Maximizing Network Reward Based on A General Framework of Monte Carlo Tree Search

Liu Weizhi

Department of Industrial & Systems Engineering
National University of Singapore

weizhiliu2009@gmail.com

November 6, 2014

Overview

1 Introduction
- Task
- Challenge & Solution

2 Methods
- Notations
- Network Construction and Reward Evaluation
- General Framework of MCTS

3 Results & Discussion
- Datasets
- Algorithms Comparision
- Best Reward

4 Conclusion

- Game Rule
  - Input: undirected network adjacency list and original color sequence
  - Initial: choose a node whatever you like and color it
  - Process: pick a node from those uncolored nodes which connect to colored nodes and color it
  - Terminate: the original color sequence runs out or no further candidate nodes to color
  - Reward: number of edges which connect to two different color nodes
  - Output: population sequence and network reward
- Goal: search the optimal network population sequence in order to maximize the total reward of the network

- Challenge
  - Naive brute force algorithm could consume enormous computation budget when the network is large enough.
  - Monte Carlo Tree Search algorithms might be a possible and efficient way.
  - However, there exists various MCTS algorithms. So which algorithm might be useful for this specified project?
- Solution
  - ***Monte carlo search algorithm discovery for one player games (Francis Maes et. al 2012)***
  - Generate abundant potential MCTS algorithms by combining basic search components.
  - Select the most appropriate algorithm generated for this specified project and then search the optimal population sequence.

Table 1: Notations

notation	definition
$\mathcal{A}_{n\times n}$	adjacent matrix of the network with n nodes
a_{ij}	element of $\mathcal{A}_{n\times n}$ which represents the number of edges between node i and j
$\vec{s}$	the original input sequence
$\vec{c}_{1\times n}$	color vector for every nodes, c_i represents the color of node i
r_{ij}	reward value between node i and node j
$R(\mathcal{A}_{n\times n}, \vec{c}_{1\times n})$	total reward of the network
R^*	current best network reward
w_k	the candidate nodes set at step k
$\vec{p}_k = (p_1, p_2, \cdots, p_k)$	population sequence at step k of which element p_i represents the ith populated nodes
$\vec{p}^*$	current best population sequence
$\tau(k)$	represents if the game enters into end at time k
B	total budget for each algorithm
$numCalls$	current times of evaluation
$\mathcal{S}$	search component
$N^{(repeat)}$	repeat times for repeat component
$N^{(select)}$	multiple factor for select component
η	weight factor for exploring of ucb value
$\mathcal{L}_i$	the lower level search component standalone parameters recursive list at level i

- Network Construction: The network $\mathcal{A}_{n\times n}$ could be constructed based on the network adjacent list by continuously update a_{ij} (number of edges between node i and j).
- Reward Evaluation
  - Redefine Color Sequence $\vec{c}_{1\times n}$:

$$c_i = \begin{cases} -1 & \text{if node } i\text{'s color is 0} \\ 0 & \text{if node } i \text{ is not colored} \\ 1 & \text{if node } i\text{'s color is 1} \end{cases} \tag{1}$$

  - Reward Calculation based on Matrix

$$r_{ij} = \frac{(c_i c_j - 1)}{2} c_i c_j a_{ij} = \frac{1}{2}(c_i^2 a_{ij} c_j^2 - c_j a_{ij} c_j) \tag{2}$$

$$R(\mathcal{A}, \vec{c}) = \frac{1}{2}\sum_i \sum_j r_{ij} = \frac{1}{4}[(\vec{c})^2 \mathcal{A}(\vec{c}^T)^2 - \vec{c}\mathcal{A}\vec{c}^T] \tag{3}$$

Helper Components

Table 2: Illustration of helper components

helper component	task	input	output
candidate	update candidate nodes set for populating by set operation rather than loop	$\vec{p}_{k-1}, p_k, w_{k-1}$	w_k
terminal	check wheter the game enters into end	$\vec{p}_k, w_k$	$\tau(k)$
reward	calculate the network reward	$\mathcal{A}_{n\times n}, \vec{c}_{1\times n}$	R
evaluate	update the budget consumption, best reward and population sequence	$\vec{p}_k, \vec{p}^*, R^*, \mathcal{A}_{n\times n}, \vec{c}_{1\times n}$ $numCalls, B$	$\vec{p}^*, R^*,$ $numCalls$
invoke	invoke other search components	$\vec{p}_k, w_k, \mathcal{S}$	$\vec{p}_m(m > k)$

Search Components

- Two types of search components: atom component and free component
- free component can invoke other search components
- atom component can only be invoked by other search components

Table 3: Illustration of search components

search component	task	type
simulate	uniformly randomly select a full population sequence	atom component
step	generate a full population sequence step by step	free component
repeat	return the best population by repeating N^{repeat} times evaluation	free component
lookahead	return the best population by evaluating full population sequence among all next candidates	free component
select	a mini version of UCB	free component

Search Components

There are basically two major differences for select component from Maes.

- Budget automatic adjustment

$$Budget(k)^{select} = Size(w_k)[(1 - \frac{Dim(\mathcal{A})N^{select}}{Size(w_k)})\frac{Size(\vec{p}_k)}{Size(\vec{s})} + \frac{Dim(\mathcal{A})N^{select}}{Size(w_k)}] \tag{4}$$

- Normalization for UCB value
    - The reward part and explore part is extremely different in terms of scale.
    - Divide reward part by the current best reward.
    - Transform explore part into $(0, 1)$ via logistic function.
    - However, the search space is too large that explore too much may not be a good choice.

