GUIDING MONTE CARLO TREE SEARCH BY SCRIPTS IN REAL-TIME STRATEGY GAMES

Submitted to: Dr. Nevin Darwish

Presented by : Yahia Ali in 4/11/2019

Author: Zuozhi Yang, Santiago Ontanón

Drexel University

Published: 2019-10-08 in AAAI

Introduction

- Monte Carlo Tree Search (MCTS) is a well known game tree search.
- The large branching factor problem is even more pronounced in real-time strategy (RTS) games.
- <u>in this paper, we study how to integrate human knowledge into MCTS</u> in order to improve the gameplay strength of MCTS under tight computational budget constraints.

Background

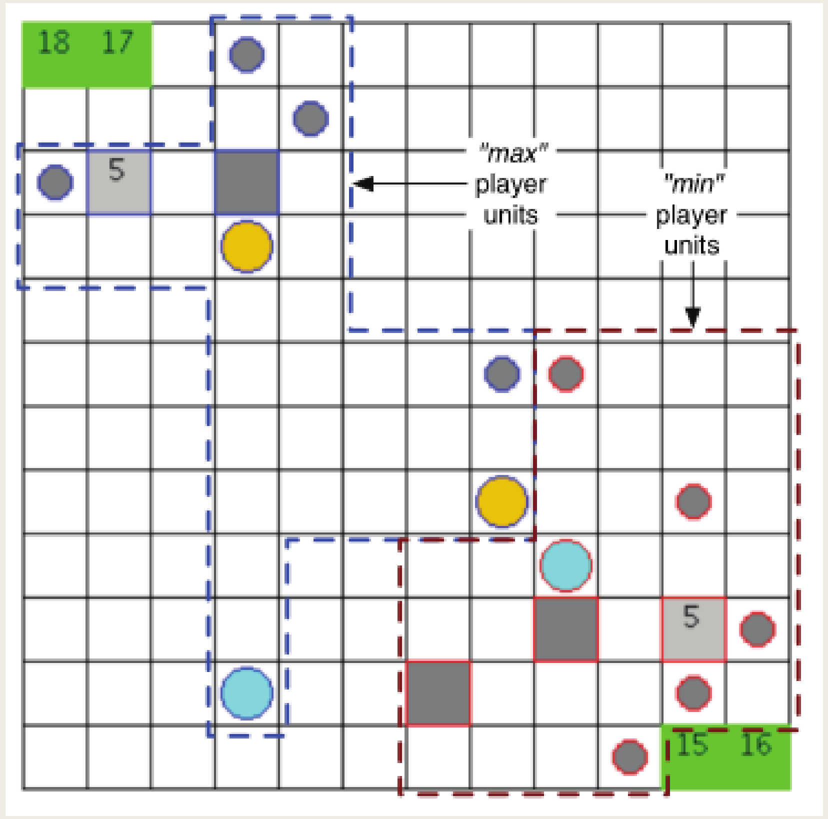
- In RTS games, players control a large number of units to mine resources, build more units, and combat with other players.
- RTS games have been receiving an increased amount of attention as they are even more challenging than games like Go or Chess in at least three different ways.
- **in this paper, we chose μRTS as our experimental domain**, as it offers a forward model for game tree search approaches such as minimax or MCTS.

Monte Carlo Tree Search in RTS Games

- Monte Carlo Tree Search is a method for sequential decision making for domains that can be represented by search trees.
- most of the successful variants of MCTS do not scale up well to RTS games due to the combinatorial growth of branching factor with respect to the number of units.

µRTS

- µRTS is a simple RTS game maintaining the essential features that make RTS games challenging from an AI point of view: simultaneous and durative actions, combinatorial branching factors and real-time decision making.
- There is one type of environment unit (minerals) and six types of units controlled by players.
- the environment can have walls to block the movement of units.



