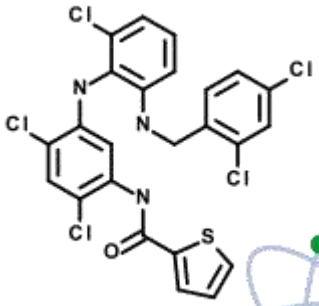
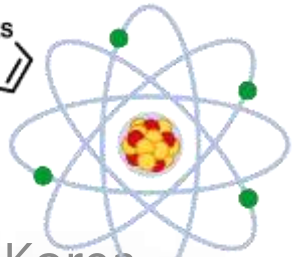
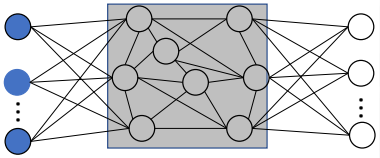
Deep Learning and Tree Search Finds New Molecules

Kazuki Yoshizoe
Search and Parallel Computing Unit, RIKEN AIP

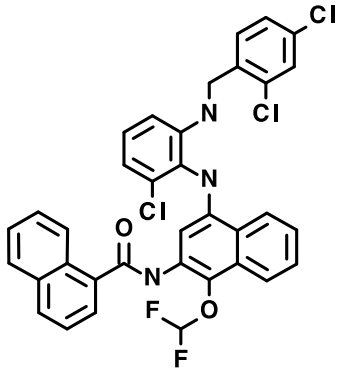




Feb. 24, 2019
The Second Korea-Japan Machine Learning Workshop
February 22 (Fri) - 24 (Sun), 2019, Haevichi Hotel/Resort, Jeju, Korea

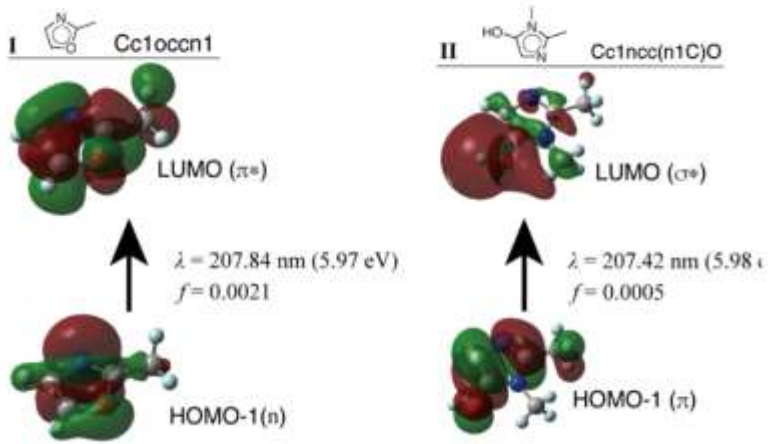


de novo Molecular Generation



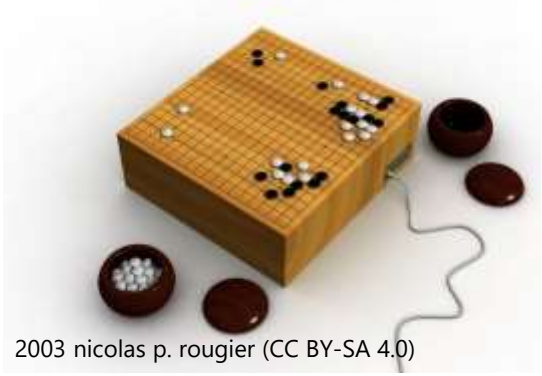
Discovering new molecules

which has high “score”.



This problem is similar to the game of Go

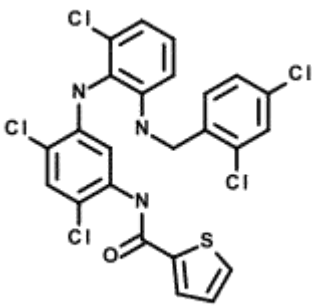
in our formulation



String Representation of Molecules

Need to define a search space
of molecules to apply AlphaGo approach

could be…
- graph based ?
- grammar based ?
- string based ?



According to chemists,
there are approx. 10^{60}
candidates of molecules

simple idea
using string like,
"H-O-H" is water

chess
10^{45}

Go
10^{170}

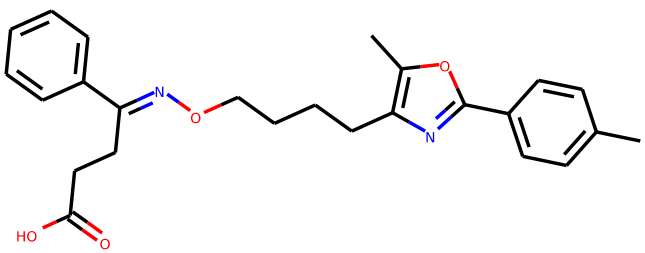
cf. Game
search space size

Actually chemists have
there own sophisticated way
of string based representation

SMILES

Simplified Molecular-Input Line-Entry System

O	Water (H and single bond omitted)
O=C=O	Carbon dioxide
N#N	Nitrogen
c1=cc=cc=c1	Benzene (c1 and c1 connect)
[Cu+2].[O-]S(=O)(=O)[O-]	Copper sulfate



Cc3ccc(c2nc(CCCCO/N=C(CCC(O)=O)c1ccccc1)c(C)o2)cc3

Defined based on the following grammar
Each symbol mean Atoms / Bonds / Rings

Atom: {C, c, o, O, N, F, [C@@H], n, -, S,Cl, [O-],[C@H], [NH+],[C@], s, Br, [nH], [NH3+], [NH2+], [C@@], [N+], [nH+], [S@], [N-], [n+],[S@@], [S-], I, [n-], P, [OH+],[NH-], [P@@H], [P@@], [PH2], [P@], [P+], [S+],[o+], [CH2-], [CH-], [SH+], [O+], [s+], [PH+], [PH], [S@@+] }
Bonds: {/,=, ¥# }
Ring: {1,2,3,4,5,6,7,8,9}
Branch: {(,)}

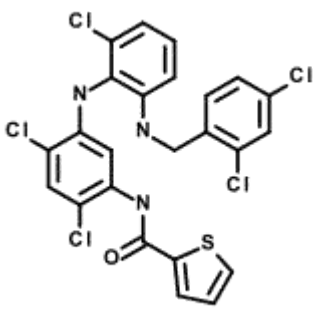
Note:
- Correct grammar does not guarantee valid molecules
- Does not cover all possible molecules
- Canonical SMILES can be defined

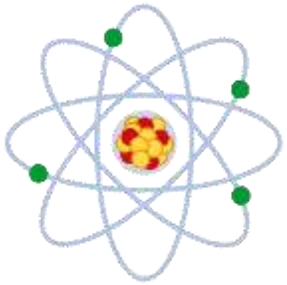
The Goal: Finding "Good" Strings

feed to simulator

Finding SMILES which achieve high "score"

O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c2ccccc2c1OC(F)F)c1cccc2ccccc12





computational chemistry tools / simulators (e.g. RDKit, *Gaussian*)

We tackle this problem using AlphaGo-like algorithms

generate molecules described in SMILES

calculate some property and use as the "score"

