있다고 하자.



- 이 유닛이 초록색 벽을 지나 주황색 지점까지 가려면 어떻게 이동해야 하는가?

A*? (Cont.)

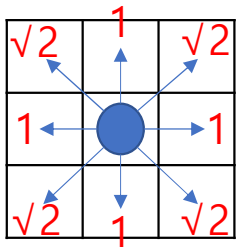
- 실제 내부적으로는 다음과 같이 격자(grid) 형태의 공간으로 관리될 것이다.



A*? (Cont.)

- 이 문제를 우리가 배웠던 Dijkstra 알고리즘으로도 풀 수는 있다.

- 그러나 Dijkstra 알고리즘은 다음과 같은 여덟 가지 이동 경로에서 거리 외에 다른 것을 전혀 고려하지 않는다.



A*? (Cont.)

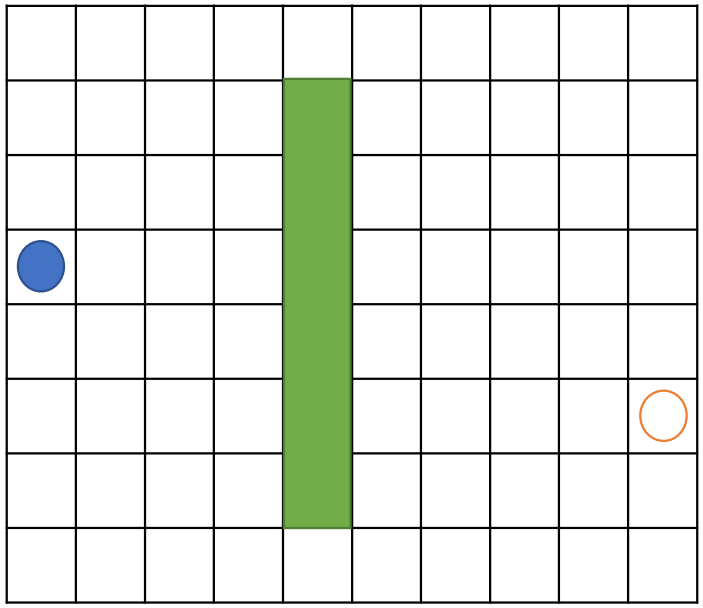
- 다시 말해, 목적지의 위치에 따라 전혀 엉뚱한 경로의 탐색을 모두 마친 다음에 비로소 목적지로 가는 경로 쪽으로 탐색하게 될 가능성이 높다.

- A*는 Dijkstra 알고리즘에서 합계 거리를 평가할 때, 우리가 알고 있는 추정값을 더해서 계산하는, Dijkstra 알고리즘의 확장이다.

- 이 추정값을 휴리스틱(heuristic)이라고 부른다.

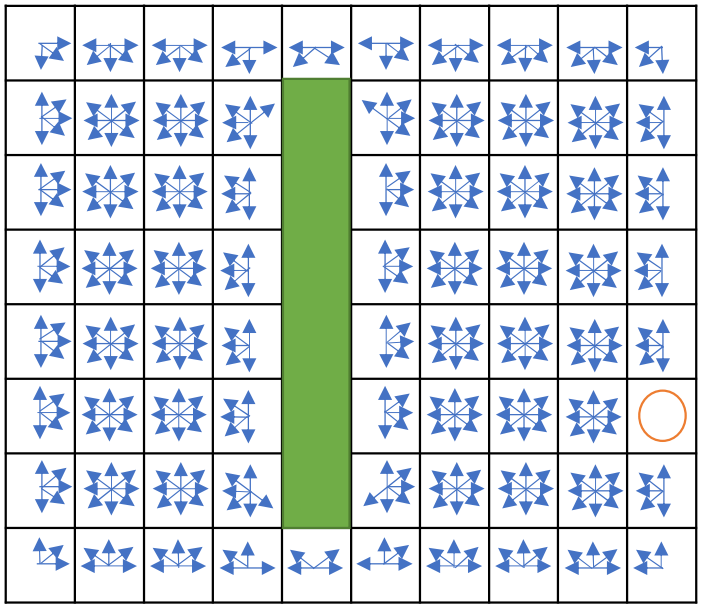
A*? (Cont.)

- 이전의 문제로 다시 돌아와 보자.