Naïve Monte Carlo Tree Search

- Naïve MCTS is a variant of MCTS specifically designed to handle RTS games.
- **All the experiments reported in this paper use Naïve MCTS as the baseline.**
- MCTS algorithms employ two policies called the tree policy and the default or playout policy.
- The former is used to determine which node of the tree to explore next, and the later is used to perform stochastic rollouts from the leaf of the tree until a terminal state is reached.
- the key feature of Naïve MCTS is it uses a sampling policy based on Combinatorial Multi armed Bandits (CMABs) called Naïve Sampling in order to scale up to the combinatorial branching factors of RTS games.

Guided Naïve Monte Carlo Tree Search

- it is usually unfeasible to produce a reliable estimation of the expected reward of the possible actions from sampling.
- scripted bots are hard-coded agents using human knowledge, which follow simple and fast strategies.
- In this paper, we propose two tree policies to in-corporate scripted bots as guidance for game tree search.
- In this way, we preserve the original search space of MCTS unchanged, and the resulting MCTS algorithms will still converge to the optimal action in the limit.

First-Choice Guided Naïve Sampling (FC-GNS)

- This tree policy utilizes a single script to inform the search.
- The first time the tree policy is used on a search node of the MCTS tree to determine which is the first child that will be added to such node. In all other circumstances, the tree policy selects a player action that consists of unit actions selected by Naïve Sampling.
- by first selecting an action based on a script, we ensure that if a node is only visited once, at least the action selected is an action that makes sense.
- since the script is only used the first time a node is visited, in the long-run, MCTS can ignore it if other actions turn out to have higher reward.

Mixed Scripts Guided Naïve Sampling (k-GNS)

- This tree policy utilizes a collection of k scripts ($k \geq 1$) to guide the search. Scripts are used in two different ways:
  - *In the very first iteration in each node of the search tree: similarly to FC-GNS, the first time the tree policy is used, the scripts are used to chose the selected action. Rather than using a single script, k-GNS chooses a script uniformly from a pool of scripts.*
  - *In each subsequent iteration: when using a local MABs in Naïve Sampling, with probability ε, the scripts will be used to select the unit action, and with probability 1 − ε, the local MABs will be used.*
- Larger ε values force the search towards the scripts, which can be useful in larger maps.
- <u>in our experiments, we will use ε =0.33 for all map except 128x128 we will use ε =0.75 . and we will use the name 1-GNS to refer to the degenerate form of k-GNS when k = 1.</u>

Experiments

- In order to evaluate our approaches, we performed experiments in μRTS, and used maps of five different sizes: 8×8, 12 × 12, 16 × 16, 32 × 32, and 128 × 128.
- For each map , both players always start with one base and one worker.
- Since FC-GNS and k-GNS are basically the result of integrating Naïve MCTS with scripts, we compare the performance of our algorithms against both Naïve MCTS and the scripts used for guidance.
- Scripted Bots used for Guidance We used four simple scripted bots provided by μRTS as guidance bots

Experiments(cont.)

