Contextual Combinatorial Bandits in Real-Time Strategy Games

Zuozhi Yang
Drexel University, Philadelphia, USA
zy337@drexel.edu

Santiago Ontañón*
Drexel University, Philadelphia, USA
so367@drexel.edu

***Abstract*—The contextual bandit problem is a richer framework than stochastic bandits that has many applications since it allows the learner has access to additional information (the "context"). This additional information can help predict the expected utility of the different arms in many cases. Moreover, combinatorial bandits are a class of bandit problem where the space of possible arms to choose from has a combinatorial structure. In this paper, we investigate the bandit problem where we have both contextual information and there is a combinatorial arm structure, which we call *contextual combinatorial bandits* (CCMABs). We apply contextual combinatorial bandits to real-time strategy (RTS) games, and study different algorithms to solve CCMABs with different trade-offs of computational efficiency and learning biases. Specifically, we focus on the problem of determining map-specific game playing policies, and formulate it as a CCMABs.**

I. Introduction

The Multi-Armed Bandit (MAB) problems [1] are amongst the most fundamental tools for modeling sequential decision problems under uncertainty. The core problem MABs tackle is the exploration vs. exploitation dilemma. In many applications, additional information and problem-specific structure make the problem richer and more challenging. Specifically, we look at two extensions to the MAB problem, contextual information and combinatorial structure. While both extensions have been well-studied separately, in this paper, we are interested in the scenario where we combine both, which we called contextual combinatorial multi-armed bandits (CCMAB). The CCMAB problem is challenging since both extensions add complexity to the problem. We propose and compare a few strategies for the problem using real-time strategy games as the testbed.

In classic stochastic bandit settings, the agent picks an arm in each round, and observes a reward generate from the underlining reward distribution of the arm. The goal is to maximize the expected cumulative reward by balancing exploration (learn more about arms that are less visited) and exploitation (take advantage of known good arms). Also, the performance of a bandit algorithm can be assessed through its expected cumulative regret. Regret is defined as the gap between the expected reward achieved by the algorithm and the expected reward of the best arm. MAB problems have been applied in many fields, including sequential clinical trials, network optimization, economics, e.g. [2], [3].

*Currently at Google

978-1-6654-3886-5/21/$31.00 ©2021 IEEE

Contextual bandit is a more general setting than the stochastic bandit setting for each iteration a context vector will be observed before an arm is selected. Over the course of learning, the context vector can be used to predict the reward distribution of unseen or less explored arms. Contextual bandit has been applied to many areas like online advertisement placement [4] and recommender systems [5].

The Combinatorial Bandits problem is a class of bandit problems with combinatorial arm structure. Specifically, unlike stochastic bandits where there is only one arm being pulled at each iteration, combinatorial bandits consist of multiple MABs and the combination of multiple arms from different MABs (called micro arms) forms the macro arm of the combinatorial bandits. Combinatorial bandits have found many applications like in network optimization and online ads placement [6], [7].

In this paper, we use real-time strategy (RTS) games as the application testbed for contextual bandits. We use CCMABs to find map-specific policies for unseen maps in RTS games. To make the problem tractable, we define a policy space defined by just six variables, one for each action type, and each variable has six discrete possible values. Thus, finding a policy means selecting one value for each variable. In a bandit setting, if we consider each policy to be an arm, the bandit problem of finding the best policy has a combinatorial structure. Moreover, we are interested in the problem of generating policies for unseen maps given a training set of maps, hence making the problem also contextual. We evaluate our approach with a collection of maps in the μRTS game simulator[1].

The remainder of this paper is organized as follows. First, we provide some necessary background on contextual bandits, combinatorial bandits, and research in RTS games in Section II. After that, we describe the contextual combinatorial bandit problem and propose three approaches to the problem in Section III. Then we demonstrate our experimental design and results in Section IV. Finally, the paper closes with discussions of the results, conclusions, and possible future work.

II. Background

Multi-armed bandit problem (MAB) is a class of problems where an agent facing a number of choices tries to balance the exploitation of existing knowledge and exploration of new knowledge. MAB has been studied in many topics such as clinical trials and ad-placement. The classic model is stochastic

[1]https://github.com/santiontanon/microrts

Algorithm 1: Stochastic Bandits

```
for each round $t$ do
    1. agent picks arm $v_t$
    2. agent observes reward $r_t$ for the chosen arm $v_t$.
```

Algorithm 2: Contextual Bandits

```
for each round $t$ do
    1. agent observes a context $\sigma_t$
    2. agent picks arm $v_t$
    3. reward $r_t$ observed for the chosen arm $v_t$.
```

bandit, which assumes a discrete, finite number, denoted as K, of arms for the agent to choose between at each time step t (the total time horizon is denoted as T if given). And each arm a has a reward distribution r_a. The pseudocode is shown in Algorithm 1. Most importantly, stochastic bandit assumes the reward distributions of the arms are independent and identically distributed (IID). However, in many applications, depending on the nature of specific problems, some generalization of stochastic bandits or specialized algorithmic design are required. In this section, we briefly introduce contextual bandits, combinatorial bandits, and why they are important in the application of RTS games.

