ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-825

On Combinatorial Actions and CMABs with Linear Side Information

Alexander Shleyfman and **Antonín Komenda** and **Carmel Domshlak**[1]

Abstract. Online planning algorithms are typically a tool of choice for dealing with sequential decision problems in combinatorial search spaces. Many such problems, however, also exhibit combinatorial actions, yet standard planning algorithms do not cope well with this type of "the curse of dimensionality." Following a recently opened line of related work on combinatorial multi-armed bandit (CMAB) problems, we propose a novel CMAB planning scheme, as well as two specific instances of this scheme, dedicated to exploiting what is called linear side information. Using a representative strategy game as a benchmark, we show that the resulting algorithms very favorably compete with the state-of-the-art.

1 INTRODUCTION

In large-scale sequential decision problems, reasoning about the problem is often narrowed to a state space region that is considered most relevant to the specific decision problem currently faced by the agent. In particular, *online planning* algorithms focus only on the current state s_0 of the agent, deliberate about the set of possible courses of action from s_0 onwards, and, when interrupted, use the outcome of that exploratory deliberation to select an action to perform at s_0. Once that action is applied in the real environment, the planning process is repeated from the obtained state to select the next action and so on.

The basic components of any sequential decision problem are its states and actions. When the number of actions is polynomial in the size of the problem description, the basic computational complexity of planning stems solely from a prohibitive—exponential in the size of the problem representation—size of the state space. This "curse of state dimensionality" seems to receive most of the attention in the automated planning research. In particular, state-space forward search algorithms, including all standard, both systematic and Monte-Carlo, online planning algorithms, implicitly assume that the action choices at any state can be efficiently enumerated.

Whatever the atomic actions of the agents are, as long as the agent can perform only one (or a small fixed number of) atomic actions simultaneously, the above assumption of enumerable action choices is typically fine. However, if the agent we are planning for either represents a team of cooperating agents, or, equivalently, has a number of concurrent actuators, then the problem exhibits a "curse of action dimensionality" via the *combinatorial structure of the action space*.

Real-time strategy games (RTS) are a great example of decision problems with combinatorial action spaces, because a player is asked to activate *in parallel* a set of units that together form the force of the player [2, 6, 7, 15, 17]. That is, the set of actions $A(s)$ available to a player at a state s corresponds to a (sometimes proper, due to some game-specific constraints) subset of the cross-product of explicitly given sets of atomic actions of her units.

Previous work on online planning in RTS mostly avoided dealing with combinatorial actions directly, either by sequencing the decisions made for individual units [7, 15], or by abstracting the combinatorial action spaces to a manageable set of choices [2, 6]. The exception seems to be a recent work of Ontañón [16] that suggested considering combinatorial actions in online planning through the lens of *combinatorial multi-armed bandit (CMAB)* problems [11, 5, 16]. In particular, Ontañón suggested a specific Monte-Carlo algorithm for online planning in CMABs, called Naive Monte-Carlo (NMC), that is driven by an assumption that the expected value of a combinatorial action can be faithfully approximated by a linear function of its components. Evaluated on the μRTS game, NMC was shown to favorably compete with popular search algorithms such as UCT and alpha-beta ABCD, which avoid dealing with combinatorial actions directly [16].

Taking on board the CMAB perspective on combinatorial actions in sequential decision problems, here we continue the study of online planning algorithms for CMABs. In particular, we formalize the basic building blocks of such algorithms, as well as the tradeoffs in computational resource allocation between them. Based on this analysis, we suggest a simple, two-phase scheme for CMAB planning, in which the first phase is dedicated solely to generating candidate combinatorial actions, and the second phase is dedicated solely to evaluating these candidates. Adopting the assumption of helpful linear "side information", we then propose two instances of this two-phase scheme, $\mathsf{LSI_V}$ and $\mathsf{LSI_F}$, that, in particular, build upon some recent developments around action selection in regular MAB problems [3, 4, 12]. Our experimental evaluation on the μRTS game as in [16] shows that both $\mathsf{LSI_V}$ and $\mathsf{LSI_F}$ substantially outperform NMC, as well as emphasizes the marginal value of both exploiting side information and of systematicity in candidate evaluation process.

2 BACKGROUND

The *multi-armed bandit (MAB)* problem is a sequential decision problem defined over a single state. At each stage, the agent has to execute one out of some $k \geq 2$ stochastic actions $\{a_1, \ldots, a_k\}$, with a_i being parameterized with an unknown distribution ν_i, with expectation μ_i. If a_i is executed, the agent gets a reward drawn at random from ν_i.