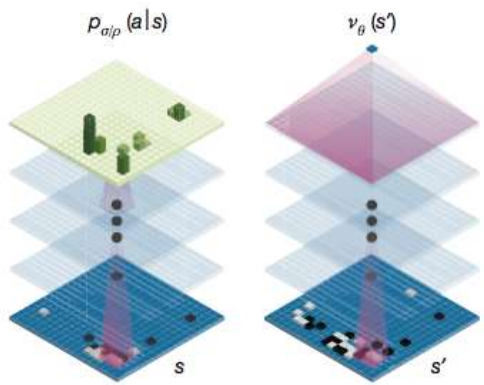
AlphaGo's two key techniques

We are using the techniques in the first version of AlphaGo, DL + MCTS

AlphaGo Zero uses RL in addition

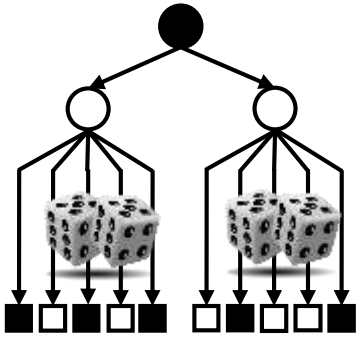
We didn't use RL, so far

Deep Learning
Recognize / Evaluate Go board
(applied to Go on 2014)



[Silver, Huang et al. 2016] Fig. 1b

MCTS
Monte-Carlo Tree Search
probabilistic tree search
(invented on 2006)



[Coulom 2006]

Reinforcement Learning
Learn from
State, Action, and Reward
(old invention, combined with DNN)

DQN

https://deepmind.com/research/dqn/

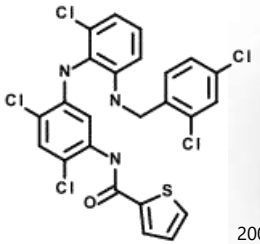
Arcade Learning Environment

https://github.com/mgbellemare/Arcade-Learning-Environment
https://www.youtube.com/watch?v=nzUiEkasXZI

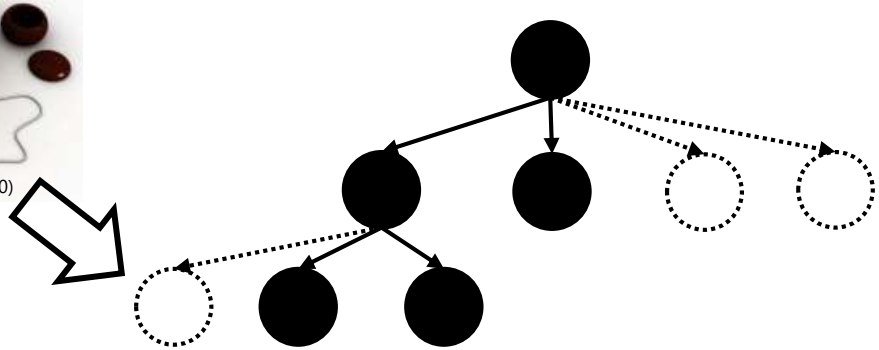
How to Search large space?

chess
10^{45}

ChemTS
10^{60}



Go
10^{170}

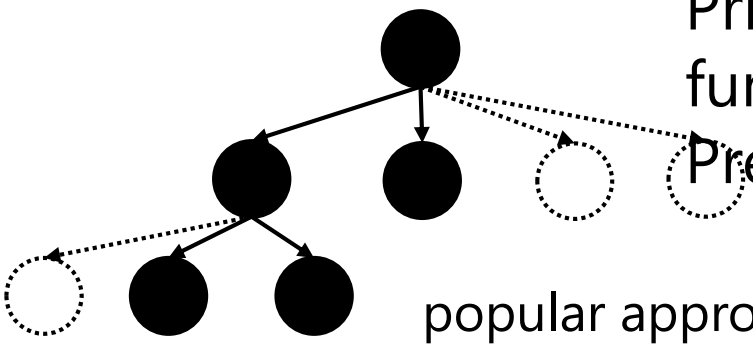


Search spaces too large for brute force exhaustive search

Pruning is necessary!
Don't search unpromising branches!

Brute force search is
- possible if 10^{20} or smaller,
- impossible if 10^{30} or greater

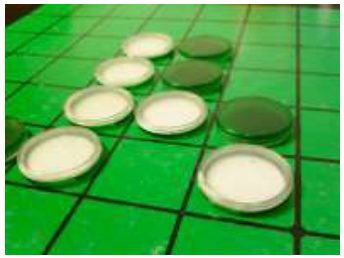
How to Prune Branches?



Prioritize nodes with some function!
Prepare **Evaluation Function**!

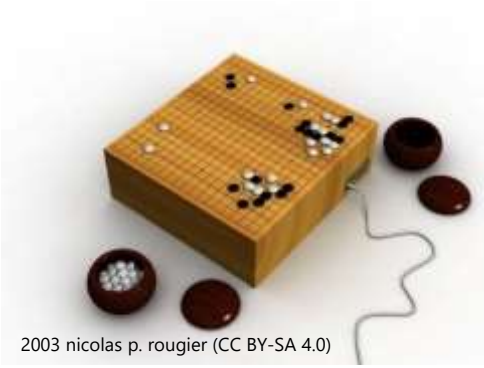
popular approach, succeeded for many domains
shortest path / puzzles / combinatorial optimization

For game AI, machine learning based (non DL)
Evaluation Function succeeded for many



What if we can’t make Evaluation Function?

This was the difficulty of Go
and the reason Google DeepMind
had focused on this game



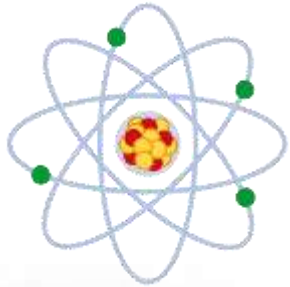
Nobody had succeeded to make accurate enough evaluation function for Go **before 2014**

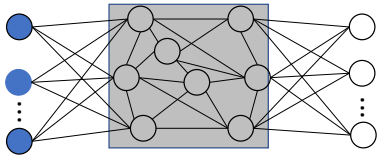
The first version of AlphaGo had used two approaches
1, Deep Neural Network based evaluation (demo)
2, Rollout based evaluation (MCTS)

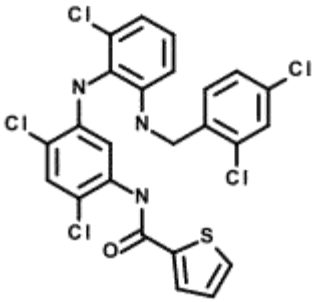
ChemTS:
An Efficient Python Library for de novo Molecular Generation

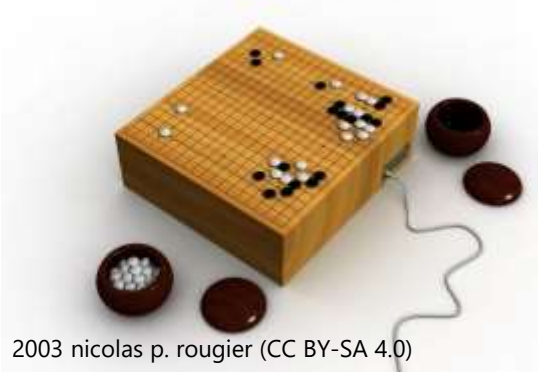
X. Yang, J. Zhang, K. Yoshizoe, K. Terayama, and K. Tsuda