A. Contextual Bandits

Contextual bandit is a generalization of stochastic bandits. In contextual bandits, rewards in each round depend on a context σ, which is observed by the agent prior to making a decision in each round. The general procedure a contextual bandit algorithm uses is shown in Algorithm 2. Contextual bandits make a different IID assumption: reward r_t is drawn independently from some distribution parameterized by the (σ_t, v_t) pair.

Contextual bandits are designed for the problem where agents can observe the context of arms and infer the rewards distribution of unseen arms. There has been a lot of work in this area, which includes algorithms such as LinUCB [8], Epoch Greedy [9], EXP4 [10], and EXP4.P [11]. There are similarities between the traditional supervised learning and contextual bandits. In the terminologies of classification problems, each context σ_t is an *example*, and the arms are the different *labels* for the example. Each label has an associated cost or reward. We can form a standard binary classification problem that there is one correct label with reward 1, and rewards for all other labels are 0. The key difference between the contextual bandit setting and standard supervised learning is that only the reward of the chosen action is revealed. Also, we can establish connections to reinforcement learning (RL). A 1-state RL problem is basically just a bandit. Contextual bandits are one step further than the stochastic bandits towards RL in that they consider the generalities across the states but the states do not have temporal relations so there is no credit-assignment problem. However, the problem of contextual bandits with combinatorial arm structure remains under-studied. We will describe the combinatorial bandits next.

B. Combinatorial Bandits

Combinatorial multi-armed bandits (CMAB) is a bandit problem specially designed to deal with inputs with combinatorial structure. Specifically, we use the following definition in this paper [12]. Specifically, a CMAB is defined by:

- A set of n variables $X = \{X_1, \ldots, X_n\}$, where variable X_i can take K_i different values, $X_i = v_i^1, \ldots, v_i^{K_i}$
- A reward distribution $R : X_1 \times \cdots \times X_n \to \mathbb{R}$ that depends on the value of each of the variables.
- A function $G : X_1 \times \cdots \times X_n \to \{true, false\}$ that determines which variable value combinations are legal.

Each v_i is called a *micro arm* or *local arm*, and each legal combination of local arms is called a *macro arm* or *global arm* V_t. The problem is to find a legal macro arm that maximizes the expected reward. Strategies to address CMABs are designed to iteratively sample the space of possible macro arms. Some existing work are MLPS [13], LSI [14], and NaïveSampling [15].

In terms of types of feedbacks in combinatorial bandits most of the existing work belongs to the semi-bandit type, where the player observes the outcomes of selected micro arms in each round. In this paper, we consider the bandit feedback, where the player only observes the reward of the global arm but no outcome of any local arm.

C. Real-Time Stratey Games and μRTS

Real-time strategy (RTS) is a sub-genre of strategy games where players aim to defeat their opponents (destroying their army and base) by strategically building an economy (gathering resources and building a base), military power (training units and researching technologies), and controlling those units. The main differences between RTS games and traditional board games are: they are simultaneous move games (more than one player can issue actions at the same time), they have durative actions (actions are not instantaneous), they are real-time (each player has a very small amount of time to decide the next move), they are partially observable (players can only see the part of the map that has been explored, although in this paper we assume full observability) and they might be non-deterministic.

RTS games have been receiving an increased amount of attention [16] as they are more challenging than games like Go or Chess in at least three different ways: (1) the combinatorial growth of the branching factor [15], (2) limited computation budget between actions due to the real-time nature, and (3) lack of forward model in most of research environments like Starcraft. Specifically, in this paper, we chose μRTS as our experimental domain, as it offers fast simulations suitable for testing bandit algorithms.

μRTS is a simple RTS game designed for testing AI techniques. μRTS provides the essential features that make RTS games challenging from an AI point of view: simultaneous and durative actions, combinatorial branching factors and real-time

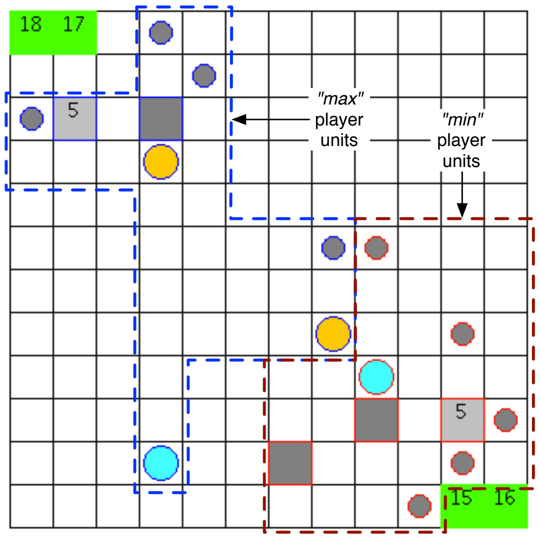


Fig. 1: A screenshot of μRTS, highlighting the units controlled by each of the two players.

decision making. The game can be configured to be partially observable and non-deterministic, but those settings are turned off for all the experiments presented in this paper. We chose μRTS, since in addition to featuring the above properties, it does so in a very minimalistic way, by defining only four unit types and two building types, all of them occupying one tile, and using only a single resource type. Additionally, as required by our experiments, μRTS allows maps of arbitrary sizes and initial configurations.