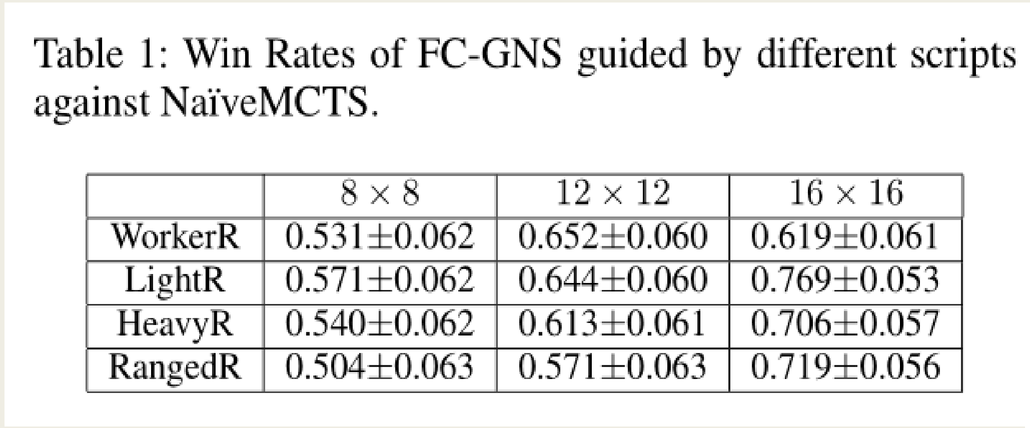


Table 1: Win Rates of FC-GNS guided by different scripts against NaïveMCTS.

	8×8	12×12	16×16
WorkerR	0.531±0.062	0.652±0.060	0.619±0.061
LightR	0.571±0.062	0.644±0.060	0.769±0.053
HeavyR	0.540±0.062	0.613±0.061	0.706±0.057
RangedR	0.504±0.063	0.571±0.063	0.719±0.056

