Practical Massively Parallel MCTS Applied to Molecular Design

생성일	@Feb 2, 2021 4:18 PM
속성	
태그	2021 ICLR MCTS

Xiufeng Yang, Tanuj Kr Aasawat, Kazuki Yoshizoe

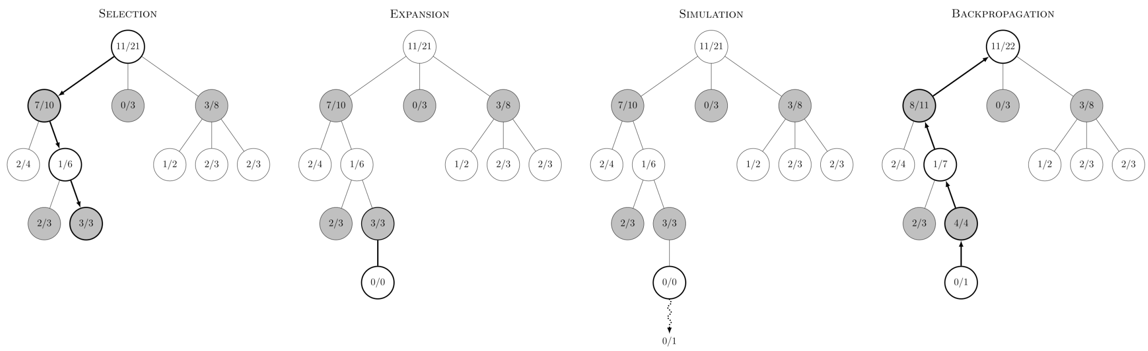
RIKEN Center for Advanced Intelligence Project

MCTS의 목적

- 제한된 시간내, 해 집합 내, 좋은 해를 탐색하는 것
- 왜 필요? combinatorial optimization or planning problem in vast **[molecule, …]** space

개요

- MP(Massively Parallel)-MCTS algorithm
    - 기존 work은 100 worker scale(shared-memory single machine environment)
        - → 1,000 worker scale(distributed memory environments)
    - non-parallel MCTS 로 42시간 걸릴 작업을 10분만에(256-cpu)
- Contributions
    - First work that applies distributed MCTS to a real-world and non-game problem
    - 간단한 모델이 MCTS를 만나 고도화된 모델을 능가할 수 있음을 보임
- [BACKGROUND] MCTS process: selection-expansion-rollout-backpropagation



- **selection**: 루트 노드 R 에서 시작해서 leaf node L 까지 선택
- **expansion**: 게임이 끝나지 않은 경우 해당 노드로 부터 새로운 노드를 생성
- **rollout**: run simulation (self-play)
- **backpropgation**: rollout된 결과로부터 root 노드까지의 상태를 업데이트

▼ why Parallelizing MCTS is challenging?

- rollout-backpropagation
  - subsequent selection steps depends on the results of the previous rollouts-backpropagation
- communication delay

▼ [BACKGROUND] UCT(UCB applied to Trees) with Virtual-loss

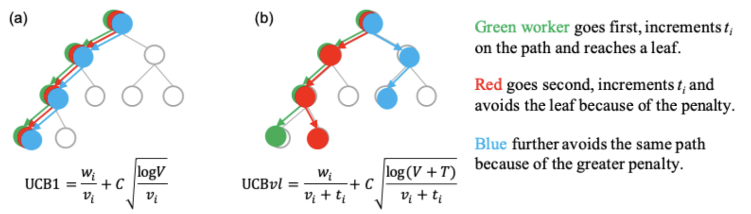


Figure 2: (a) parallel UCT using UCB1 (failed) (b) parallel UCT with *virtual loss*, and the search paths of three parallel workers shown in solid circles, (green, red, and blue, from left to right).

v_i: number of visits

w_i : cumulative reward

C: controlling exploitation and exploration

V: the total number of visits to all children

t_i: 자식의 하위 트리에서 현재 검색중인 worker수

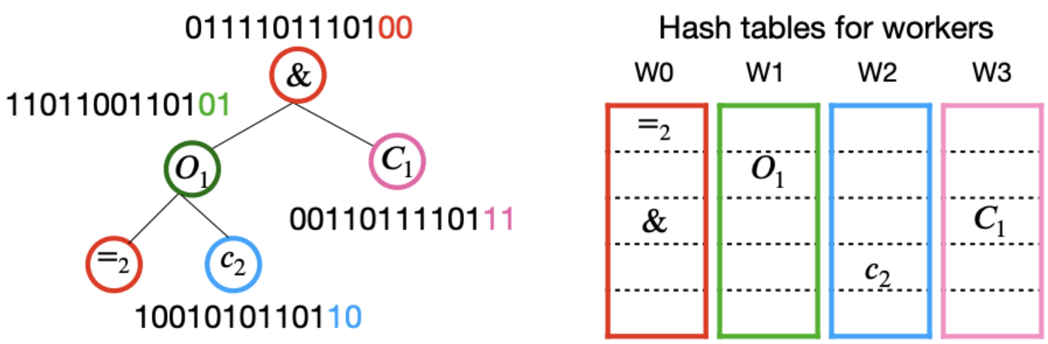
T: sum of t_i of the children

- UCB (Upper Confidence Bound)