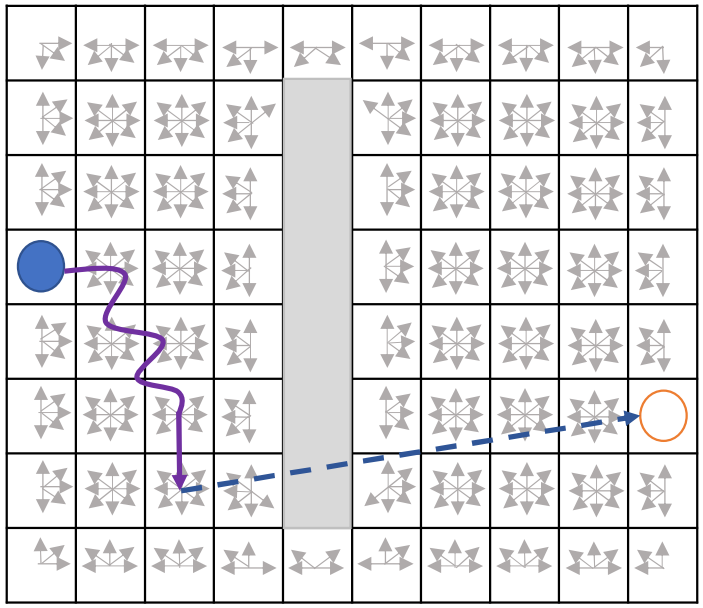
A*? (Cont.)

- 이 문제를 단순히 인접 셀 간의 이동 경로만 생각하면 다음과 같은 그래프 문제가 된다.



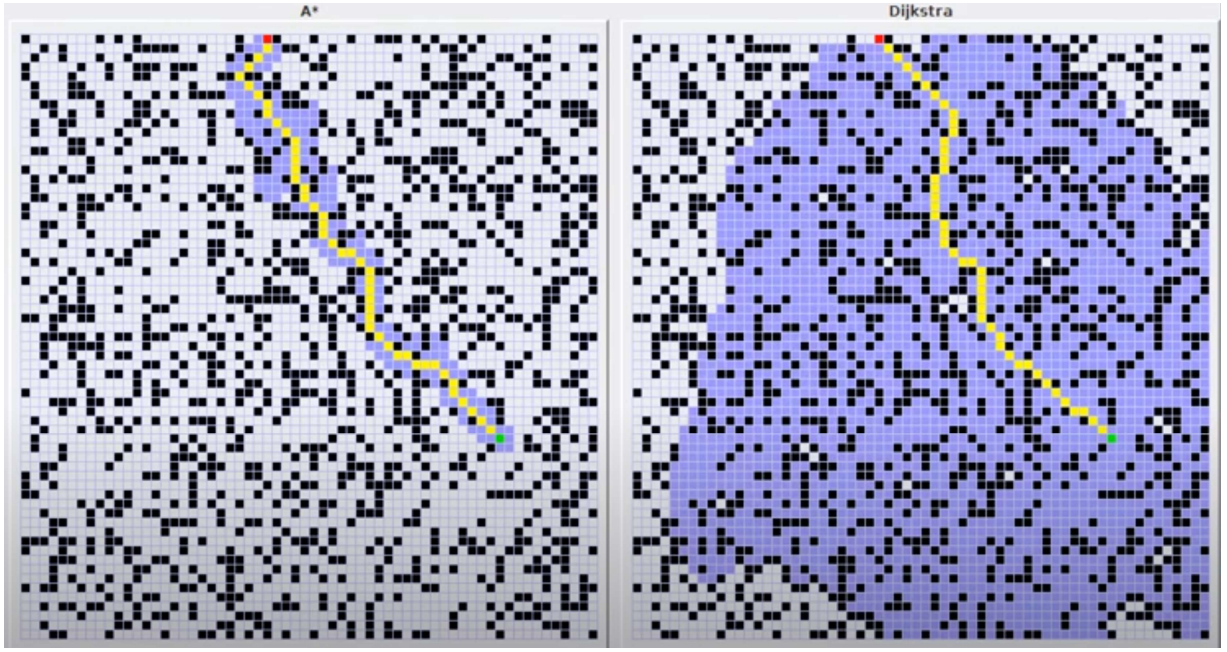
A*? (Cont.)

- 이 문제에 A*를 적용하면, 거리를 평가할 때, 우리가 알고 있는 합계 거리와 유클리드 거리로 추정한 휴리스틱 값을 더해서 평가한다.



A*? (Cont.)