There is one type of environment unit (minerals) and six types of units controlled by players (bases, barracks, workers, and light, heavy and ranged military units). Additionally, the environment can have walls to block the movement of units. A example screenshot of game is shown in Figure 1. The squared units in green are Minerals with numbers on them indicating the remaining resources. The units with blue outline belong to player 1 and those with red outline belong to player 2. The light grey squared units are Bases with numbers indicating the amount of resources owned by the player, while the darker grey squared units are the Barracks.

μRTS policies can have many forms. For the purpose of studying CCMAB, we parameterize the policy with six variables, each of which represents the probability of making one type of action. For each variable, we discretize the probability to six buckets to represent it from low to high. Therefore, we have a policy space with combinatorial structure. The policy determines the behavior of all of the units throughout a game. And the context provided to the agent is a feature vector describing the map. We introduce the policy parameterization and context generation in detail in Section III. In this paper, our aim is to learn from a training set of maps and see if the agents can predict the reward distribution of new maps and choose suitable map-specific policies.

III. Contextual Combinatorial Bandits

In this section, we describe the contextual combinatorial multiarmed bandits (CCMAB) [17]. The problem deals with the situation where the bandit problem with combinatorial structure and a context is given before observing the reward. The context given can be at different levels of the combinatorial bandits. First, a context can be given at the macro arm level which potentially used to infer reward distribution of other macro arms. Second, contexts can be given at the micro arm level and the learning agent should combine these context to infer reward of other macro arm. For this paper, we focus on the first scenario where a context is given at the macro arm level.

In CCMAB, The reward r_t in each round t depends both on the context σ_t and the chosen macro arm V_t. And V_t consist of a set of n micro arms $V_t = \{v_t^1, \dots, v_t^n\}$.

The difference between CCMAB and contextual bandits is that in contextual bandits there is a single variable representing the arm to choose, while in CCMAB there are n variables to choose values for, defining a combinatorial space. And the difference between CCMAB and combinatorial bandits is that in combinatorial bandits, the reward r only depends on the chosen combination of n variables and in CCMAB the reward depends on both the context vector σ and the the chosen combination of arms.

To measure the performance of the strategies to address CCMAB, we use the notion *regret*, which is the difference between the expected reward of the selected macro arm V_t, and the expected reward of an optimal macro-arm V^*. There are many ways to compute the regret and the most commonly used and what we used in this paper is the *cumulative regret*, which is the sum of differences between r^* and the reward obtained by the selected macro arms at each iteration.

$$R_T = \sum_{t=1}^{T}(r^* - r_t)$$

Thus, in our problem settings, we will optimize the CCMAB for T iterations to find the best macro arm with the lowest cumulative regret or highest cumulative rewards.

We present and compare three approaches for contextual combinatorial bandits: naïve aggregation, pair-wise aggregation, and contextual global optimization approach.

A. Naïve Aggregation

In naïve aggregation, we make the naïve assumption that reward distribution of the macro arm can be factored by estimating a reward distribution for each micro-arm, i.e. we assume the reward for each micro-arm is independent. Because of the naïve assumption, we can break down the CCMAB to n contextual bandit problems. As shown in Algorithm 3, at each round t, the agent first observes a context vector σ_t. σ_t has d dimensions to represent d features of the context. Then each micro MAB observes σ_t and pick a micro arm v_t^n according to pre-defined contextual bandit algorithm. For example, if we use contextual bandit algorithms with a policy

Algorithm 3: Naïve Aggregation

```
for each round t do
    1. macro MAB observes the context $\sigma_t$
    2. each micro MAB $X_n$ picks an arm $v_t^n$ according
       to learner $L^{X_n}$
    3. macro MAB $V_t$ picks the combination formed
       by $\{v_t^1, \ldots, v_t^n\}$.
    3. reward $r_t$ observed for the chosen marcro arm,
       and $r_t$ is fed back to each $v_t^n$.
```

class as mentioned before. A policy is based on a machine learning model L, and the learning algorithm is used to predict the mean reward. The macro arm is selected as the legal combination of micro arms with the highest sum of mean rewards. Since we have n micro MABs, for each micro MAB X_n we have a learner L^{X_n}.The reward r_t observed is fed back to each bandit policy of micro MABs.

The advantage of naïve aggregation is that thanks to the naïve assumption, the problem is simplified from $O(K^n)$ to $O(nK)$, with n being the number of micro MABs and K being the number of arms of each micro MAB. However, the efficiency we gained is because of we trade-off the interdependence between the local MABs. Thus, the next technique, called pair-wise aggregation, tries to bring back some of the interdependence on top of the naïve aggregation.