It uses three components
- MCTS (UCT)
- RNN based rollout
- Computational chemistry simulator



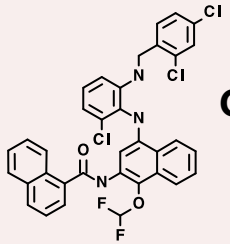






AlphaGo

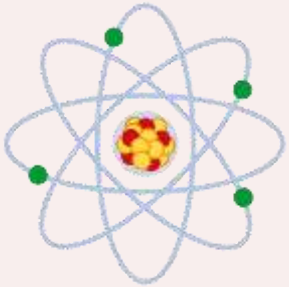
ChemTS



O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c2ccccc2c1OC(F)F)c1cccc2ccccc12

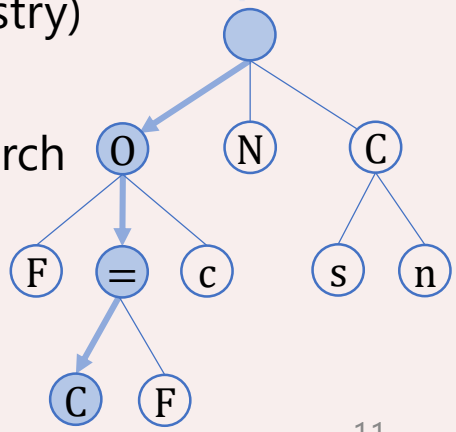
CNN (ResNet) ⟷ RNN (GRU)

ML based rollout ⟷ RNN based rollout

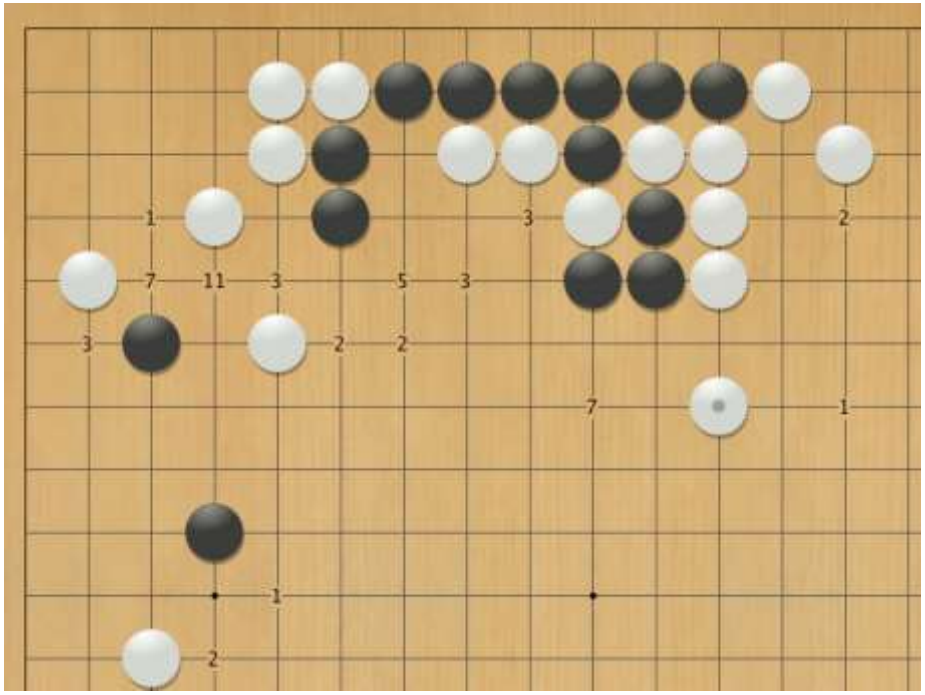


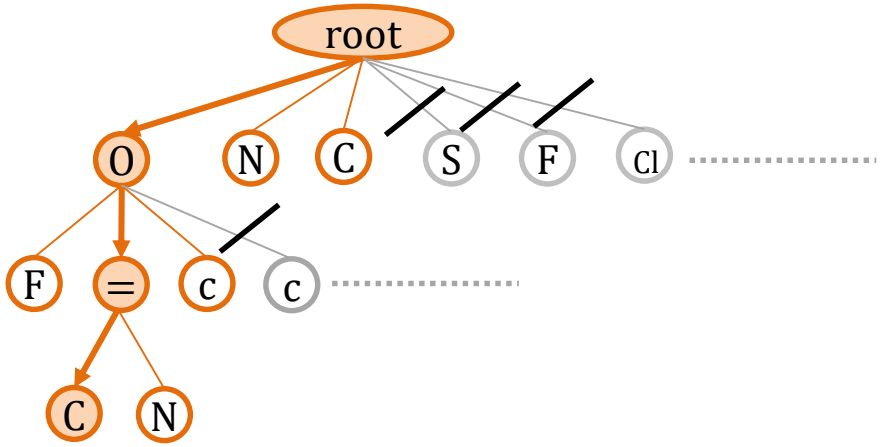
win / loss rewards ⟷ reward from simulator (computational chemistry)

Monte-Carlo Tree Search (P-UCT) ⟷ Monte-Carlo Tree Search (vanilla UCT)