- node selection policy
- optimistic estimation
- $\frac{w_i}{v_i}$: cumulative reward (exploitation)
- $\sqrt{\frac{log(V)}{t_i}}$: uncertainty (exploration)

- an approach to planning that converges asymptotically to the optimal policy in single
agent domains and to the minimax value function in zero sum games
  - Multi-armed bandits with episode context, Christopher D Rosin, 2011

▼ [BACKGROUND] Hash driven parallel search



- TDS(Transposition-table Driven Scheduling)
- communication overhead

방법

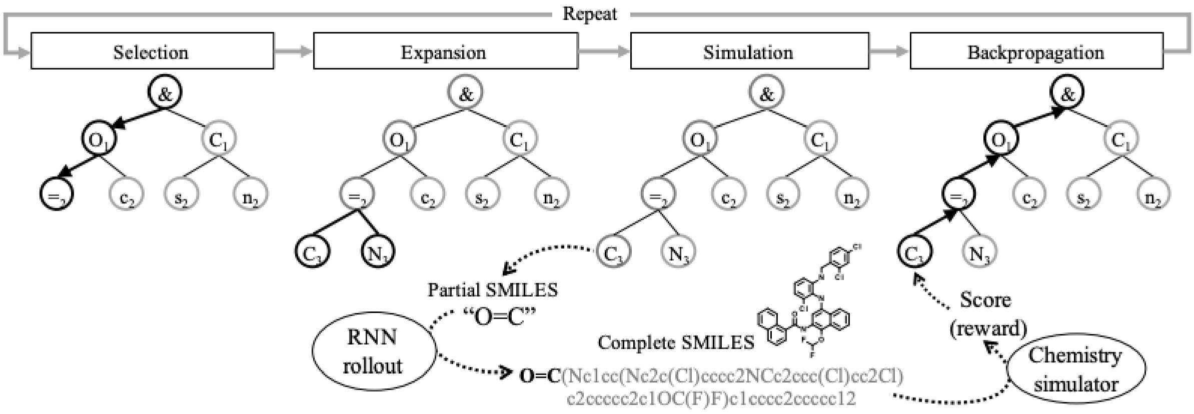
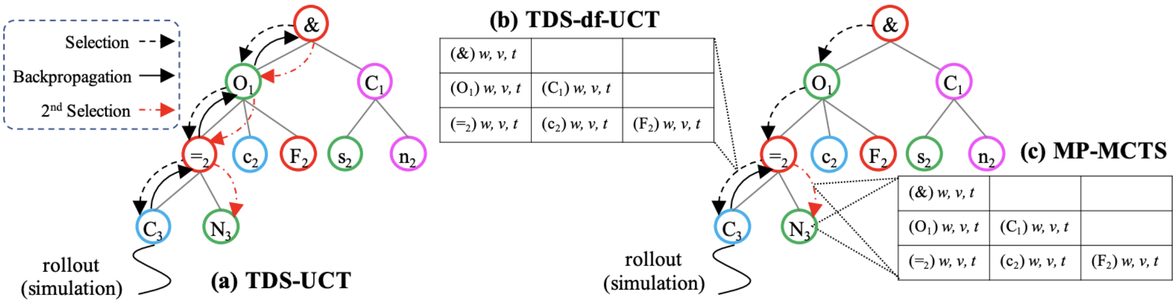


Figure 1: Four steps of (non-parallel) MCTS, with simulation for molecular design.