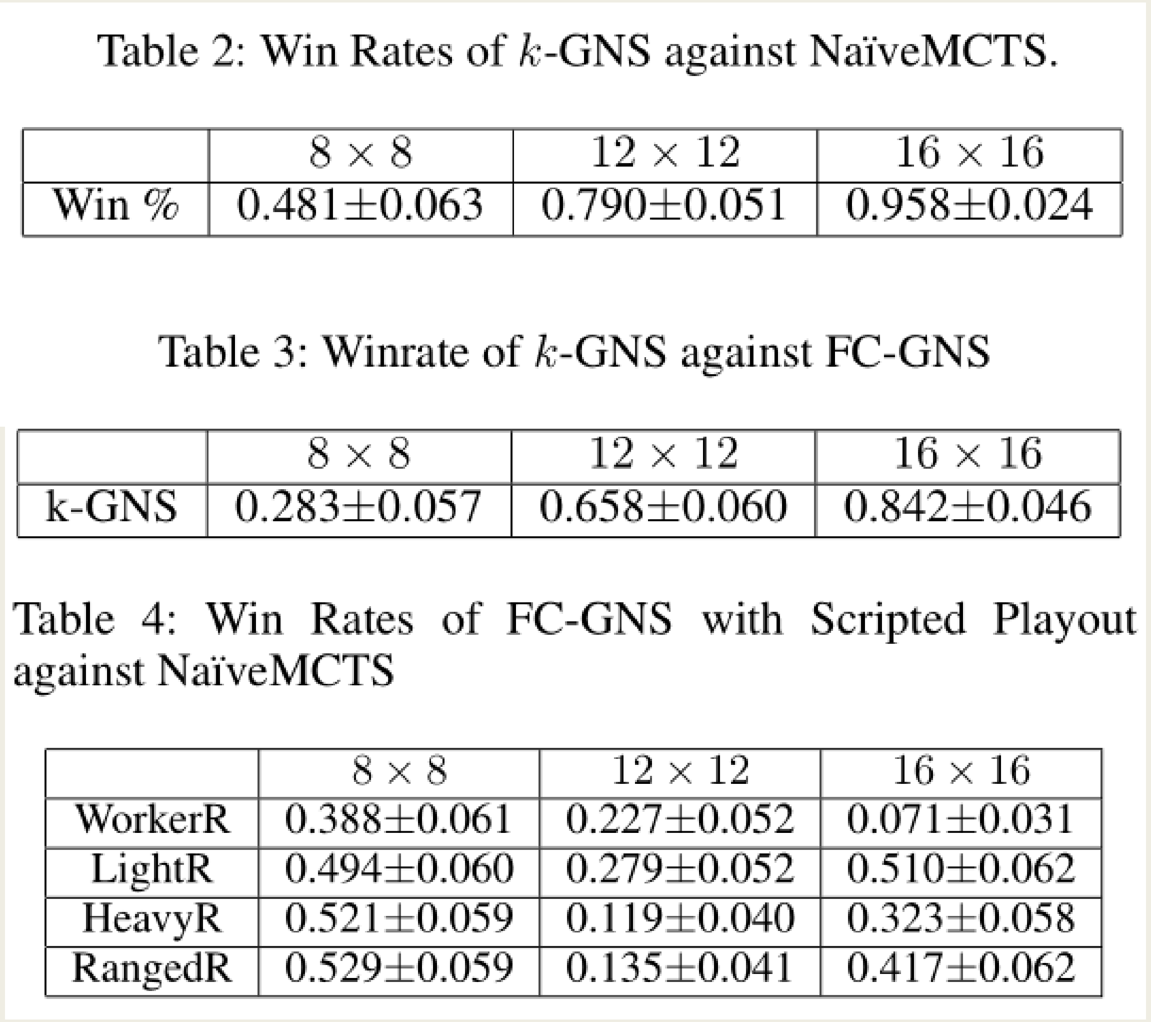


Table 2: Win Rates of k-GNS against NaïveMCTS.

	8×8	12×12	16×16
Win %	0.481±0.063	0.790±0.051	0.958±0.024

Table 3: Winrate of k-GNS against FC-GNS

	8×8	12×12	16×16
k-GNS	0.283±0.057	0.658±0.060	0.842±0.046

Table 4: Win Rates of FC-GNS with Scripted Playout against NaïveMCTS

	8×8	12×12	16×16
WorkerR	0.388±0.061	0.227±0.052	0.071±0.031
LightR	0.494±0.060	0.279±0.052	0.510±0.062
HeavyR	0.521±0.059	0.119±0.040	0.323±0.058
RangedR	0.529±0.059	0.135±0.041	0.417±0.062

Experiments(cont.)