Search space pruning: Go, SMILES



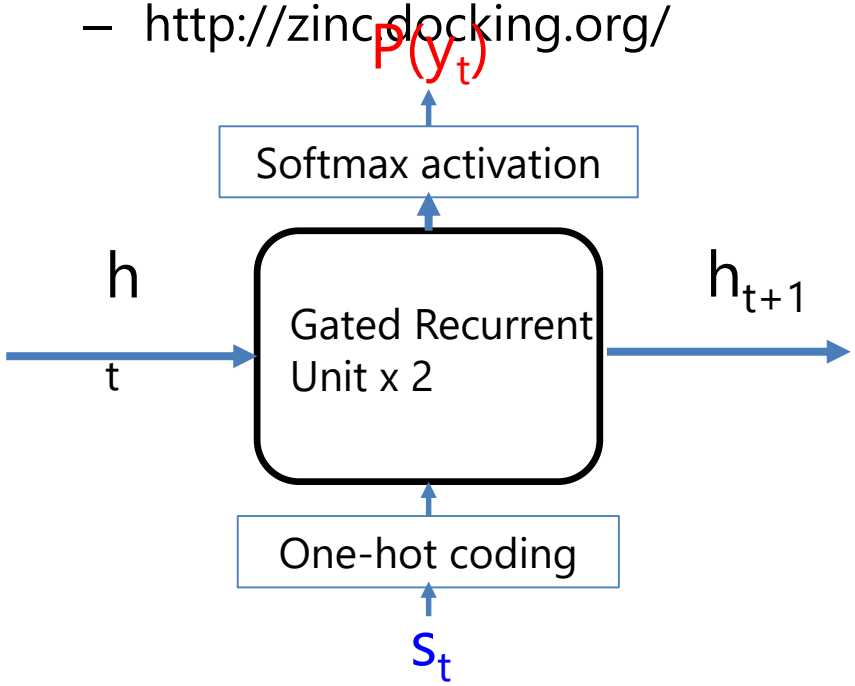


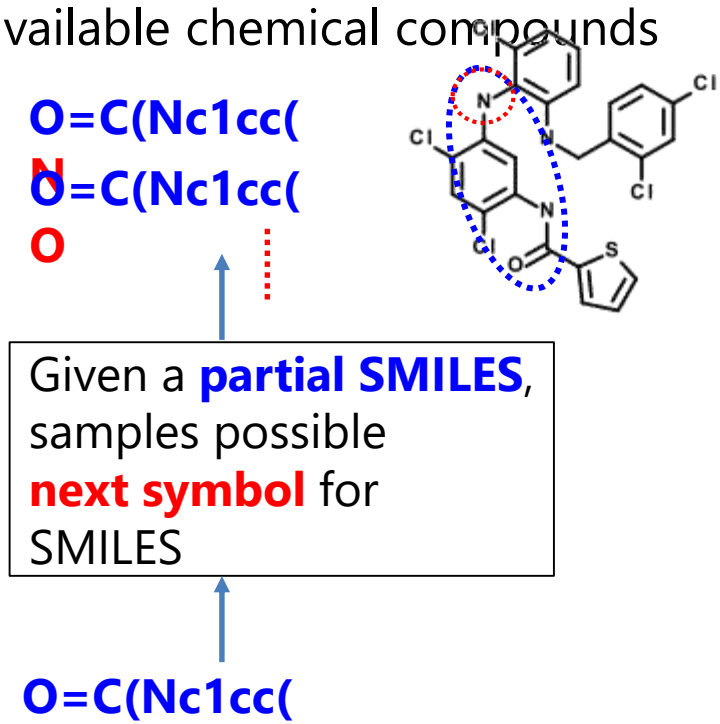
Search space can be pruned using the probability

Train RNN using Chemical DB

- Input: partial String $s_1,\ldots,s_T$
- Output: Distribution of the next symbol $P(y_1),\ldots,P(y_T)$

- Training data: one dataset in ZINC database (250,000 compounds)
  - a curated collection of commercially available chemical compounds
  - http://zinc.docking.org/



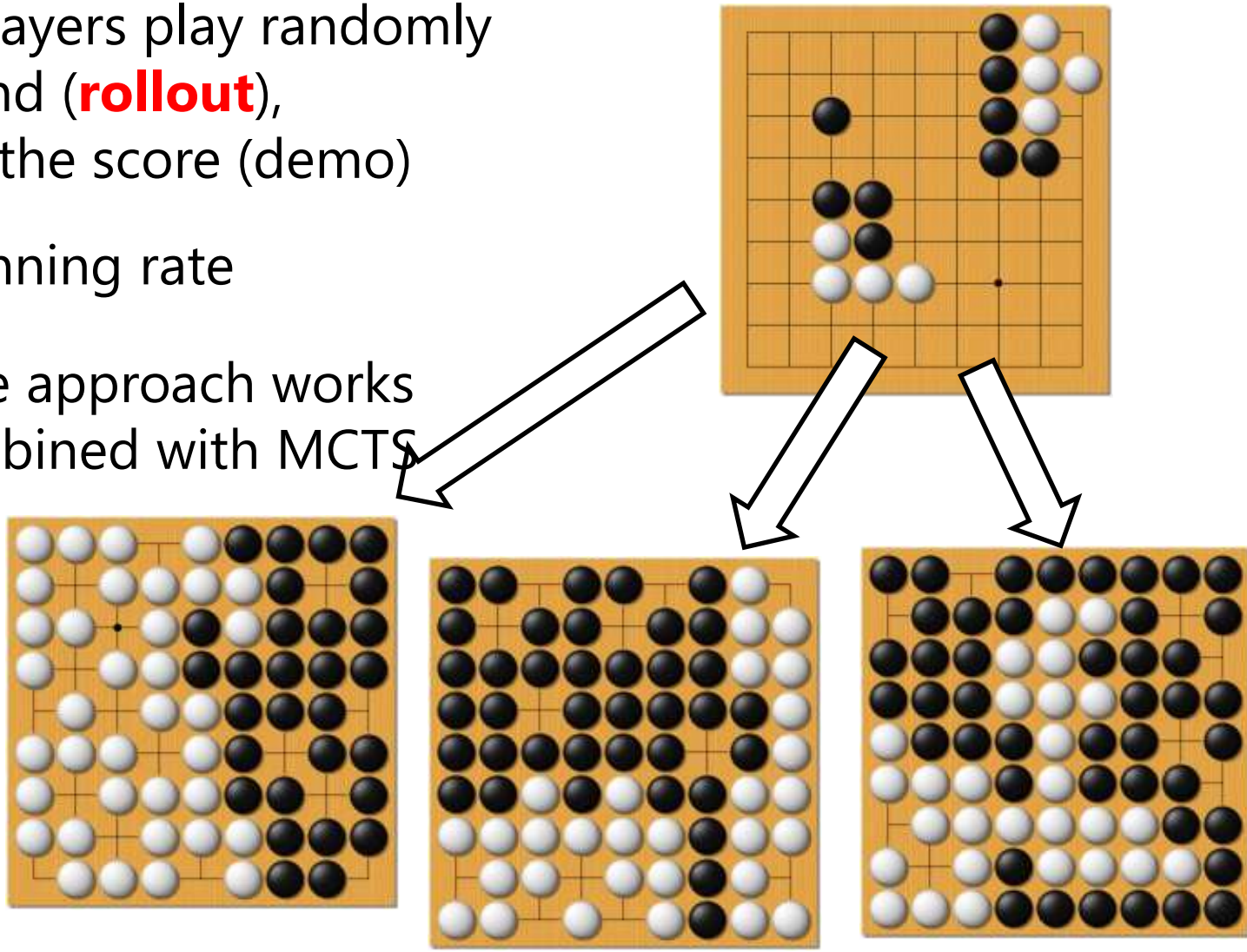


Rollout based evaluation: Go

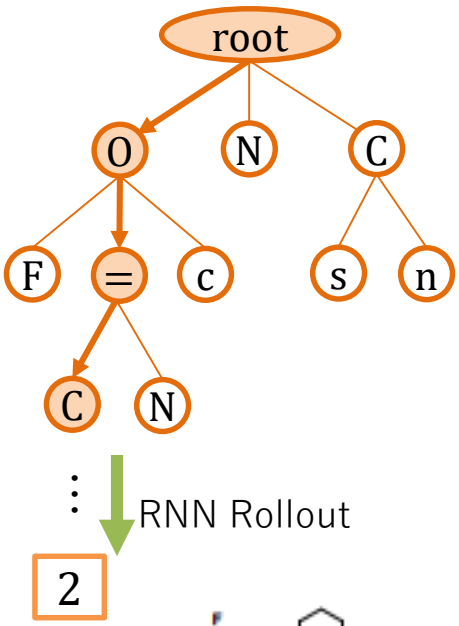
Let both players play randomly
until the end (**rollout**),
and count the score (demo)

Get the winning rate

This simple approach works
well if combined with MCTS



RNN based Rollout for Chemistry

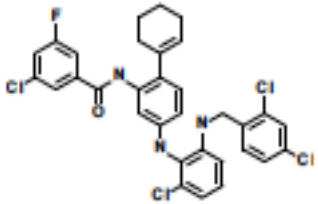


Output of our RNN was
the probability distribution of the next symbols
So, we can do Rollout.