- **Molecular design on MCTS**



▼ [BACKGROUND]**TDS-UCT**: HASH-DRIVEN PARALLEL MCTS (*2011, Scalable distributed monte-carlo tree search, Yoshizoe et al*)

- **selection**: sends a selection message, ex) root $\rightarrow O_1$(green), If a worker receives a selection message, it selects the best child of the node and pass the message to another worker until the message reaches a leaf node. (worker-count t_i of each node is incremented during the step.)
- **expansion**: same as in MCTS
- **rollout**: done by home processor of the leaf
- **backpropagation**: The workers pass the backpropagation message(with the reward) to the parent until it reaches the root node
- **Scalability is limited**: root node 주변의 communication 경합, 메시지가 증가할수록 상위노드는 communication에 더 많은 시간을 사용, 100 worker를 넘어서면 확장성이 빠르게 감소

▼ [BACKGROUND]**TDS-DF-UCT**: DEPTH-FIRST REFORMULATION OF TDS-UCT (*2011, Scalable distributed monte-carlo tree search, Yoshizoe et al*)

- **Observation**: promising part of the tree does not change so often after each simulation
- Branches are sorted such way that the left-most branch is the best
- A leaf is expanded when the number of visits reaches a preset threshold. otherwise, 2nd selection step is started.
- The next selection step will likely reach a node that is close to the previously selected leaf node
- **Each job message** contains the history table which contains the history of the siblings of the nodes in the current path.
- Unlike TDS-UCT, TDS-df-UCT will backpropagate only if the UCBvl value of the nodes in the current path is exceeded by one of the other siblings in the history table.
- 다른 worker가 트리에서 더 유망한 부분을 발견하여 propagation을 자주 건너뜀 → tree가 얕고 넓어지는 현상

▼ **MP-MCTS**: ADDRESSING SHORTCOMINGS OF TDS-DF-UCT

- 각 node 는 자신의 history table을 유지함
- 최신 정보의 propagation을 가속화
- TDS-UCT는 t 만 저장

Experiments

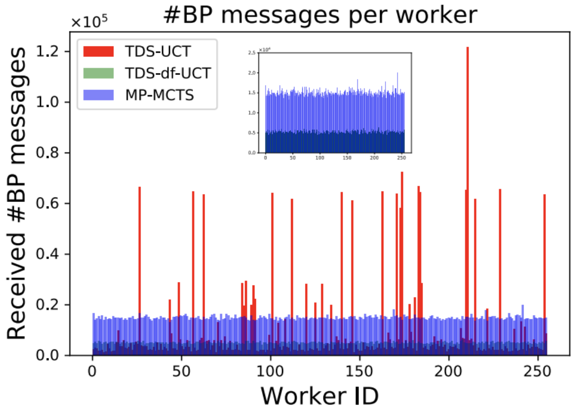
▼ **Setup**

- J(S): logP - (SA score + large Ring score)
- reward: valid- J(S)/(1+J(S)) otherwise, -1.0
- 2-layer GRU (256 dim), SMILES (64 dim)
- add branch - 누적 확률 0.95
    - rollout to generate complete SMILES
- MPI (Message Passing Interface) - mpi4py

- run for 10 minutes on up to 1024 cores of a CPU cluster
- a worker is assigned to one core

- **Result**

Methods \ cores	4	16	64	256	1024
TDS-UCT	5.83±0.31	6.24±0.59	7.47±0.72	7.39±0.92	6.22±0.27
TDS-df-UCT	7.26±0.49	8.14±0.34	8.59±0.49	8.22±0.41	8.34±0.46
MP-MCTS	**6.82±0.76**	**8.01±0.61**	**9.03±0.85**	**11.46±1.52**	**11.94±2.03**
⋆non-parallel-MCTS (#cores × 10 minutes)	6.97 ± 0.49	8.54 ± 0.34	9.23 ± 0.53	11.17 ± 0.88	–



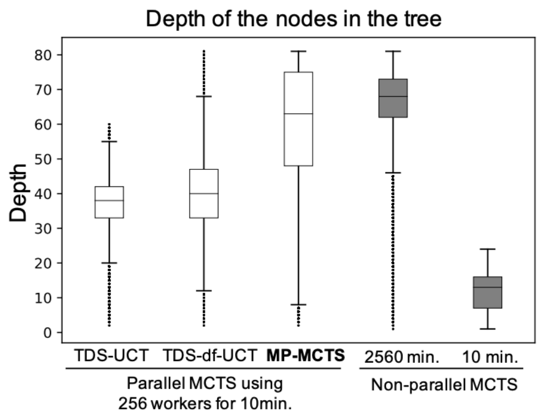


Table 2: Comparison of the best three penalized logP scores

Methods	1st	2rd	3rd
JT-VAE (2018) (Jin et al., 2018)	5.30	4.93	4.49
GCPN (2018) (You et al., 2018a)	7.98	7.85	7.80
Mol-CycleGAN (2020) (Maziarka et al., 2020)	9.76	7.29	7.27
MolecularRNN (2019) (Popova et al., 2019)	10.34	10.19	10.14
GRU-based (Yang et al., 2017)	6.47	5.65	5.01
MP-MCTS using GRU	**15.13**	**14.77**	**14.48**