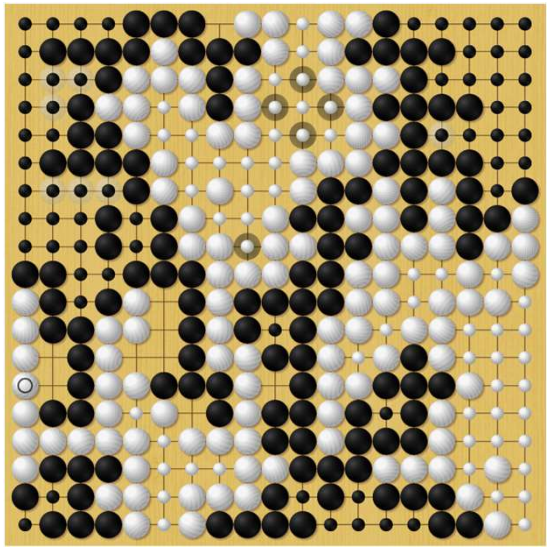
- 효과
  - 직선 거리에 더 가까운 경로를 우선적으로 탐색한다.
  - 대부분의 경우, Dijkstra 알고리즘보다 최단 경로를 훨씬 빠르게 찾는다.



AlphaGo와 MCTS

바둑(Go)?

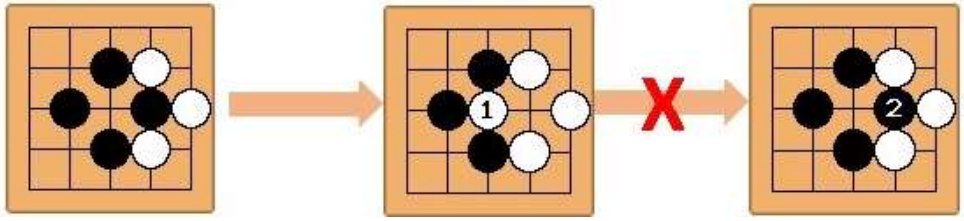
- 최소 2500년보다도 더 전에 중국에서 만들어진 보드 게임이다.
  - 중국 고전《논어(論語)》에 ' 혁(弈) ' 이라는 이름으로 등장한다.
- 흑과 백이 번갈아가며 두며, 게임이 끝나면 서로의 집(영역)을 계산해 집이 더 많은 쪽이 승리한다.



바둑(Go)? (Cont.)

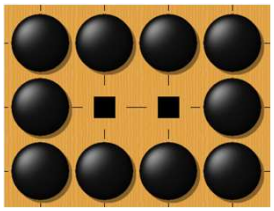
- 기본 규칙
  - 상대방의 돌을 둘러싸면, 그 돌을 들어내고 잡은 것으로 한다.
    - 잡은 돌은 하나당 1집의 가치를 가진다.

  - 패(Ko) 규칙 – 바로 이전 장면과 동일한 장면으로 만드는 수는 금지된다.

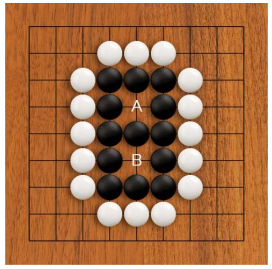


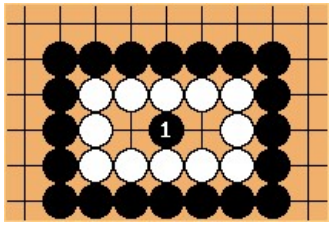
바둑(Go)? (Cont.)

- 계가(Scoring) – 집 세기
  - 바둑을 다 두었으면, 서로의 집을 계산한다.
  - 둘러싼 빈 공간 한 칸마다 1집으로 하여 센다.



  - 잡은 돌은 하나당 1집으로 하여 센다.
  - 사활(Life and Death)
    - 양 플레이어의 동의 하에, 무조건 잡을 수 있는 돌은 죽은 돌로 보아 잡은 것으로 간주.





바둑의 기력 체계

- 아마추어의 기력 체계
  - 급(級)
    - 18급부터 시작
    - 기력이 올라갈수록 급수 숫자가 낮아짐
  - 단(段)
    - 아마추어 자격증에 의해 공식적으로 인정되는 기력
    - 초단(1단) ≒ 1급
    - 기력이 올라갈수록 숫자가 높아짐
    - 협회에 따라 아마6단 또는 아마7단이 최고 등급
- 프로의 기력 체계
  - 초단(初단)부터 구단(九단)까지 존재
  - 기력이 올라갈수록 숫자가 높아짐