Input: a partial SMILES (e.g. "O=C")

Output: a complete SMILES

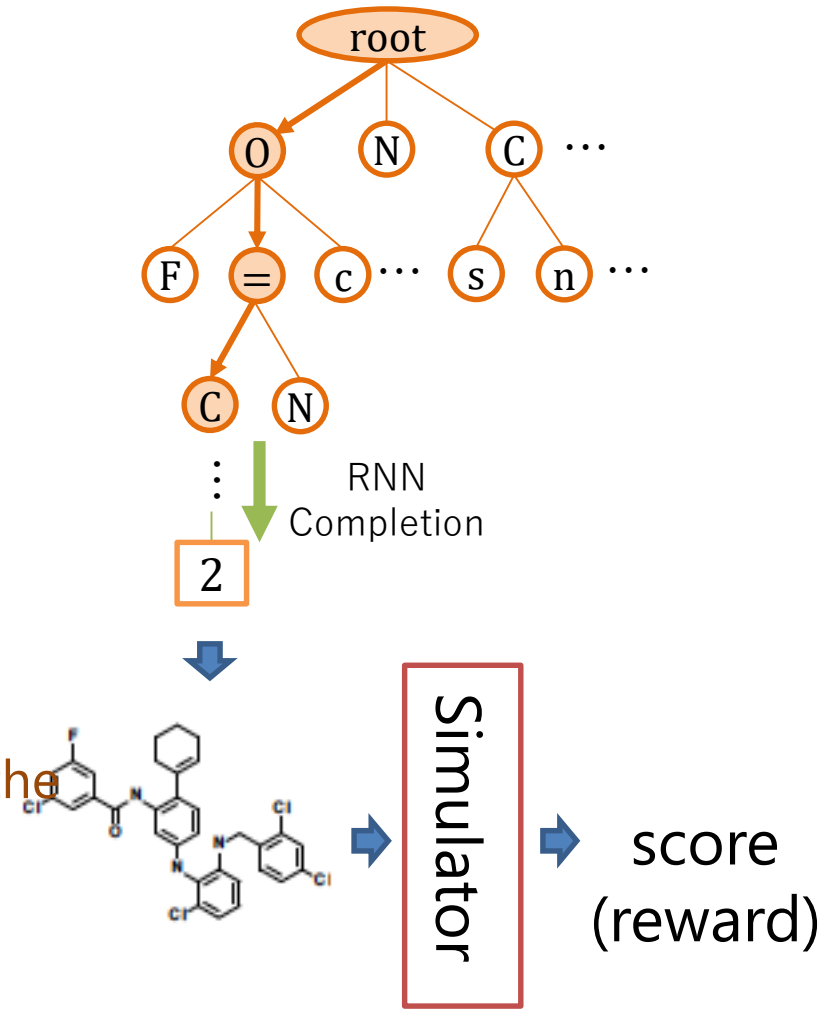
(e.g. **O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c2ccccc2c1OC(F)F)c1cccc2ccccc12)**



O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2 Cl)c2ccccc2c1OC(F)F)c1cccc2ccccc12

ChemTS combines MCTS and RNN

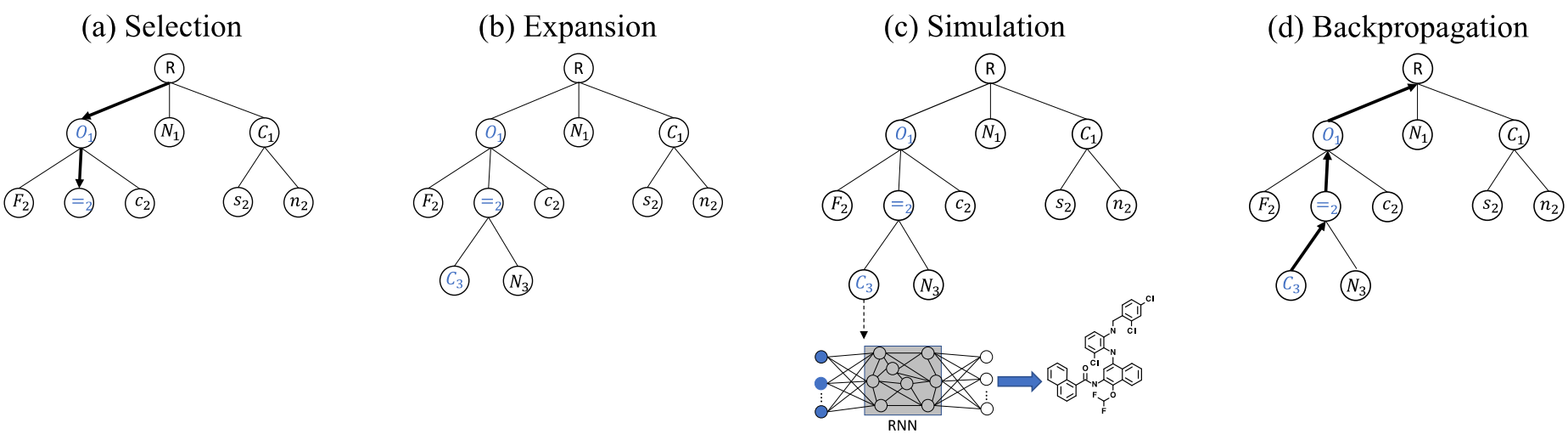
- Define search space based on SMILES
  - N^{th} letter on N^{th} level
- Use MCTS to search the space
  - we used vanilla UCT (AlphaGo used P-UCT)
- Rollout
  - Search tree defines first N letters of SMILES. RNN completes the rest of the string
  - Reward is given by a simulator
    - returns a physical property of the given molecule



Monte-Carlo Tree Search (UCT)

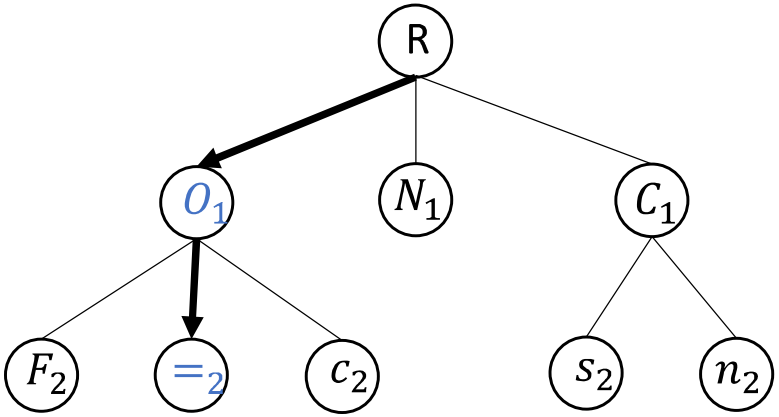
Upper **C**onfidence bound applied to **T**rees

- starts with root-node-only tree
  - depth n symbols represent n-th letter of SMILES
- search tree grows following the 4 steps shown below