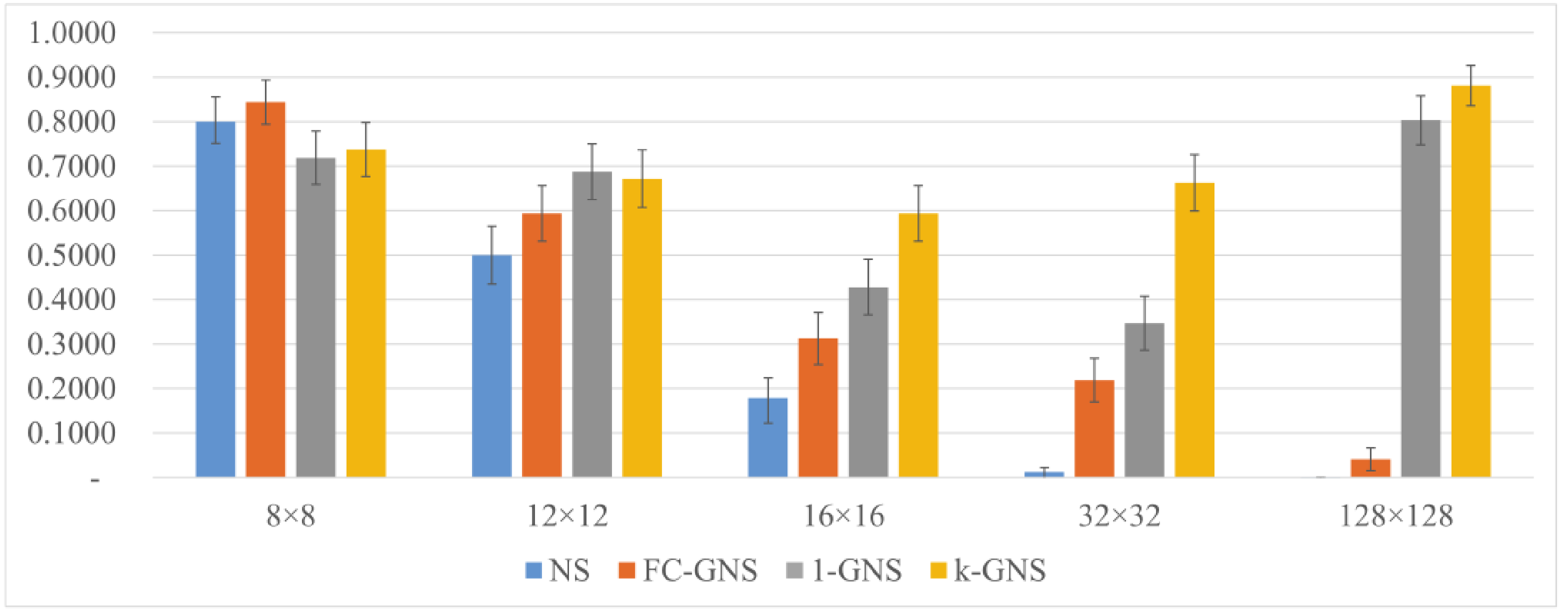
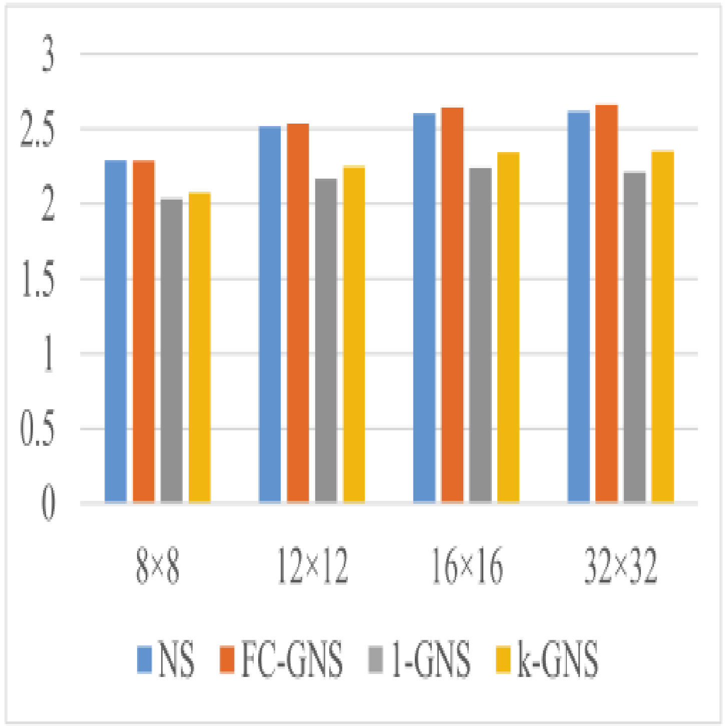
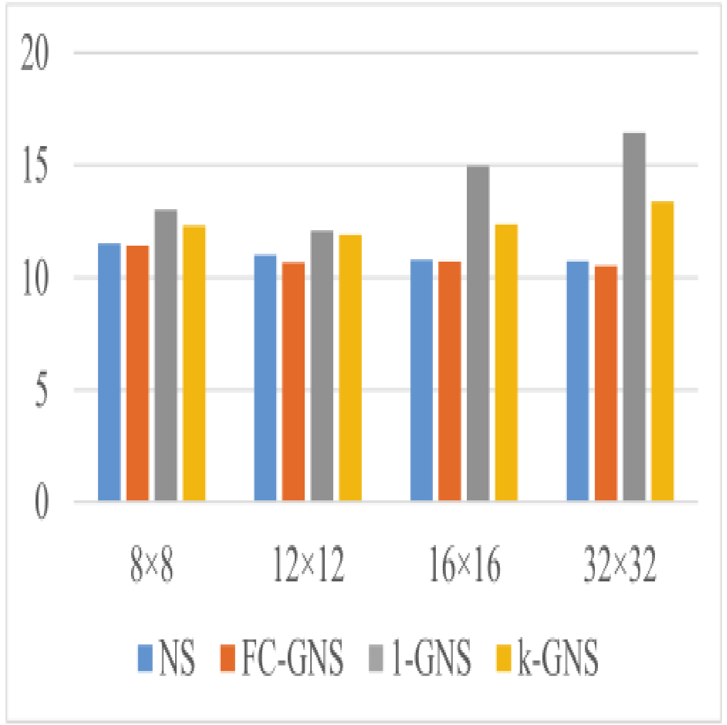