Elo rating

- Elo rating
  - 체스와 같은 2인용 게임의 실력 평가 모델
  - https://www.goratings.org/en/

Rank	Name	♂♀	Flag	Elo
1	Shin Jinseo	♂		3830
2	Ke Jie	♂		3724
3	Park Junghwan	♂		3722
4	Gu Zihao	♂		3664
5	Ding Hao	♂		3637
6	Byun Sangil	♂		3630
7	Mi Yuting	♂		3626
8	Yang Dingxin	♂		3623
9	Iyama Yuta	♂		3587
10	Fan Tingyu	♂		3574

  - 수학적으로 상위 랭커가 하위 랭커를 이길 확률

Elo Differnce	Probability	Elo Differnce	Probability
10	51.44	340	87.62
20	52.88	350	88.23
30	54.31	360	88.82
40	55.73	370	89.38
50	57.15	380	89.91
60	58.55	390	90.42
70	59.94	400	90.91
80	61.31	410	91.37
90	62.67	420	91.82
100	64.01	430	92.24
110	65.32	440	92.64
120	66.61	450	93.02
130	67.88	460	93.39
140	69.12	470	93.74
150	70.34	480	94.06
160	71.53	490	94.38
170	72.68	500	94.68
180	73.81	510	94.96
190	74.91	520	95.23
200	75.97	530	95.48
210	77.01	540	95.72
220	78.01	550	95.95
230	78.98	560	96.17
240	79.92	570	96.38
250	80.83	580	96.57
260	81.71	590	96.76
270	82.55	600	96.93
280	83.37	650	97.68
290	84.15	700	98.25
300	84.90	750	98.68
310	85.63	800	99.01
320	86.32	850	99.26
330	86.98	900	99.44

체스(Chess) vs. 바둑(Go)

- 체스(Chess)
  - 게임 트리 크기: $\approx 10^{123}$
  - 1997년에 IBM의 Deep Blue가 세계 최고 챔피언인 Garry Kasparov를 이기면서 인간의 수준을 뛰어넘었다.

- 바둑(Go)
  - 게임 트리 크기: $\approx 10^{360}$
  - 2016년에 Google DeepMind의 AlphaGo가 세계 최고 챔피언인 이세돌 九단을 이기면서 인간의 수준을 뛰어넘었다.

바둑이 컴퓨터에게 왜 어려운가?

- Huge search space
  - 체스는 매 장면마다 둘 수 있는 수가 평균적으로 35가지
  - 바둑은 매 장면마다 둘 수 있는 수가 평균적으로 250가지
  - 체스는 search space가 작아서 alpha-beta pruning 같은 알고리즘으로도 충분.

- Evaluation function
  - 체스는 state가 주어지면 판의 유불리를 approximate할 수 있는 수학적 모델이 존재한다.
  - 바둑은 사활을 판단하는 알고리즘이 없어서 evaluation function이 없다.

AlphaGo 이전의 바둑 AI

- 1968 – 최초의 컴퓨터 바둑 프로그램 등장
  - Albert Lindsey Zobrist, “Feature Extraction and Representation for Pattern Recognition and the Game of Go,” University of Wisconsin (Ph.D. Thesis), 1970
  - Zobrist hashing을 고안 - “동일한 국면”을 감지하는 알고리즘
  - 수준: 막 입문한 사람보다도 못한 수준

- 1998 – 높은 핸디캡을 받고 인간을 이긴 프로그램 등장
  - 아마추어 강자를 상대로 25–30점 접바둑에서 승리
  - 수준: 막 입문한 사람 정도의 수준

AlphaGo 이전의 바둑 AI

- 2004 – GNU Go 3.6
  - 인터넷 바둑 사이트(KGS)에서 12k~13k 정도로 평가됨
- 2006 – Crazy Stone (Rémi Coulom)
  - 최초로 MCTS 알고리즘 채용
  - AlphaGo 알고리즘에 영향을 준 바둑 AI
- 2006 - MoGo (Wang, Gelly, Munos, . . .)
  - 9줄 바둑에서 프로를 이김 (2008)
  - 19줄 바둑에서 9점 바둑으로 프로를 이김 (2008)
  - AlphaGo 알고리즘에 영향을 준 바둑 AI

AlphaGo 직전의 바둑 AI

- Crazy Stone (Rémi Coulom, 프랑스)
- Zen (오지마 요지, 가토 히데키, 일본)

- 위 두 프로그램이 컴퓨터 바둑 대회를 우승하고 있었다.
- 2013년에 Crazy Stone이 이시다 요시오 九단을 19줄 4점 접바둑에서 이기면서 프로와 4점 치수로 인정되었다.
- 당시 사람들은 컴퓨터 프로그램이 프로 기사를 이기려면 최소 10년 이상이 걸릴 것으로 내다 보았다.