1. Selection

(a) Selection



- Traverse the branch with the highest UCB1 value and select a leaf node
- UCB1 is shown on the left.
  - ➢ Random tie-breaking

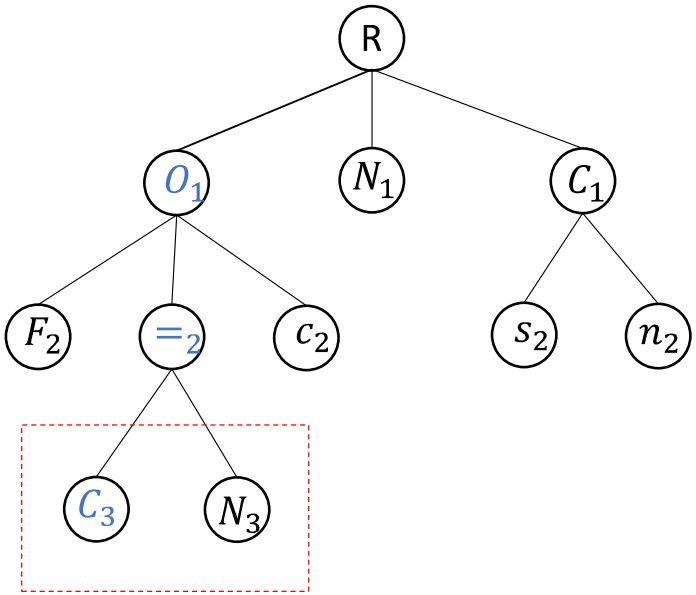
$$\frac{w_i}{s_i} + \sqrt{\frac{2 \ln t}{s_i}}$$

for branch i,
w: total reward
s: nu. visits
t: sum of s_i

AlphaGo uses a different variation of UCT (P-UCT)

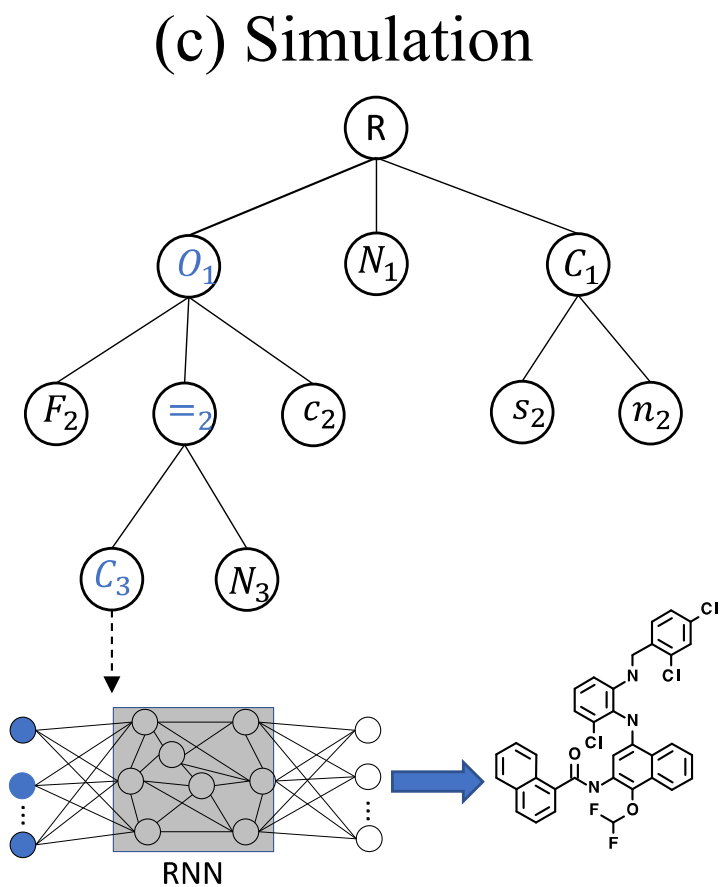
2. Expansion

(b) Expansion



- Expand the selected leaf node
  - ➢ Generate top-k children based on probability

3. Simulation

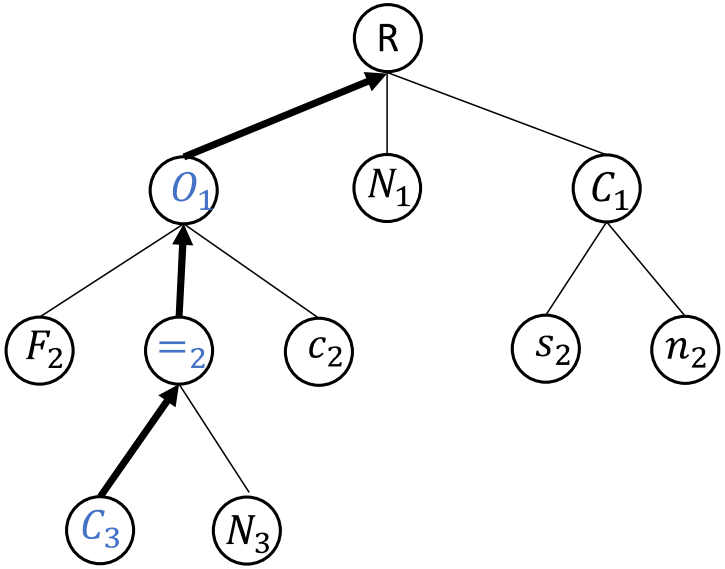


- RNN generates the SMILES string starting from the symbols in the path
  - "O=C" in this case (shown at the bottom)
  - Also converted to molecular structure
- Call external computational physics simulator and calculate reward
  - If the generated SMILES were invalid, return small reward

O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c2ccccc2c1OC(F)F)c1cccc2ccccc12

4. Backpropagation

(d) Backpropagation



- Update the values of the nodes on the path
  - nu. visits
  - total reward
- Recalculate UCB1 Value

Repeat the 4-steps until timeout

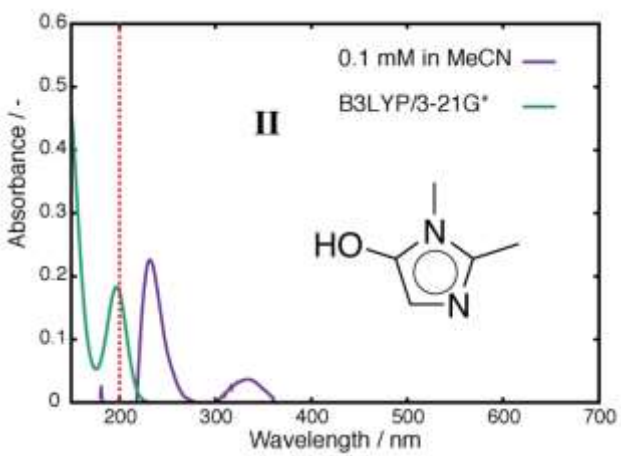
Experimental Settings: Score definition

1, "drug-likeliness" score (a benchmark problem)

$$J(m) = logP(m) - SA(m) - RingPenalty(m)$$

$\log P(S)$: octanol-water partition coefficient
$SA(S)$: synthesizability
$\mathrm{RingPenalty}(S)$: penalty for unrealistically large rings

2, UV absorption score (for peak absorbed wave length)