B. Pair-Wise Aggregation

As we have discussed, the naïve aggregation trade-off interdependence of micro MABs for efficiency. Here we describe pair-wise aggregation that brings back partial interdependence. We consider a pair of micro arms of two different micro MABs $\{a_i^{X_n}, a_j^{X_m}\}$ to be an arm in the pair-wise MAB. Thus, for a macro MAB with n K-armed micro MABs, the number of legal pairs will be $O(K^2n^2)$. And each macro arm V_t picked at time t, will generate training examples of K^2 pairs.

Algorithm 4 demonstrates how pair-wise aggregation works. On top of naïve aggregation, we build pair-wise learners on the side and pick arms by merging the two. First, the agent observe the context vector σ_t and reveal σ_t to both micro MAB learners L and all the relevant pair-wise learners P. L and P will separately predict the expected reward. Then we can pick the macro arm according to the prediction. Finally, the observed reward is revealed to L and P to update the learners.

The advantage of adding pair-wise dependence is that it is a more realistic modelling of many real world problems. For example in the news placement example, it is very likely that the news showed together are in the same category or theme. Adding modeling of pair-wise relations allows us to capture some interdependence of the micro MABs.

C. Contextual Global Optimization

In this section we describe a simple approach of incorporating the full interdependence between the micro arms. In

Algorithm 4: Pair-Wise Aggregation

```
for each round t do
    1. macro MAB observes the context $\sigma_t$
    2. each micro MAB $X_n$ picks an arm $v_t^n$ according
       to learner $L^{X_n}$ and all relevant pair-wise learner
       $P^{X_n \times X_m}$
    3. macro MAB $V_t$ picks the combination formed
       by $\{v_t^1, \ldots, v_t^n\}$.
    3. reward $r_t$ observed for the chosen macro arm,
       and $r_t$ is fed back to each $v_t^n$ and $P^{X_n \times X_m}$
```

Algorithm 5: Contextual Global Optimization

```
for each round t do
    1. macro MAB observes the context $\sigma_t$
    2. Pick the macro-arm $V_t$ according to the context
       $\sigma_t$ using the context learner $L$ as
       $argmax_{V_t \in X} L([\sigma, V_t])$.
    3. reward $r_t$ observed for $V_t$.
    4. Construct training example $\{[\sigma, V_t], r_t\}$ and feed
       it back to the context learner..
```

this strategy, we "flatten" the combinatorial structure and only consider the macro arms. We refer to this method as contextual global optimization. For a problem with context vector σ that has d dimension and m possible values in each dimension, the number of contexts is $O(m^d)$. And for combinatorial bandits, the number of macro arms is $O(K^n)$. To apply contextual global optimization, we construct *context-macroarm* vectors, which consist of the concatenation of the context vector σ to the vector representing a given macro-arm, and then we can train a machine learning model (the *context learner* L) to predict the expected reward of a given macro-arm in a given context by learning to predict the reward given this context-macroarm vector. Essentially we convert the CCMAB problem into a contextual bandit problem where the macro-arm is represented by the specific micro-arms that compose the macro-arm.

Specifically, during the bandit optimization process, we first receive the context vector σ, then we select the macro arm according to the contextual learner. When the reward r_t is observed, we construct a training example as $\{[\sigma, V_t], r_t\} = \{[\sigma_1, \ldots, \sigma_d, v_t^1, \ldots, v_t^n], r_t\}$. This example is given to a context learner, which is retrained every iteration with the set of collected training examples to enable the model to generalize. The learner can be any machine learning model like a decision tree, a Bayesian net, neural network depending on the need (for simplicity, we used decision trees in our experiments, which gave us the best tradeoff of speed/performance in our experiments given the amount of data we collect).

By applying contextual global optimization we convert the problem to a contextual bandit problem with $O(K^n)$ arms and context space of $O(m^d)$. In this way, we have the benefit of capturing the full interdependence of the micro MABs and we can apply contextual bandit algorithms like ϵ-greedy with classification algorithms, as shown in Algorithm 5.

When applying ϵ-greedy, in each iteration, we have ϵ chance picking a random arm, and $1 - \epsilon$ chance, we find the one for which the context learner predicts the highest reward given the current context. This requires an *argmax* operation on L. While some differentiable models allow for an implementation of such argmax operation in an efficient way, in our experiments with a decision tree, we just iterate over all possible macro arms and obtain the one for which L reports the highest reward (this can implemented more efficiently than full iteration over the complete combinatorial macro-arm space, by iterating over the set of leaves of the decision tree, for example).

IV. Experiments and Results

In this section, we describe the experimental setup and report results. In order to evaluate the three methods described above, we perform experiments in μRTS. A collection of 6 maps of different starting configurations are selected as our different "contexts". The goal is to use proposed bandit algorithms to choose good parameterized stochastic game-playing policies for a new map they had not seen before. To test this, we train the learning models of our bandits in five of those maps, and then we test them in the sixth, unseen, map in a leave-one-out setting.