Most research on MABs has been devoted to the setup of reinforcement learning-while-acting, where the performance of the agent is assessed in terms of its cumulative regret, the sum of differences between the expected reward of the best arm and the obtained rewards. Good algorithms for learning-while-acting in MAB,

[1] Technion, Israel, email: alesh@tx,akomenda@tx,dcarmel@ie.technion.ac.il

like UCB1 [1], trade off between exploration and exploitation. These MAB algorithms also gave rise to popular Monte-Carlo tree search algorithms for online planning in multi-state sequential decision problems (e.g., MDPs and sequential games), such as ε-MCTS [18], UCT [14], and MaxUCT [13].

However, as it was first studied in depth by Bubeck et al. [4], learning-while-acting and online planning are rather different problems that should favor different techniques. Unlike in learning-while-acting, the agent in online planning may try the actions "free of charge" a given number of times N (not necessarily known in advance) and is then asked to output a recommended arm. The agent in online planning is evaluated by his *simple regret*, i.e., the difference $\mu^* - \mu_i$ between the expected payoff of the best action and the average payoff obtained by his recommendation a_i. In other words, the rewards obtained by the agents at planning are fictitious. Therefore, good algorithms for online planning in MABs, like uniform-EBA [4], Successive Rejects [3], and SequentialHalving [12], are focused solely on exploration, and they already gave rise to efficient Monte-Carlo tree search algorithms for online planning in multi-state sequential decision problems such as BRUE [9] and MaxBRUE [10].

In contrast to regular MAB problems, in which rewards are associated with individual actions and a single action is executed at each stage, in *combinatorial multi-armed bandit (CMAB) problems*, the rewards are associated with certain subsets of actions, and the agent is allowed to simultaneously execute such subsets of actions at each stage [11, 5, 16]. In terms closest to problems that motivated our work in the first place, i.e., sequential decision problems for teams of cooperative agents, a CMAB problem is given by a finite set of $n \geq 1$ classes of actions $\{A_1, \ldots, A_n\}$, with $A_i = \{a_{i;1}, \ldots, a_{i;k_i}\}$, and a constraint $\mathcal{C} \subseteq \mathcal{A} = [A_1 \cup \{\epsilon\}] \times \cdots \times [A_n \cup \{\epsilon\}]$, where ϵ denotes "do nothing", and thus $\mathcal{A}$ is the set of all possible subsets of actions, with at most one representative from each action class. We refer to every set of actions $\mathbf{a} \in \mathcal{A}$ as a combinatorial action, or *c-action*, for short. Each c-action $\mathbf{a}$ is parameterized with an unknown distribution $\nu(\mathbf{a})$, with expectation $\mu(\mathbf{a})$. At each stage, the agent has to execute one out of some $2 \leq K = |\mathcal{C}| \leq \prod_{i=1}^{n} k_i$ c-actions, and if c-action $\mathbf{a}$ is executed, then the agent gets a reward drawn at random from $\nu(\mathbf{a})$.

Whether our setup is online planning in CMABs or learning-while-planning in CMABs, it is easy to see that CMAB problems with $K = O(poly(n))$ can be efficiently approached with regular MAB algorithms. However, if the problem is only loosely constrained and thus the c-action space grows exponentially with n (as it is typically the case in RTS-like planning problems), then the algorithms for regular MAB problems are no-go because they all rely on assumption that each c-action can be sampled at least once. This led to devising algorithms for CMAB learning-while-planning [11, 5] and online planning [16], all making certain assumptions of "side information", usefulness of which depends (either formally or informally) on the properties of μ over the polytope induced by $A_1 \times \cdots \times A_n$. Such a "side information" basically captures the structure of μ targeted by the algorithm, but the algorithm can still be sound for arbitrary expected reward functions. This is, for instance, the case with the Naive Monte-Carlo algorithm of Ontañón [16], which we describe in detail and compare to, later on.

3 ONLINE PLANNING IN CMABS

Due to the "curse of action space dimensionality", at a high level, any algorithm for online planning in CMABs should define two strategies:

1. a *candidate generation strategy*, for reducing the set of candidates from $\mathcal{C}$ to a reasonably small subset $\mathcal{C}^* \subseteq \mathcal{C}$ of candidates, and
2. a *candidate evaluation strategy*, for identifying the best c-action in $\mathcal{C}^*$ by gradually improving the corresponding estimates of μ.

Given such a pair of strategies, the overall algorithm can then apply them, either sequentially or in interleaving, to sample the selected c-actions. The question is, of course, what pair of strategies to adopt, and how to combine between them so to best exploit the available planning time. The only previous proposal in that respect corresponds to the recent Naive Monte-Carlo (NMC) algorithm of Ontañón [16]. At a high level, NMC constitutes a composition of ε-greedy sampling strategies, operated under an assumption that μ is linear in the atomic actions of the CMAB, i.e., $\mu(\mathbf{a}) = \sum_{i=1}^{n} \sum_{j=1}^{k_i} 1_{\{a_{i;j} \in \mathbf{a}\}} w_{i;j}$, where $1_{\{\cdot\}}$ is the indicator function, and $w_{i;j} \in \mathbb{R}$. Specifically, at each stage, NMC follows the candidate generation/evaluation strategy with probability $\varepsilon_0/(1-\varepsilon_0)$, respectively, where:

1. The candidate generation strategy generates and samples a candidate c-action $\mathbf{a}$ by selecting atomic actions from each set A_i independently and ε-greedily: with probability ε_1, $\mathbf{a}$ will contain the "empirically best atomic action" from A_i, and with probability $(1-\varepsilon_1)$, the i-th component of $\mathbf{a}$ will be selected from A_i uniformly at random. Atomic action $a_{i;j}$ is "empirically best" in A_i if, so far, the average reward of the (c-action) samples involving $a_{i;j}$ is the highest among the elements of A_i.
2. The candidate evaluation strategy samples the empirically best action in (the current) set $\mathcal{C}^*$.

At the end, the algorithm output (and the agent performs) the best empirical action in $\mathcal{C}^*$.

Assuming $\varepsilon_0, \varepsilon_1 < 1$, every c-action in $\mathcal{C}$ will eventually be generated and then sampled infinitely often. Thus, NMC converges in the limit to the best c-action in $\mathcal{C}$, and this independently of whether the assumption of μ's linearity in atomic actions actually holds. Moreover, in an empirical evaluation on μRTS game in [16], NMC was shown to substantially outperform the standard tree search algorithms, such as UCT and ABCD, showing the promise of CMAB planning algorithms in decision problems with combinatorial actions. This precisely was the departing point for our work here.

Considering the dynamics of NMC, we note that candidate generation and candidate evaluation in it are stochastically interleaved, and the interleaving is made at the resolution of single samples. One possible motivation for such an interleaving might be in exploiting samples made at the evaluation samples to improve the estimated side information (aka linear function coefficients) for the generation steps. However, a closer look suggests that such an interleaving of candidate generation and candidate evaluation disadvantages the planning process twofold.

- If m samples are getting devoted directly to candidate generation, then the algorithm will generate up to m new candidates. Thus, even if the side information assumptions hold, a vast majority of the candidates will unavoidably be generated without a quality guidance of this side information as the latter is being acquired gradually over the candidate generation process.
- More importantly, while NMC converges to the best c-action in the limit, for no reasonable budget of samples $N = o(K)$ it can provide any meaningful guarantees on the quality of the recommended action, not only with respect to the entire set of choices $\mathcal{C}$ (which is understandable), but even *with respect to the generated subset of candidates* $\mathcal{C}^*$.

The latter issue appears to be especially concerning, and, in particular, it stems from the fact that, after any number $N = o(K)$ of samples, the best empirical mean among the c-action in $\mathcal{C}^*$ might be based on just a single sample of the respective c-action.

Taking that on board, in what follows we examine the prospects of algorithms that exhibit no interleaving of candidate generation and candidate evaluation at all. These algorithms take a simple two-phase approach of dividing the overall sample allowance N between the candidate generation phase that *runs first*, using N_g samples, and the candidate evaluation phase that *runs second*, using $N_e = N - N_g$ samples. The motivation behind this simple two-phase scheme is twofold. Fixing some k c-action candidates $\mathcal{C}^*$ induces a problem of online planning in regular MAB, and state-of-the-art algorithms for this problem guarantee that the probability of choosing sub-optimal c-action from $\mathcal{C}^*$ decreases *exponentially* with N_e [4, 3, 12]. Reversely, suppose that, given a sample allowance N_e for the candidate evaluation phase, the algorithm of our choice for this phase guarantees that the recommended c-action will indeed be the best among $\mathcal{C}^*$ with probability of at least $\delta(k)$. If we are interested in choice-error probability of at most δ, then there is no point in coming up with more than some $k(\delta, N_e)$ candidate c-actions, and thus the candidate generation phase can/should be optimized to selection of precisely that number of candidates.

In what follows, we suggest and evaluate two simple variants of two-phase online planning for CMAB, $\mathsf{LSI_V}$ (short for "linear side information from vertices") and $\mathsf{LSI_F}$ (short for "linear side information from facets"). Both algorithms assume the same type of helpful side information, namely that μ is faithfully approximated by a function that is linear in the atomic actions of the CMAB, and differ only in the way this side information is actually estimated. Some auxiliary notation: $[\![n]\!]$ for $n \in \mathbb{N}$ denotes the set $\{1, \ldots, n\}$. For a finite-domain, non-negative, real-valued function f, $\mathcal{D}[f]$ denotes a probability distribution over the domain of f, obtained by normalizing f as a probability function using a normalization of our choice. For such a probability distribution $\mathcal{D}[f]$ and a non-empty subset S of the f's domain, by $\mathcal{D}[f \restriction_S]$ we refer to the conditional of $\mathcal{D}[f]$ on S. Finally, the operation of drawing a sample from a distribution $\mathcal{D}$ is denoted by $\sim \mathcal{D}$.