Gaussian simulates the spectrum

α^* : target wave length

α : simulated wave length

reward: $r = \frac{-0.01|\alpha^* - \alpha|}{1 + 0.01|\alpha^* - \alpha|}$

Gaussian is a compute chemistry software

Related Work for

De novo Molecular Generation

- Most existing methods make molecules by combining predetermined fragments

- De novo generation by deep neural networks
  - Variational autoencoder (Gómez-Bombarelli et al., Arxiv 2016, Kusner et al., ICML 2017)
  - Recurrent neural network + Feedback from external ML model (Segler et al., Nature 555, 2018)

- ChemTS (https://github.com/tsudalab/ChemTS)
  - Monte Carlo tree search + Recurrent neural

1, Drug-likeliness results (speed)

Comparison with existing methods (our method in **bold**)

Methods	2 h	4 h	6 h	8 h	Mol./min.
ChemTS[3]	**4.9 ± 0.4**	**5.4±0.5**	**5.5±0.4**	**5.6±0.5**	**41±1.6**
RNN+BO	3.5 ± 0.3	4.5±0.2	4.5±0.2	4.5±0.2	8.3±0.0
Only RNN	4.5 ± 0.3	4.6±0.3	4.8±0.3	4.8±0.3	41±1.4
CVAE+BO[2]	−30±27	−1.4±2.2	−0.6±1.1	−0.0±0.9	0.1±0.1
GVAE+BO[1]	−4.3±3.1	−1.3±1.7	−0.2±1.0	0.3±1.3	1.4±0.9

avg. and s.d. of the score

molecules generated per minute

References

[1] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. "Grammar variational autoencoder". ICML2017.

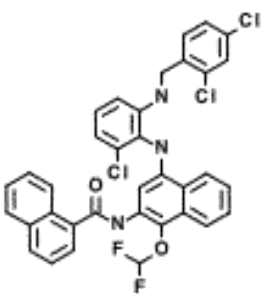
[2] R. Gómez-Bombarelli, D. Duvenaud, J. Miguel Hernández-Lobato, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. "Automatic chemical design using a data-drive continuous representation of molecules". arXiv:1610.02415, 2016.

[3] X. Yang, J. Zhang, K. Yoshizoe, K. Terayama, K. Tsuda. "ChemTS: an efficient python library for de novo molecular generation". Science and Technology of Advanced Materials (STAM), 2017 Dec 31;18(1):972-6.

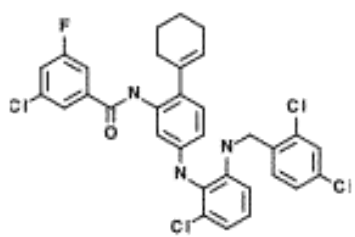
1, drug-likeliness results (molecules)

Top five discovered by ChemTS. (**bold red** parts were found in tree)

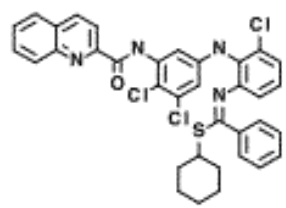
SMILES representation	*J(S)*
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c2ccccc2c1OC(F)F)c1cccc2ccccc12	6.56
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)ccc1C1=CCCCC1)c1cc(F)cc(Cl)c1	6.43
O=C(Nc1cc(Nc2c(Cl)cccc2N=C(SC2CCCCC2)c2ccccc2)cc(Cl)c1Cl)c1ccc2ccccc2n1	6.34
O=C(Nc1cc(Oc2ccc(Cl)cc2Cl)ccc1Nc1cc(Cl)ccc1Cl)c1ccc(Cl)cc1	6.33
O=C(Nc1cc(Nc2c(Cl)cccc2Cl)c(Cl)cc1Br)N(c1ccccc1)c1ccc(Cl)cc1	6.26



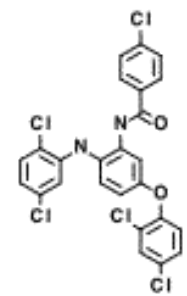
$J = 6.56$



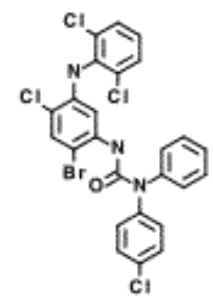
$J = 6.43$



$J = 6.34$



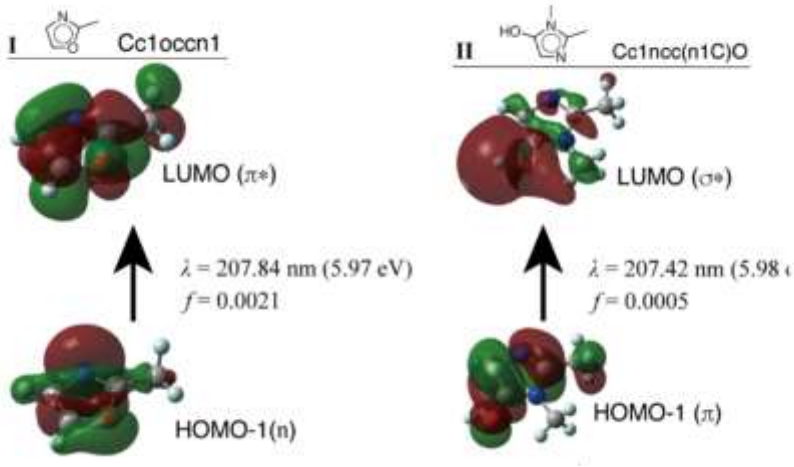
$J = 6.33$



$J = 6.26$

2, UV absorption results

- Calculated UV absorption
  - using *Gaussian*
    (compute chemistry tool)
- finding molecules with the targeted peak absorbing wave length
  - for 200, 300, 400, 500, 600nm
  - higher score if peak is closer to target
  - actually measured some of these (non-toxic, stable ones)
    two of them shown on the right
- longer simulation time
  - runs DFT calculation
    (DFT: Density Functional Theory)
    (details omitted)



discovered molecules for 200nm target

Reference

M. Sumita, X. Yang, S. Ishihara, R.Tamura, and K. Tsuda. “Hunting for Organic Molecules with Artificial Intelligence: Molecules Optimized for Desired Excitation Energies”, ACS Cent Sci. 2018 Sep 26;4(9):1126-1133.