Algorithms Generator

Table 4: MCTS algorithms examples

algorithm	recursive expression
rmc(N_1^{select}, N_2^{select})	step(repeat(N_1^{select}, step(repeat(N_2^{select}, simulate()))))
nmc(1)	step(lookahead(simulate()))
nmc(2)	step(lookahead(step(lookahead(simulate()))))
uct(N^{repeat}, N^{select}, η)	step(repeat(N^{repeat}, select(N^{select}, η, simulate())))

A possible simulation path should be set 1, set 2, set 3, set 6, set 4, set 5 and set 7 considering the difficulity of different datasets.

Table 5: Descriptions of datasets

dataset	network size (range)	sequence size
set 1	10	10
set 2	153	20
set 3	153	130
set 4	961	400
set 5	5002	4000
set 6	483	400
set 7	11748	9000

Optimality and Efficiency

In order to find the most suitable MCTS algorithm for this project,10000 budget are allocated to each algorithm and 10 independent simulation runs are conducted.

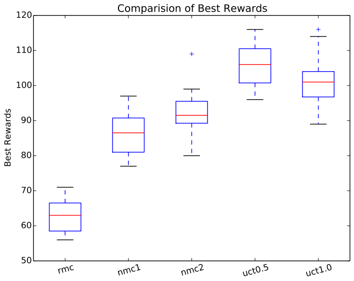

Figure 1: Best Rewards

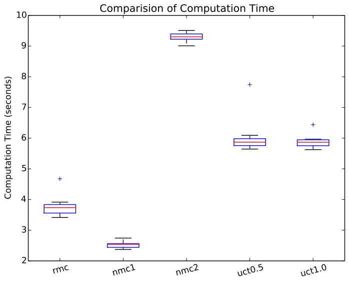

Figure 2: Computation Time

Apparently, uct is most suitable for this project comparing with any other popular MCTS algorithms

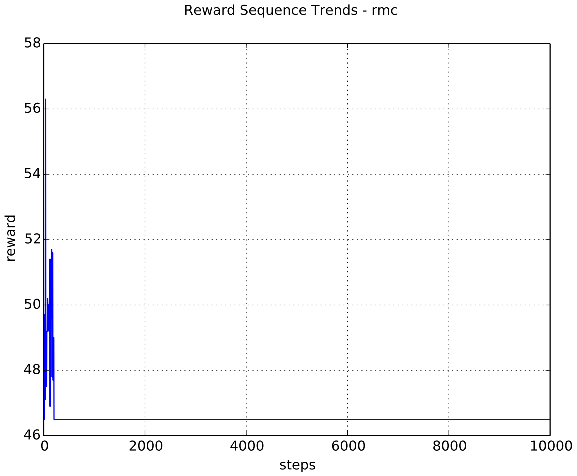

Figure 3: Trends of Reward Sequence for rmc

Algorithms Comparision

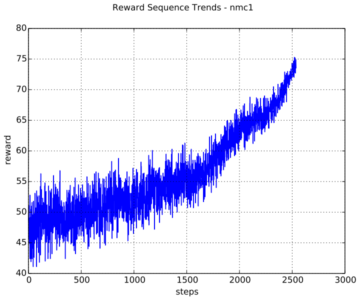

Figure 4: Trends of Reward Sequence for nmc1

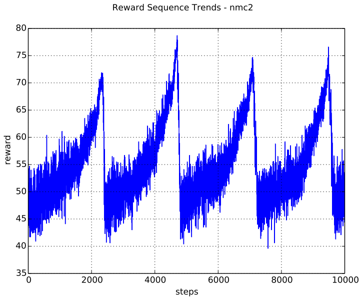

Figure 5: Trends of Reward Sequence for nmc2

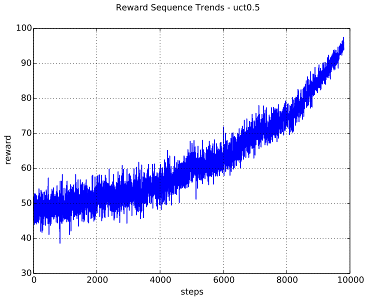

Figure 6: Trends of Reward Sequence for uct0.5

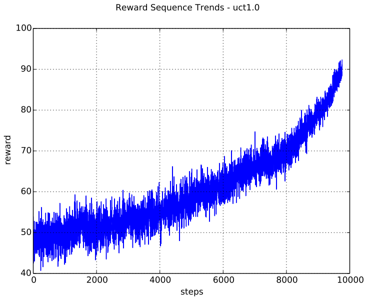

Figure 7: Trends of Reward Sequence for uct1.0

Best Reward

Table 6: Best Reward

dataset	current best reward
set 1	19
set 2	157
set 3	2612
set 4	248
set 5	1170
set 6	587
set 7	9979

- Summary
  - uct0.5 and uct1.0 perform best in both criteria of optimality and efficiency among the five candidate algorithms
  - uct seems like to learn the tree structure step by step
  - nmc1 is the most fast algorithm which can also guarantee a better soultion than rmc
- Limitations
  - More statistics should be stored for the deeper nodes.
  - Randomly select a node to reduce the branching factor in the first layer.
  - More deep understanding of the network and color sequence.

Thanks!
Q&A