Figure 1a depicts the two-phase sampling scheme underlying both $\mathsf{LSI_V}$ and $\mathsf{LSI_F}$. Given a partition of sample budget N into N_g and N_e, the algorithms first generates $k(N_e)$ c-actions (GENERATE), and then evaluates these c-actions to recommend one of them (EVALUATE). The GENERATE procedure comprises

(1) generating a weight function $\widehat{R}$ from atomic actions (adopting the linear side information assumption);
(2) schematically generating a probability distribution $\mathcal{D}_{\widehat{R}}$ over c-action space $\mathcal{C}$, biased "towards" $\widehat{R}$; and
(3) sampling (up to) $k(N_e)$ c-actions $\mathcal{C}$ from $\mathcal{D}_{\widehat{R}}$.

EVALUATE then implements the recent *SequentialHalving* algorithm of Karnin et al. [12] for action recommendation (aka online planning) in regular MABs. Any other algorithm for this problem will do as well, but *SequentialHalving* provides the best formal guarantees to date, and it is the algorithm we have used in our empirical evaluation discussed later on.

Steps (1) and (2) of GENERATE are formulated above at high level, and there is a number of ways one can implement these steps. Considering step (1), if μ is indeed linear in atomic actions and $\mathcal{C}$ comprises all the possible combinations of atomic actions, then one can simply (i) pick an arbitrary set of $|A|$ c-actions that span the atomic actions A, (ii) use the average rewards obtained from sampling these actions equally often to construct a linear $|A| \times |A|$ system, (iii) solve this system to obtain the coefficients $w_{i;j}$ of μ, and (iv) skip the EVALUATE step, recommending the c-action that maximizes μ. However, both μ can be very much non-linear, and the constraint $\mathcal{C}$ can be arbitrary complex. Thus, the side information should be estimated and used in a way that relies on, yet is not constrained by, the side information assumption.

Given that, in SIDEINFO, both algorithms partition the sample allowance N_g equally between the atomic actions, and, for each atomic action $a_{i,j}$, set its weight $\widehat{R}(a_{i,j})$ to the average reward obtained from sampling some c-actions containing $a_{i,j}$. This is precisely the point where $\mathsf{LSI_V}$ and $\mathsf{LSI_F}$ slightly differ, and Figure 1b depicts the two corresponding versions of the EXTEND subroutine.

- In $\mathsf{LSI_V}$, all the m samples in $a_{i,j}$'s budget are dedicated to a single c-action, notably the c-action comprising only $a_{i,j}$. (In RTS, this corresponds to a c-action that activates unit i while leaving all other units idle.)
- In $\mathsf{LSI_F}$, the m samples in $a_{i,j}$'s budget go to some $\leq m$ c-actions containing $a_{i,j}$ that are generated uniformly at random.

In other words, $\mathsf{LSI_V}$ establishes weights $\widehat{R}$ by sampling all the $\sum_{i=1}^{n} k_i$ neighbors of a single vertex of the polytope induced by $\mathcal{A}$, and $\mathsf{LSI_F}$ establishes weights $\widehat{R}$ by sampling the α facets induced by the atomic actions A on that polytope. A priori, the relative attractiveness of $\mathsf{LSI_V}$ and $\mathsf{LSI_F}$ can be assessed only heuristically: The closer μ is to satisfy the assumption of linearity, the more advantageous $\mathsf{LSI_V}$ seems to be, and the other way around, the farther μ is from linearity, the more relatively reliable is the side information provided by $\mathsf{LSI_F}$.

Proceeding now with step (2) of GENERATE, i.e., using the weight function $\widehat{R}$ to fix a probability distribution $\mathcal{D}_{\widehat{R}}$ over $\mathcal{C}$, in Figure 1c we show two specific realizations of this step, GENERATE-ENTROPY and GENERATE-UNION. Both these realizations are motivated by the fact that $\mathcal{C}$ can comprise an arbitrary subset of the entire cross-product of the atomic action classes, and in both, $\mathcal{D}_{\widehat{R}}$ is specified implicitly, via auxiliary distributions over subsets of atomic actions, and step (2) is effectively combined with step (3) of sampling $k(N_e)$ c-actions $\mathcal{C}$ from $\mathcal{D}_{\widehat{R}}$.