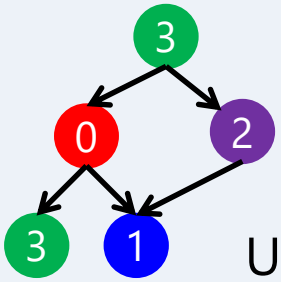
Conclusions

- MCTS + RNN + Simulator was effective for chemistry
  - Still Proof-of-Concept level, but promising
  - https://github.com/tsudalab/ChemTS

Main Takeaway

- Search + DL + HPC can be a strong tool
- Please consider combining ML with Search when solving complex problems

Utilizing Massive Parallel MCTS for finding more molecules



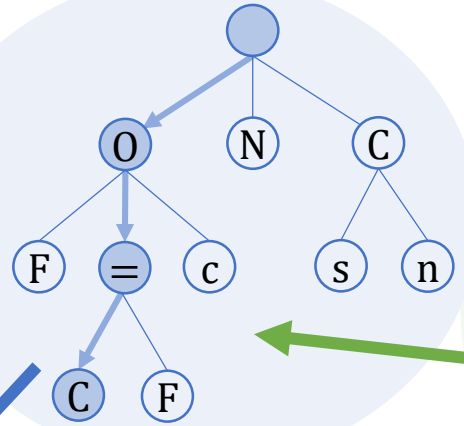
Parallel UCT Scales up to 1,000+ CPU cores

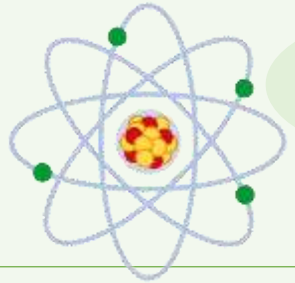
Using part of the techniques in the following paper

K. Yoshizoe et al., "Scalable Distributed Monte-Carlo Tree Search",
Symposium on Combinatorial Search (SoCS), 2011.

Search algorithms

Monte-Carlo Tree Search

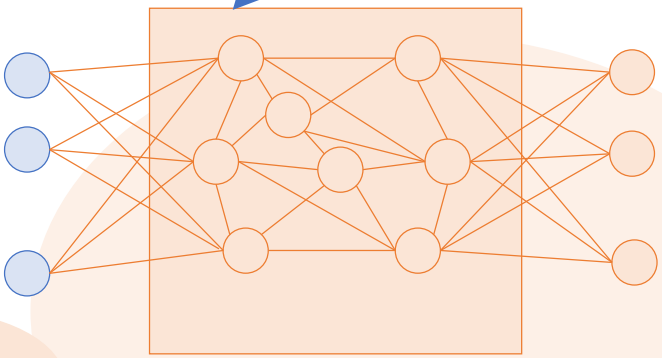




HPC

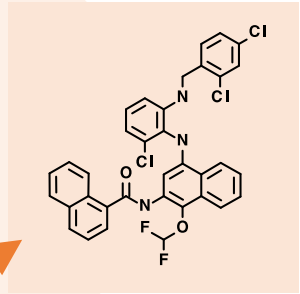
High Performance Computing

feed back **score** calculated by physical simulation



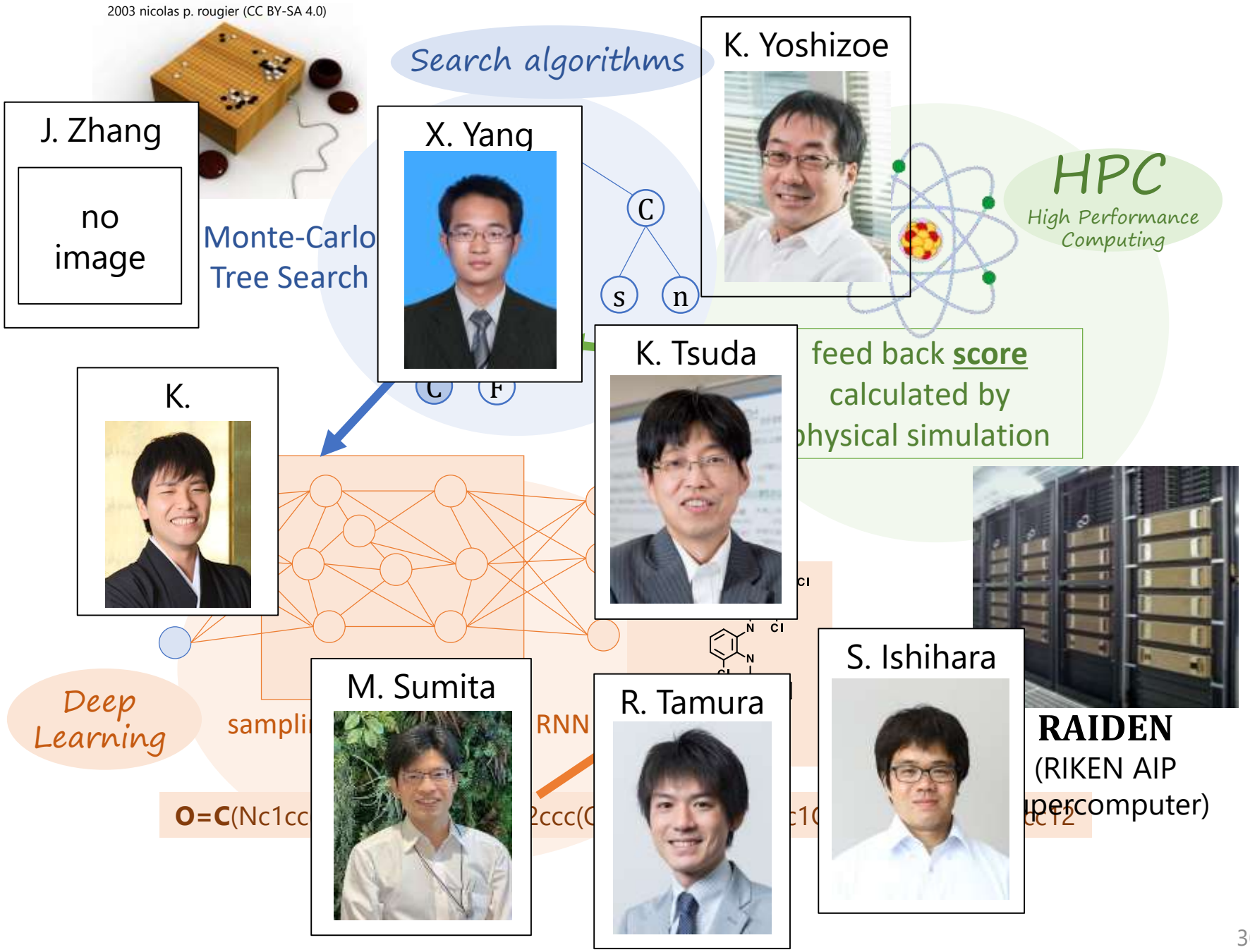
Deep Learning

sampling molecules with RNN



RAIDEN
(RIKEN AIP supercomputer)

O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c2ccccc2c1OC(F)F)c1cccc2ccccc12


2003 nicolas p. rougier (CC BY-SA 4.0)
Search algorithms
J. Zhang
no image
Monte-Carlo Tree Search
X. Yang
K. Yoshizoe
HPC
High Performance Computing
feed back score calculated by physical simulation
K.
K. Tsuda
Deep Learning
sampli
RNN
M. Sumita
R. Tamura
S. Ishihara
RAIDEN
(RIKEN AIP supercomputer)
O=C(Nc1cc
2ccc(C
c1C
dc12

So, who am I?

Parallel computing lab at graduate school	Search Algorithms
Digital wireless communication (at **FUJITSU**)	Game AI algorithms
Biometric security (finger vein recognition)	Parallel Search Algorithms
Bioinformatics (a bit)	Multiple Testing (statistics) (a bit)

Computer Go book (in Japanese)

I am now working for RIKEN AIP
(Center for Advanced Intelligence Project)

Wanted!
People with HPC background
and interested in AI

Our supercomputer RAIDEN ranked 4th in Green500. (Jun 2017)
Has 432 Tesla V100 GPUs
(I am in charge of the selection, procurement and maintenance)

雷電 **RAIDEN**