A. Policy Parameterization

We employ a simple stochastic parameterization of the policy, where we define a weight vector $\mathbf{w} = (w_1, ..., w_6)$, where each of the six weights $w_i \in [0, 1]$ corresponds to each of the six types of actions in the game:

- NONE: no action.
- MOVE: move to an adjacent position.
- HARVEST: harvest a resource in an adjacent position.
- RETURN: return a resource to a nearby base.
- PRODUCE: produce a new unit (only bases and barracks can produce units, and only workers can produce new buildings).
- ATTACK: attack an enemy unit that is within range.

A policy is totally represented by the vector $\mathbf{w}$. During gameplay, the action for each unit is selected proportionally to this weight vector. To choose the action for a given unit, the following procedure is used: given all the available actions for a unit, a probability distribution is formed by assigning each of these actions the corresponding weight in $\mathbf{w}$, and then normalizing to turn the resulting vector into a probability distribution. If the weights of all the available actions are 0, then an action is chosen uniformly at random. Notice that this defines a very simple space of policies, but as we will see below, it is surprisingly expressive, and includes policies that are stronger than it might initially seem.

B. Bandit Optimization of Gameplay Policy

We discretize each value w_i in $\mathbf{w}$ to six values $[0, 1, 2, 3, 4, 5]$. Thus, the optimization of the policy can be modeled as a combinatorial bandit where the macro MAB is all of the possible policies and each macro arm consists of six micro MABs with six micro arms to represent the discretized values. The reward are calculated from the game play results against our target bot called `RndBiased` bot, which is built-in intro μRTS. `RndBiased` is actually expressible in our space of policies. It is a biased random agent where HARVEST, RETURN, and ATTACK has five times the weights than other action types (approximate weight vector $[0.06, 0.06, 0.28, 0.28, 0.06, 0.28]$).

To test whether the optimized contextual combinatorial bandit can generalize to unseed contexts, we employ the *leave-one-out* cross validation technique, which means we select one of the maps as testing map, train the agent on the rest of the maps, and repeat the optimization for each map. Thus, the optimization cycle works like this:

1) A random map from the training set is chosen and the corresponding context vector σ is revealed to the agent.
2) The policy picked by the bandit algorithm will play 10 games against the target bot in the chosen map.
3) The average winrate is recorded as the reward and revealed to the bandit algorithm.

This optimization process is repeated for 10000 iterations. After the optimization, the context vector of the testing map is revealed to the bandit. The selected testing policy will run 1000 games against target bot in the testing map. And the average winrate is recorded as final performance of the bandit algorithm for further comparison.

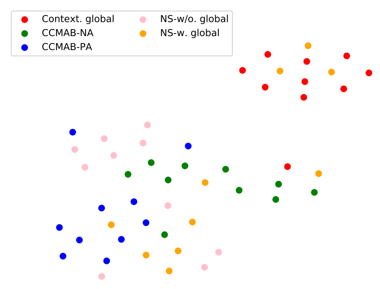
Additionally, we add two more baseline algorithms for comparison: naïve sampling (with global optimization) [15] and naïve sampling without global optimization. The two algorithms serve as ablation studies of the contextual bandits, because they captures the combinatorial structure but not the contextual information. Naïve sampling is a combinatorial multiarmed bandit strategy that is *not* contextual, hence it will just find the best macro-arm in the set of training maps, and use that in the test map, without adapting the policy to the test map. Naïve sampling without global optimization is the same as Naïve sampling, but removing the *global MAB* (this is to have a direct non-contextual comparison to our Naïve Aggregation approach).

We select six maps, listed below, as our set of context maps.

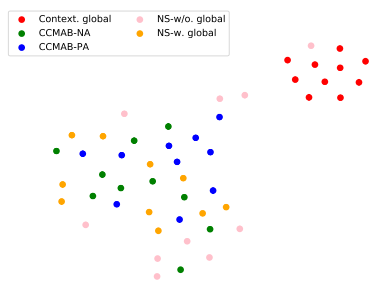
- Map 1: *8x8/basesWorkers8x8A.xml*: In this map of size 8 by 8, each player starts with one base and one worker. Games are cut-off at 3000 cycles.
- Map 2: *8x8/FourBasesWorkers8x8.xml*: In this map of size 8 by 8, each player starts with four bases and four worker. Games are cut-off at 3000 cycles.
- Map 3: *OneWorker8x8.xml*: In this map of size 8 by 8, each player starts with one worker and more resources. Games are cut-off at 3000 cycles.
- Map 4: *FourWorker8x8.xml*: In this map of size 8 by 8, each player starts with four workers. Games are cut-off at 3000 cycles.
- Map 5: *FourRanged8x8.xml*: In this map of size 8 by 8, each player starts with four ranged units. Games are

TABLE I: Cross-validated winrates for naïve aggregation, pair-wise aggregation, contextual global optimization, and two baseline algorithm based on naïve sampling.