- In GENERATE-ENTROPY, the atomic action classes are ordered in the increasing order of entropy that is exhibited by the corresponding probability distributions $\mathcal{D}[\{\widehat{R}(a_{i;1}), \ldots, \widehat{R}(a_{i;k_i})\}]$, as measured by an entropy measure H (such as the Shannon entropy, or some other Renyi entropy [8]). These measures quantify the diversity of probability distributions, and minimize on the least diverse distributions, which are uniform distributions. Hence, if c-actions are generated by sampling the atomic action classes sequentially, yet these sequential choices are interconstrained, sampling the action classes in the increasing order of $H(\mathcal{D}[\{\widehat{R}(a_{i;1}), \ldots, \widehat{R}(a_{i;k_i})\}])$ prioritizes classes in which the different atomic actions actually differ in their purported value, and thus the choice really matters.
- In GENERATE-UNION, the action classes are not sampled independently, but each c-action added to $\mathcal{C}$ is generated by sampling the union of all the atomic actions according to $\mathcal{D}[\widehat{R} \restriction_A]$, iteratively updating the conditional A to contain only actions from classes that are yet to be represented in the constructed c-action candidate.

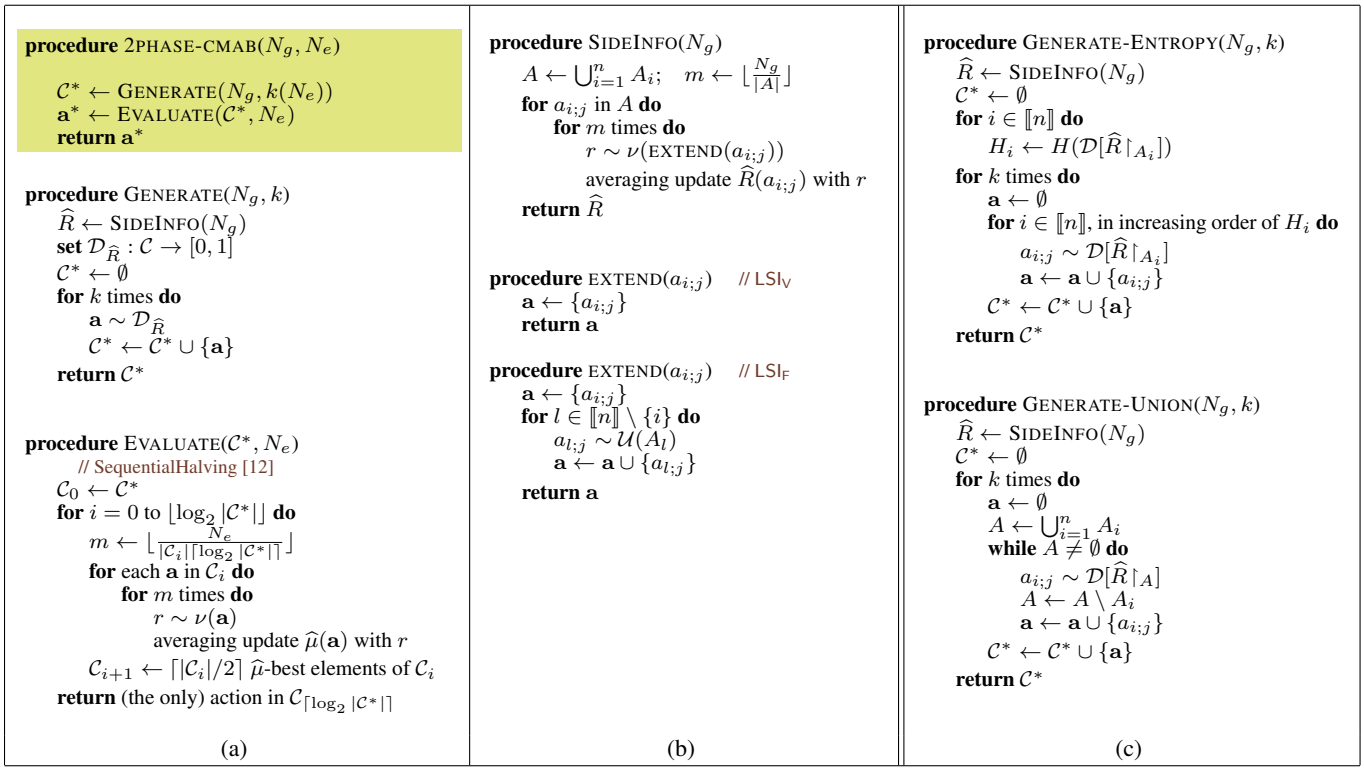


Figure 1. (a) The general 2PHASE-CMAB planning scheme, as well as the LSI scheme for candidate generation and evaluation, (b) specifics of the LSI$_V$ and LSI$_F$ instances of 2PHASE-CMAB, and (c) two specific procedures for LSI candidate generation.

4 TWO-PHASE CMAB MEETS RTS

In what follows, we report on an empirical evaluation of 2PHASE-CMAB on top of the μRTS game platform of Ontañón [16].[2] Our objective in this evaluation was

- to examine the relative effectiveness of 2PHASE-CMAB in general, and of LSI$_V$ and LSI$_F$ in particular,
- to examine the marginal contribution of the two-phases of 2PHASE-CMAB, and
- to examine the relevance of CMAB planning to multi-state sequential planning problems with combinatorial actions, such as RTS games.