AlphaGo의 역사

- 2015: DQN(Deep Q-Network)
  - Deep Learning + Reinforcement Learning (Q-Learning)
  - 아타리 2600 게임들을 인간 수준으로 플레이

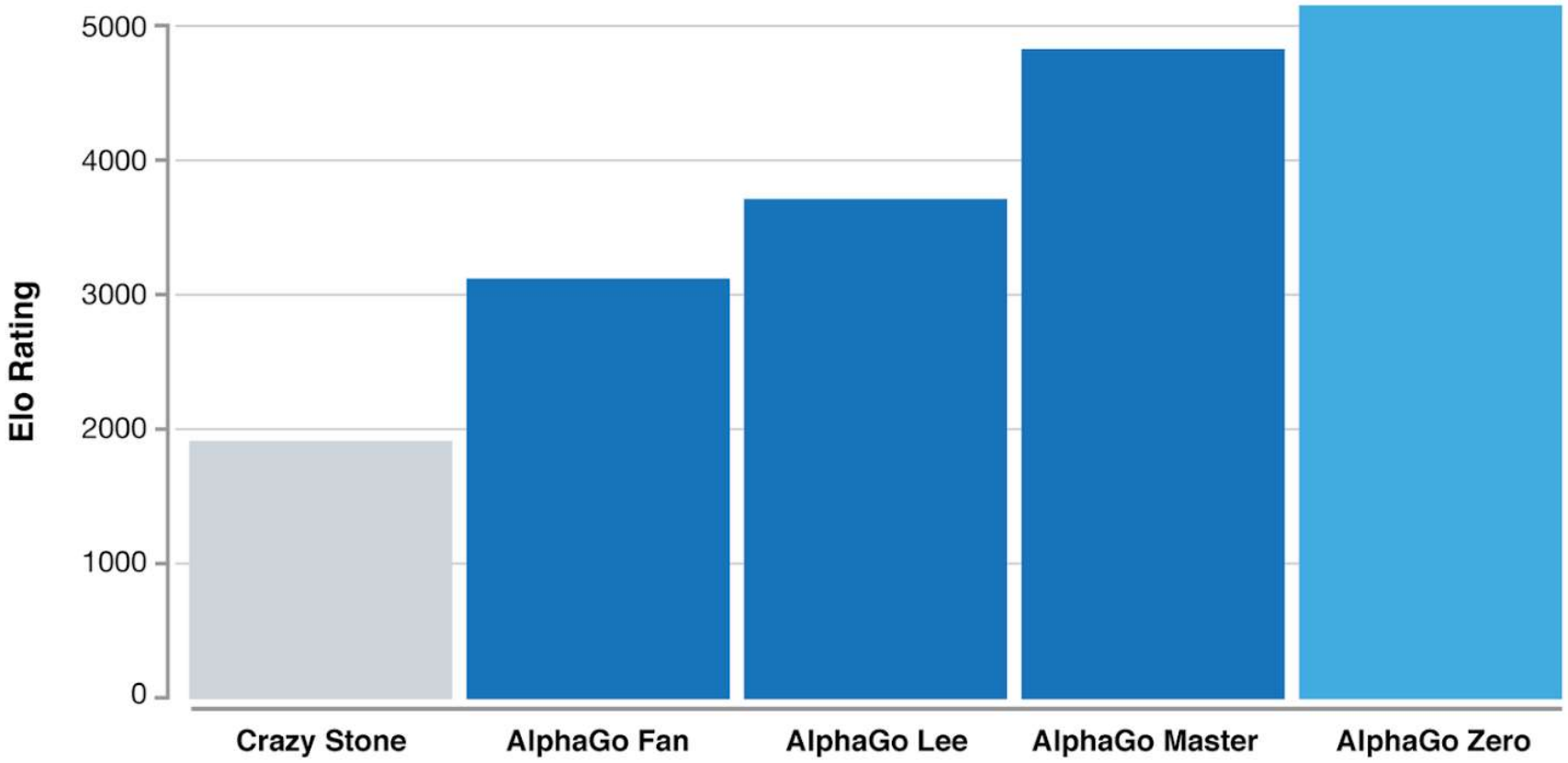
- 2015: AlphaGo Fan
  - 판후이(樊麾, 二단): 중국 태생의 프랑스인 프로 바둑 기사
  - 유럽 바둑 챔피언 판후이를 상대로 5-0으로 승리
  - David Silver et al., “Mastering the game of Go with Deep Neural Networks & Tree Search,” Nature, 2016