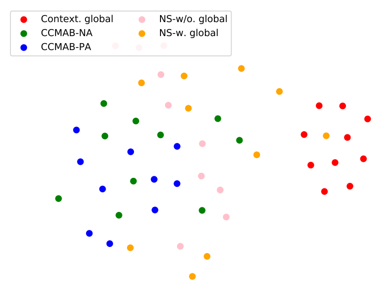
	Naïve Agg.	Pair-wise Agg.	Contextal Global Opt.	NS w/o. Global Opt.	NS w. Global Opt.
Map 1	0.839	**0.886**	0.587	0.851	0.759
Map 2	0.528	0.515	**0.898**	0.487	0.524
Map 3	0.738	0.738	**0.747**	0.728	0.735
Map 4	0.922	0.922	**0.932**	0.910	0.914
Map 5	0.946	0.941	**0.949**	0.939	0.941
Map 6	0.903	0.914	0.907	**0.921**	0.915
Average	0.812	0.819	**0.836**	0.806	0.798



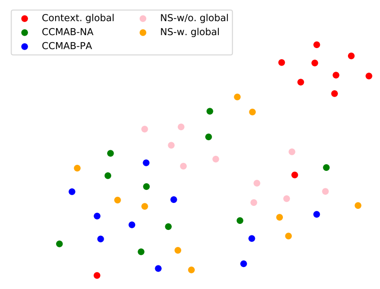
(a) t-SNE Plot for Map1



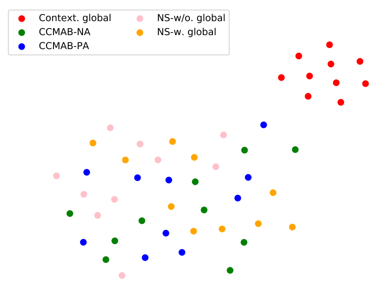
(b) t-SNE Plot for Map2



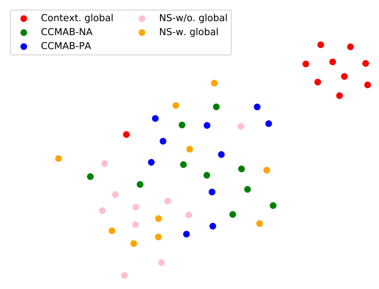
(c) t-SNE Plot for Map3



(d) t-SNE Plot for Map4



(e) t-SNE Plot for Map5



(f) t-SNE Plot for Map6

Fig. 2: t-SNE visualizations for the policies selected by the five algorithms in the six maps over ten runs.

cut-off at 3000 cycles.

- Map 6: *TwoRangedLight8x8.xml*: In this map of size 8 by 8, each player starts with two ranged units and two light units. Games are cut-off at 3000 cycles.

The context vector calculated for the maps is simple: a vector of the number of each type of units for either player. For example, Map 1 has the corresponding context vector $[1, 0, 1, 0, 0, 0]$.

C. Implementation Details

In this part, we describe the implementation details for the three algorithms. Overall, we use decision trees as the machine learning model as our policy learner. The reason is that during the bandit optimization, the dataset is often unbalanced, and decision trees are less sensitive to unbalanced dataset. Also, the computational complexity of training and testing decision trees are low comparing to more complicated methods. Specifically, we use the J48 model provided by Weka.

For naïve aggregation, we trained a J48 classifier for each micro arm. Thus, in total we have $K \cdot n = 36$ classifiers. We uses ϵ-greedy to balance exploration and exploitation, which means for each iteration we have ϵ probability of selecting a random macro arm, and $1 - \epsilon$ probability of selecting the best macro arm. We use a linearly decaying ϵ in our experiments, starting at $\epsilon = 0.5$ and reaching $\epsilon = 0.00$ in the last optimization iteration. In naïve aggregation, the best macro arm is the combination of the micro arms with best winrate separately. Thus, we perform a greedy selection of micro arms to construct the best macro arm.

For pair-wise aggregation, we have two parts. The first part is identical to the naïve aggregation model. The second part we have the modelling for pair-wise relations. We train one J48 classifiers for each pair of micro arms from different