The μRTS platform already contained an implementation of Naive Monte-Carlo (NMC), with parameters optimized as in [16], and we have added an implementation of LSI$_V$ and LSI$_F$; henceforth, superscripts e and u denote the versions of these algorithms using GENERATE-ENTROPY and GENERATE-UNION, respectively.

4.1 μRTS

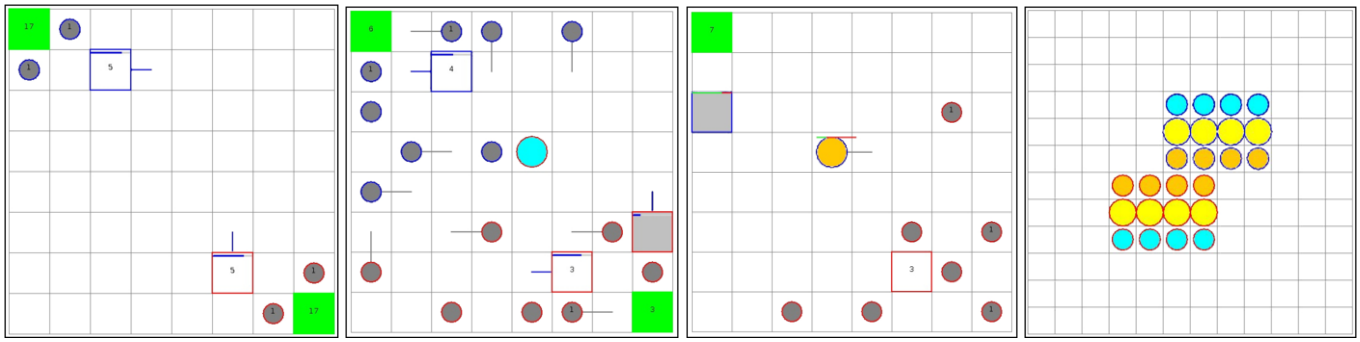
μRTS is a two-player zero-sum game that exhibits standard features of popular RTS games, and in particular, heterogeneous units with durative actions that can be activated concurrently. In our experiments we use the 8×8 grid environment. The environment is fully observable, and each grid cell can be occupied either by a single unit, or by a building, or by a resource storage. The storages each have a limited supply of the resource, and they can be used by both players. Table 1 shows the parameters of units and buildings. A player can build working units and combat units. The working units are all identical (*Worker*), and they can move the resources around, build buildings, and attack other units. As attackers, however, they are weak. The combat units come in three types—light melee (*LMelee*), heavy melee (*HMelee*), and ranged unit (*Ranged*)—all better attackers than working units, each with its own strengths and weaknesses. In general, movements and attacks are possible only within the 4-neighbourhood of the unit; all actions are durative and not interruptible. Finally, working units and combat units can be built only in *Base* and *Barracks* buildings, respectively.

	HP	*Cost*	*T(Move)*	*Damage*	*Range*	*T(Prod)*
Base	10	10	—	—	—	250
Barracks	4	5	—	—	—	200
Worker	1	1	10	1	1	50
LMelee	4	2	8	2	1	80
HMelee	4	2	12	4	1	120
Ranged	1	2	12	2	3	100

Table 1. Parameters of different buildings and units in μRTS. *HP* stands for health points, *Cost* is in the resource units, *T(Move)* is the duration of a single move (in simulated time units), *Damage* and *Range* represent decrease of *HP* of the target unit and the range of the attack, respectively, and *T(Prod)* is the duration of producing the unit/building.

For each player, the initial state of the game contains one *Base*, one *Worker* near the base, and one nearby resource storage, which, even if the resources are gathered optimally, suffices only for $1/3$ of the maximal game duration. The game is restricted to 3000 simulated

[2] We would like to thank Santiago Ontañón for making the μRTS platform available to the public.



(a) (b) (c) (d)

Figure 2. Three typical phases of the game in terms of unit counts: (a) early, (b) mid, and (c) end phases. Picture (d) depicts a "face-up" complex decision scenario: top to down, the rows of units are *Ranged*, *HMelee*, and *LMelee* of one player, and then, in reverse order, the units of the other player.

time units, and typically, it evolves in three phases in terms of the number of controlled units (and thus, the number of alternative c-actions): (i) early game (less units), (ii) mid game (more units) and (iii) end game (less units). Figures 2 depict representative states from these three phases.