Figure 2: Win rates of NaïveMCTS, FC-GNS and k-GNS against the four rush scripts

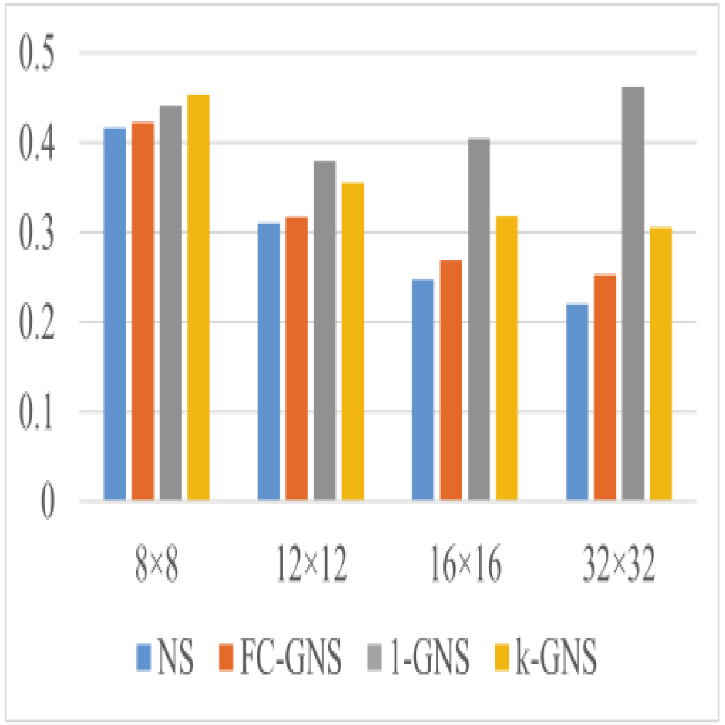
Experiments(cont.)



(a) Average Number of Children per Node



(b) Max Tree Depth



(c) Proportion the best action was visited at the top level

Comparison of the Shape of the Search Tree

- we collected statistics of the shape of the search trees. The statistics are collected from 48 games between FC-GNS and k-GNS and vanilla Naïve MCTS (24 games for FC-GNS and 24 for k-GNS). We use data from the first 1000 game cycles of each game.
- In Figure 3, we compare the shape of the trees generated by FC-GNS and k-GNS to the shape of vanilla Naïve MCTS from three perspectives:
  - a) *average number of children per node*
  - b) *maximum tree depth*
  - c) *the proportion of times that MCTS visited the selected action at the root of the tree.*
- In summary, FC-GNS generates trees that are very similar to vanilla Naïve MCTS, but k-GNS generates narrower and deeper trees. Moreover, the guidance has a stronger influence in larger maps, where we can see that the percentage of times the final action is one of the actions proposed by the script is higher than the percentage of times vanilla Naïve MCTS would have selected one of those actions.

Conclusions

- The objective of this paper is to study how to incorporate scripts that contain human knowledge into the tree policy of MCTS in order to improve its performance in the context of RTS games.
- We presented two tree policies (FC-GNS and k-GNS) that do such integration, one using a single script and the other using a collection of scripts.
- The experimental results showed that both tree policies can improve the gameplay strength upon the baseline algorithm. They also scale well to larger maps , where the baseline algorithm struggles to perform.

Thank You!