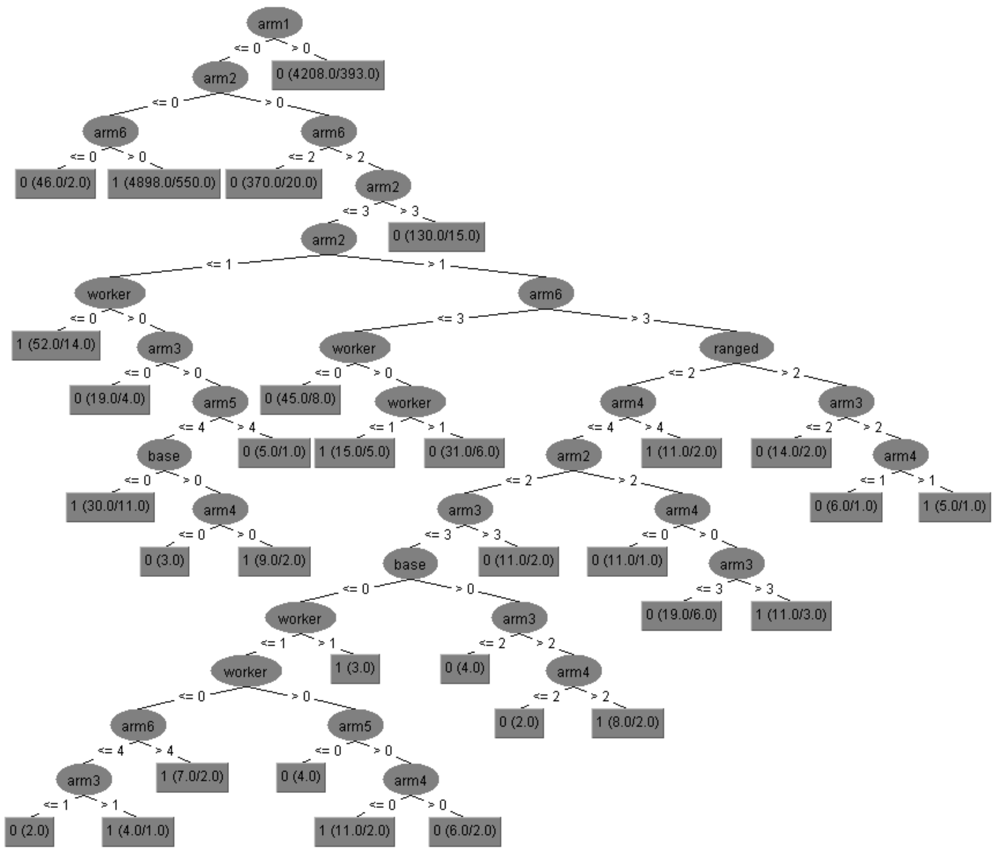


Fig. 3: Example visualization of the decision tree trained from contextual global optimization.

macro MABs. In total, we have $\frac{n(n-1)}{2} \cdot K^2 = 540$ classifiers. Exploration/exploitation is the same as naïve aggregation, the same reward will be revealed the pair-wise learners. For naïve aggregation, greedy selection will give us the macro arm with highest average winrate. For pair-wise aggregation, we go over all combinations to find the macro arm with highest average score. Once we have the naïve aggregation and the pair-wise score for each macro-arm, we just calculate the average score for each macro-arm and select the one with the highest score. Again, notice that this requires iterating over all possible macro-arms (which is feasible in our experiments), although mode efficient ways to do this process will be devised in future work.

For contextual global optimization, we simply have one classifier that takes the context-macroarm vector as the input. Again, we use ϵ-greedy to balance exploration and exploitation, with linearly decaying ϵ starting at $\epsilon = 0.5$ and going down to zero. When we are making exploitation selections, the macro arm of highest predicted winrate with respect to the current context is selected. The picked augmented macro arm and reward received together will be the training examples for the classifier.

D. *Experimental Results*

In our experiments we compare the policies resulting from the different bandit strategies using the cross-validated win rate against the target bot. As we can see in Table I, the best performing method overall is global contextual optimization with 0.836 total winrate and being the best bot in four maps. The next is pair-wise aggregation strategy with 0.819 total winrate, followed by naïve aggregation with 0.812 winrate. We can see that considering the interdependence between the micro MABs are clearly helpful for the performance of the optimization, since pair-wise aggregations outperformed naïve aggregation, and global optimization outperformed pair-wise aggregation. Moreover, the contextual global optimization approach performed the best or close to the best in all maps except for Map 1 (we explain the reason for this below).

The two baseline methods that do not consider contextual information have the worst performance. Interestingly, Naïve Sampling without global optimization slightly outperformed Naïve Sampling with global optimization, with 0.806 and 0.798 winrate respectively, probably due to overfitting.

Then we used *t-SNE* [18] to visualize the resulting macro arms of the algorithms in the six maps, shown in Figure 2. Sub-figure (a) to (f) in Figure 2 represents the results in Map 1-6 respectively. And in each sub-figure, the results of each algorithm is labeled with a different color. There each algorithm is run 10 times, thus for each algorithm, there are 10 points representing results from separate runs. A clear phenomenon we can observe in this visualization is that the result of the global contextual optimization are quite different from the ones of other algorithms. In fact, contextual global optimization almost always selects the macro arm that are effectively equivalent to $[0, 0, 0, 0, 0, 1]$ across all of the maps.