4.2 Experiments

In our experiments, we compared the performance of LSI_V and LSI_F to that of NMC. In μRTS settings, the latter was already shown to substantially outperform both state-of-the-art tree search algorithms, such as ABCD, ε-greedy, and UCT, as well as regular MAB algorithms and some handcrafted heuristics [16]. As a baseline, we also added two basic algorithms that were already implemented in μRTS, namely

Random, selecting a random action for each agent as soon as it can act, and

LRush, a handcrafted heuristic policy, corresponding to, first, optimally gathers resources with one of the workers and building a *Barracks* as soon as it becomes possible, and after that, building only *LMelee* units which go towards and attack the closest enemy units and buildings.

Considering 2PHASE-CMAB, we have implemented LSI_V and LSI_F with both GENERATE-ENTROPY and GENERATE-UNION, resulting in four 2PHASE-CMAB algorithms: $\mathsf{LSI}^e_\mathsf{V}$, $\mathsf{LSI}^u_\mathsf{V}$, $\mathsf{LSI}^e_\mathsf{F}$, and $\mathsf{LSI}^u_\mathsf{F}$. In line with the ε_0 parameter of NMC being preset to 0.25, we have set the N_g and N_e parameters of 2PHASE-CMAB to $0.25N$ and $0.75N$, respectively. The number of candidates $k(N_e)$ was set so that the first iteration of SequentialHalving will sample each candidate at least once. The H-measure in GENERATE-ENTROPY was set to the Shannon entropy.

Likewise, to assess the marginal value of exploiting side information, we have also implemented a simplified instance, noSI, that selects $k(N_e)$ c-action candidates uniformly at random, and then passes these candidates to EVALUATE that implements SequentialHalving like LSI_V and LSI_F. In other words, noSI is a purified version of 2PHASE-CMAB that relies on no side information whatsoever. Importantly, to allow a meaningful comparison, the number of candidates $k(N_e)$ in noSI was set exactly as in LSI_V/LSI_F variants, yet the trivial GENERATE phase of noSI then uses only $k(N_e) \ll N_g$ samples, throwing out the residual $N_g - k(N_e)$ samples.

To reduce as much as possible the effect of the variance, as well as of possible biases of the game simulator, each algorithm played against each other algorithm 600 games, 300 games as "player 1" and 300 games as "player 2". For all the CMAB algorithms (including NMC),

- the overall computational effort was set to $N = 2000$ samples per decision;
- each sample comprised a simulated rollout of 200 game time units, with actions along the rollouts being selected at random, with a bias towards towards attacking a unit in reach, if there is such; and
- for rollouts ending at non-terminal states, the reward was assessed by a build-in evaluation function, reflecting the number of units and their health at the rollout's end-state.

This lookahead and evaluation procedure is similar to the one used in the original evaluation of NMC [16].

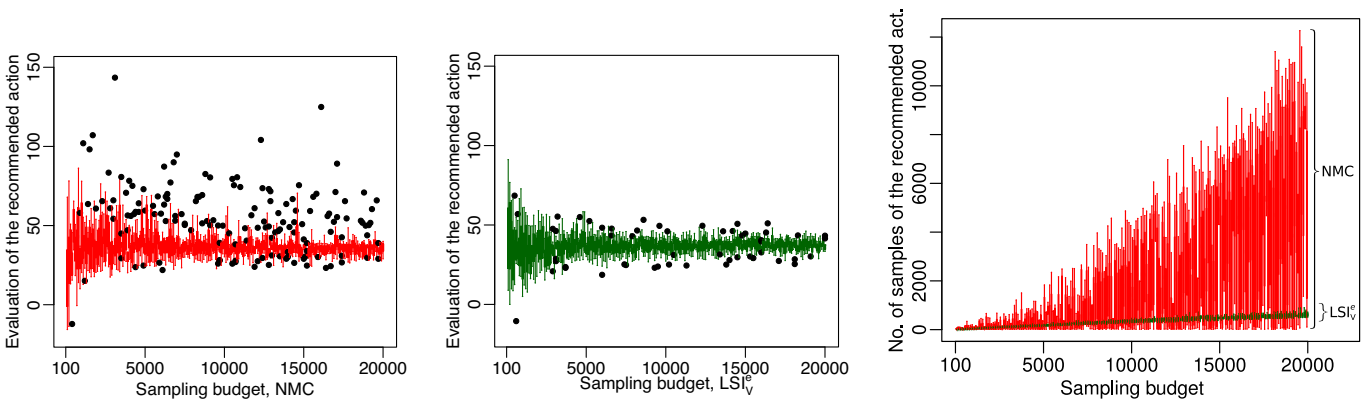
Table 2 shows the results of this head-to-head competition between the algorithms.