AlphaGo의 역사

- 2016: AlphaGo Lee
  - 이세돌 九단을 상대로 4-1로 승리
  - AlphaGo Fan을 개량한 버전에 병렬 컴퓨팅 동원(약 1900대 컴퓨터)

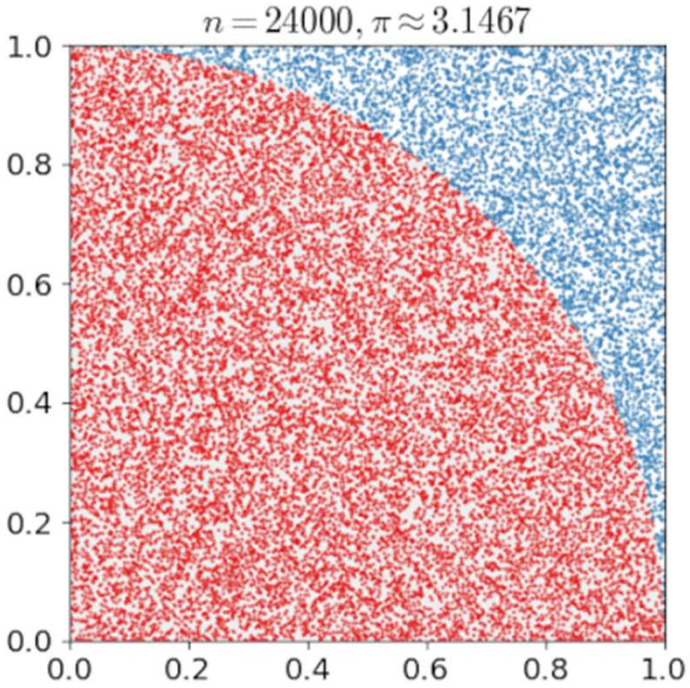
- 2017: AlphaGo Master
  - AlphaGo Lee 때의 버그를 수정하기 위해 신경망 설계와 강화학습 방식 변경
  - 세계 1위 기사인 커제를 상대로 3-0으로 승리

- 2017: AlphaGo Zero
  - 인간의 바둑에 대한 정보 없이도 AlphaGo Master보다 강한 AI


Elo Rating
5000
4000
3000
2000
1000
0
Crazy Stone
AlphaGo Fan
AlphaGo Lee
AlphaGo Master
AlphaGo Zero

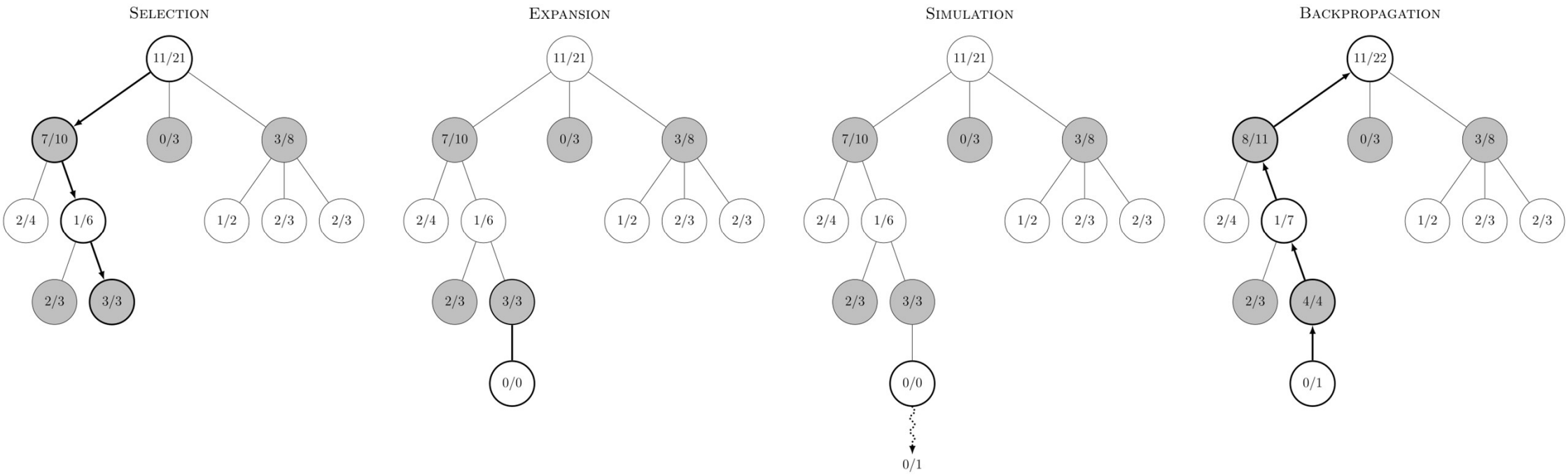
MCTS(Monte Carlo Tree Search)