The reason can be found from the visualization of one of the trees that were learned in our experiments, shown in Figure 3. In this decision tree, any policy with the first two element being 0 and the last element greater than 0 will be classified to win with high conviction. Due to the order in which we search the macro-arms for selecting the best, this results in selecting the macro-arm mentioned above. Our interpretation is that the decision tree finds a solution that is performing well and generalizable in most of the maps that we use. And Map 1 is an example of this solution fails to generalize. We believe this phenomenon occurs as the set of maps we used for experimentation are not varied enough as for the model to learn generalizable policies, and context learner often learned that the $[0, 0, 0, 0, 0, 1]$ policy does well in all maps. Hence, this points out the need for a more varied training set. However, it is worth noting that it is interesting that such a simple strong policy exists in the space and Naïve Sampling did not find it. We believe this is because the machine learning model is able to perform generalizations such as ("arm $1 > 2$") that the probabilty estimation procedure done in Naïve Sampling cannot capture.

V. Discussion and Conclusions

In this paper we have introduced the CCMAB problem, which corresponds to the bandit problem with contextual information and combinatorial arm structures. Then we proposed three strategies to tackle the CCMAB problem: naïve aggregation, pair-wise aggregation, and contextual global optimization. The three strategies consider no micro arm interdependence, partial micro arm interdependence, and full micro arm interdependence. Specifically, naïve aggregation only consider micro MABs separately and does greedy aggregation. Pair-wise aggregation adds the pair-wise relations of the micro MABs upon the naïve aggregation. Lastly, contextual global optimization consider the full interdependence by treating combinatorial structure of arms as additional contexts.

Then we designed experiments in μRTS to compare the three strategies. Specifically, the task is to find the best map-specific game-playing policies whose parameterization have the combinatorial structures in unseen maps given a set of training maps. The result showed that the more interdependence that the strategy considers, the better performance it can obtain. Also, through the t-SNE visualization, we observed that the strategy that considers the full interdependence of the micro arms generate policies that differ more from the ones that does not consider full interdependence or contextual information.

For future work, we believe there are still large space for improvements for the CCMAB strategies. For example, the scalability of the algorithms can be crucial in more complex problems, as our implementations often rely on macro-arm enumeration. Also, for exploration/exploitation balancing, we simply used ϵ-greedy strategies. Further, we want to study the regret bounds of these algorithms, which we did not study in this paper, where we limited ourselves to empirical experimentation. Finally, we want to apply our solution to the CCMAB problem to domains like player modeling, where MAB approaches have shown promise [19] and to Monte Carlo Tree Search, by selecting contextualized policies for either the tree policy or the default policy.

References

[1] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[2] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *arXiv preprint arXiv:1204.5721*, 2012.

[3] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[4] K. Hofmann, S. Whiteson, M. de Rijke *et al.*, "Contextual bandits for information retrieval," in *NIPS 2011 Workshop on Bayesian Optimization, Experimental Design, and Bandits, Granada*, vol. 12, 2011, p. 2011.

[5] J. Mary, R. Gaudel, and P. Preux, "Bandits and recommender systems," in *International Workshop on Machine Learning, Optimization and Big Data*. Springer, 2015, pp. 325–336.

[6] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1466–1478, 2012.

[7] A. Nuara, F. Trovo, N. Gatti, and M. Restelli, "A combinatorial-bandit algorithm for the online joint bid/budget optimization of pay-per-click advertising campaigns," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[8] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 661–670.

[9] J. Langford and T. Zhang, "The epoch-greedy algorithm for contextual multi-armed bandits," *Advances in neural information processing systems*, vol. 20, no. 1, pp. 96–1, 2007.

[10] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM journal on computing*, vol. 32, no. 1, pp. 48–77, 2002.

[11] A. Beygelzimer, J. Langford, L. Li, L. Reyzin, and R. Schapire, "Contextual bandit algorithms with supervised learning guarantees," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 19–26.

[12] S. Ontanón, "The combinatorial multi-armed bandit problem and its application to real-time strategy games," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 9, no. 1, 2013.

[13] Y. Gai, B. Krishnamachari, and R. Jain, "Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation," in *2010 IEEE Symposium on New Frontiers in Dynamic Spectrum (DySPAN)*. IEEE, 2010, pp. 1–9.

[14] A. Shleyfman, A. Komenda, and C. Domshlak, "On combinatorial actions and cmabs with linear side information," in *ECAI*, 2014, pp. 825–830.

[15] S. Ontañón, "Combinatorial multi-armed bandits for real-time strategy games," *Journal of Artificial Intelligence Research*, vol. 58, pp. 665–702, 2017.

[16] S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A survey of real-time strategy game ai research and competition in starcraft," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 5, no. 4, pp. 293–311, 2013.

[17] L. Chen, J. Xu, and Z. Lu, “Contextual combinatorial multi-armed bandits with volatile arms and submodular reward,” *Advances in Neural Information Processing Systems*, vol. 31, pp. 3247–3256, 2018.

[18] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[19] R. C. Gray, J. Zhu, D. Arigo, E. Forman, and S. Ontañón, “Player modeling via multi-armed bandits,” in *International Conference on the Foundations of Digital Games*, 2020, pp. 1–8.