- 몬테카를로 방법(Monte Carlo method)
  - 랜덤 샘플링을 통해 전체 확률을 추정하는 방법



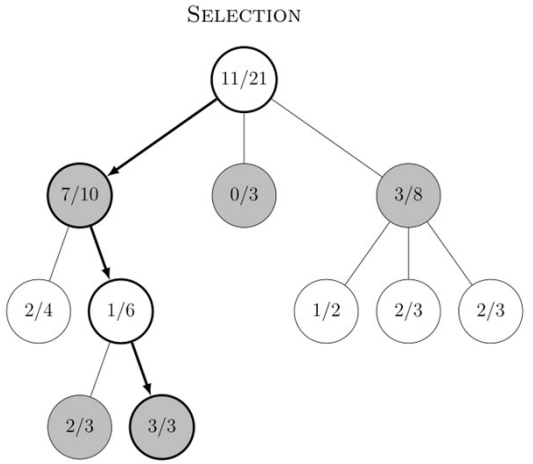
MCTS(Monte Carlo Tree Search) (Cont.)

- MCTS는 몬테카를로 방법에 game tree search를 결합한 알고리즘



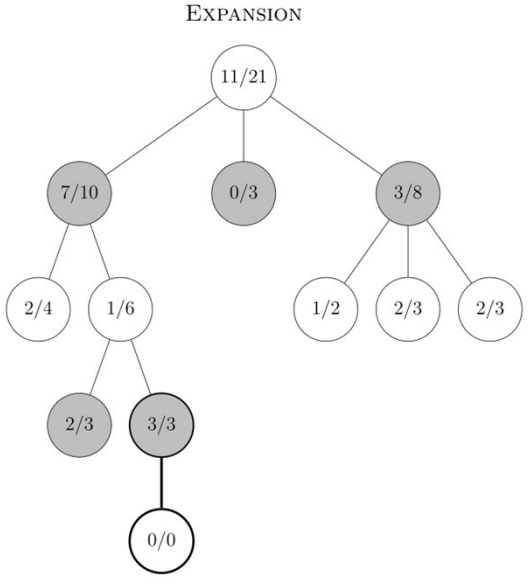
MCTS(Monte Carlo Tree Search) (Cont.)

- Selection 단계
  - Leaf node에 도달할 때까지 수읽기할 노드를 선택하는 단계
  - “어떻게 node를 선택하는가”
    - UCB를 비롯한 많은 알고리즘들이 나와 있음
    - UCB(Upper Confidence Bounds)
      - 승률 + sqrt((ln(parent.visits)) / (5 * visits))



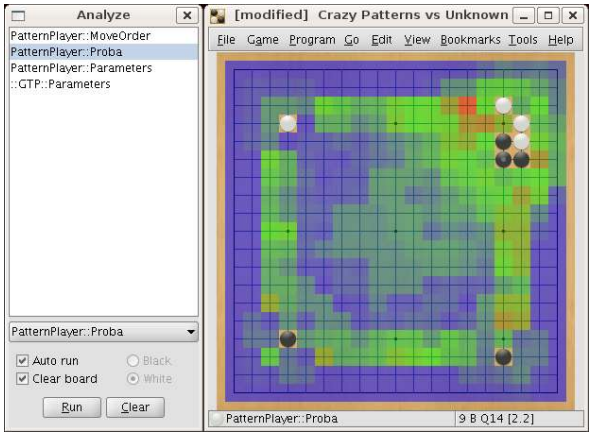
MCTS(Monte Carlo Tree Search) (Cont.)

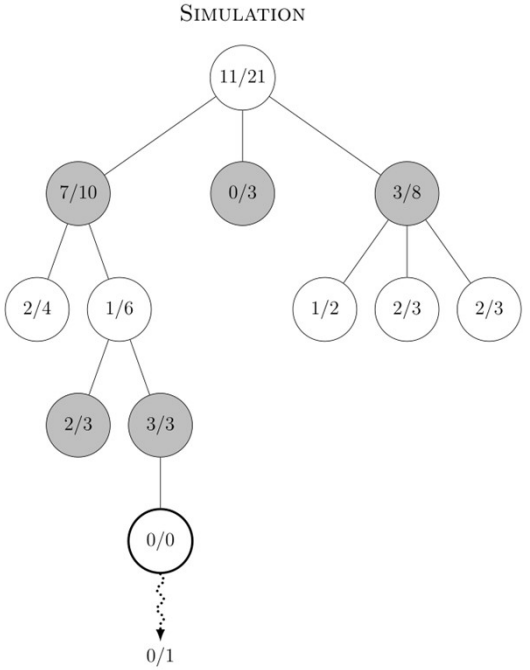
- Expansion 단계
  - 선택한 leaf node의 child nodes를 expand
  - 메모리 공간 절약을 위해, 항상 여는 것이 아니라 visits 수가 특정 threshold 값 위에 있을 때만 확장
    - 프로그래머가 정하는 값 (예: 10)



MCTS(Monte Carlo Tree Search) (Cont.)

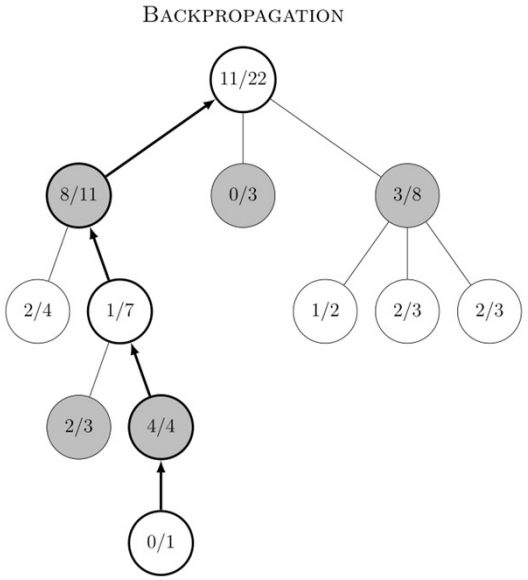
- Simulation 단계
  - 선택한 leaf node의 이후부터 게임의 결과가 나올 때까지의 상황을 랜덤에 의한 시뮬레이션으로 결과를 산출한다. (승리 or 패배)
  - 단순 랜덤에 의한 시뮬레이션보다는 간단한 Machine Learning 테크닉을 사용한 policy 확률 분포를 사용하는 경우가 많음





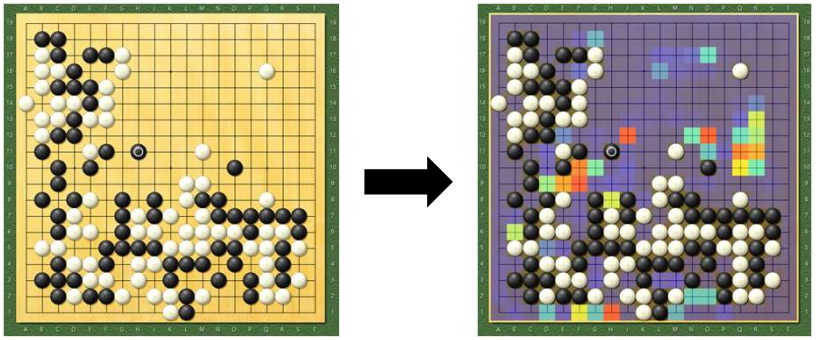
MCTS(Monte Carlo Tree Search) (Cont.)

- Backpropagation 단계
  - Simulation 단계에서 얻어진 결과를 root node에까지 누적시킨다.

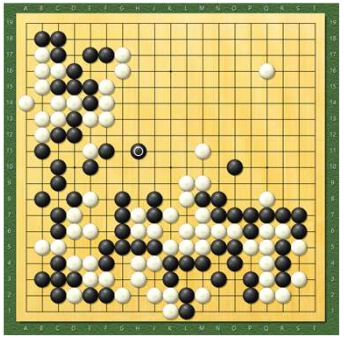


AlphaGo Zero

- Neural Network
  - Policy network + Value network를 하나로 통합 (두 개의 output head)
  - Policy network



  - Value network



64% (백이 이길 확률)

AlphaGo Zero (Cont.)

- MCTS
  - Simulation 단계를 Value network의 output으로 대체하였다.

- Self-play reinforcement learning
  - Self-play 게임은 1,600 MCTS simulations으로 생성됨
  - Self-play 시 root node에 Dirichlet noise 추가
    - 새로운 수를 탐색하도록 함