- Consistently with the previous experiments of Ontañón [16], all the CMAB algorithms easily defeated both Random and LRush, with the LSI instances never losing to these two baselines. The latter is not so for NMC, but its outperforming of Random and LRush is also a clear cut.
- All the four LSI instances of 2PHASE-CMAB consistently outperformed noSI. At the same time, the performance of the LSI instances among themselves was rather on par. In sum, this performance of LSI vs. noSI testifies both for the usefulness of linear side information (at least in this specific benchmark), as well as for the ability of the four GENERATE procedures of LSI instances to home in on this side information.
- All the five 2PHASE-CMAB instances, *including* noSI, substantially outperformed NMC. These results, and especially the result for noSI vs. NMC, strongly testify for the importance of a systematic candidate evaluation, and a controlled choice of the number of candidates to evaluate.

The latter point is even stronger supported by the results of an additional experiment that we performed on a complex decision scenario with 12 combat units at each side (see Figure 2d), and 5184 applicable c-actions per player. Figures 3a and 3b show that the empirical mean and variance of the value that NMC and $\mathsf{LSI}^e_\mathsf{V}$ estimated for the c-actions they ended up recommending were rather similar, and this consistently over different sizes of sample budget. At the same time, the number and the magnitude of the outliers, especially of those overestimating the evaluation, were substantially larger with NMC. These overestimates are critical in strategic scenarios as the

w/t/l →	Random	LRush	NMC	noSI	$\mathsf{LSI}^e_\mathsf{V}$	$\mathsf{LSI}^u_\mathsf{V}$	$\mathsf{LSI}^e_\mathsf{F}$	$\mathsf{LSI}^u_\mathsf{F}$
Random	38/27/35	94/0/6	100/0/0	100/0/0	100/0/0	100/0/0	100/0/0	100/0/0
LRush		0/100/0	96/0/4	98/0/2	100/0/0	100/0/0	100/0/0	100/0/0
NMC			41/13/46	52/12/36	54/14/32	55/14/31	54/15/31	52/15/32
noSI				42/17/41	46/17/40	47/14/38	44/17/39	46/17/37
$\mathsf{LSI}^e_\mathsf{V}$					40/18/42	42/19/40	42/16/42	43/18/39
$\mathsf{LSI}^u_\mathsf{V}$						41/17/42	39/16/44	45/15/40
$\mathsf{LSI}^e_\mathsf{F}$							43/17/40	41/17/42
$\mathsf{LSI}^u_\mathsf{F}$								41/16/43

Table 2. The results of the head-to-head competition: the percentage of wins/ties/looses of the column algorithm against the row algorithm.



(a) (b) (c)

Figure 3. Empirical mean and variance of the estimated value of the c-action recommended by NMC (a) and $\mathsf{LSI}^e_\mathsf{V}$ (b) in a complex "face-up" decision scenario (Figure 2d), after different sampling budgets (x-axis), ten runs per sample budget. The dots show the outliers that deviate three standard deviation from the mean. Respectively, (c) depicts the variance in the number of samples dedicated to the recommended c-action by NMC and $\mathsf{LSI}^e_\mathsf{V}$.

"face-up" scenario used in the experiments, because making a particularly bad decision here can determine loosing the entire game. An example of a such decision in μRTS is whether to build *Barracks* or not at an early stage of the game. As it is illustrated in Figure 3c, these drastically overestimating outliers are caused by the extreme variance in the number of samples used by NMC to estimate the value of the c-action that ends up being recommended, with quite often the recommendation being based on just a single estimating sample of the respective c-action. On the contrary, the variance in the number of samples used to estimate the value of the recommended c-action in the systematic candidate evaluation procedure of LSI_V is almost negligible, making the c-action selection process much more robust.

ACKNOWLEDGMENTS

This work was partly supported by USAF EOARD (grant no. FA8655-12-1-2096), the Technion- Microsoft Electronic-Commerce Research Center, and a Technion fellowship.

References

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer, 'Finite-time analysis of the multiarmed bandit problem', *Machine Learning*, **47**(2-3), 235–256, (2002).
[2] R. Balla and A. Fern, 'UCT for tactical assault planning in real-time strategy games', in *IJCAI*, pp. 40–45, (2009).
[3] S. Bubeck and R. Munos, 'Open loop optimistic planning', in *COLT*, pp. 477–489, (2010).
[4] S. Bubeck, R. Munos, and G. Stoltz, 'Pure exploration in finitely-armed and continuous-armed bandits', *Theor. Comput. Sci.*, **412**(19), 1832–1852, (2011).
[5] W. Chen, Y. Wang, and Y. Yuan, 'Combinatorial multi-armed bandit: General framework and applications', in *ICML*, pp. 151–159, (2013).
[6] M. Chung, M. Buro, and J. Schaeffer, 'Monte Carlo planning in RTS games', in *IEEE-CIG*, (2005).
[7] D. Churchill, A. Saffidine, and M. Buro, 'Fast heuristic search for RTS game combat scenarios', in *AIIDE*, (2012).
[8] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 2 edn., 2006.
[9] Z. Feldman and C. Domshlak, 'Monte-Carlo planning: Theoretically fast convergence meets practical efficiency', in *UAI*, (2013).
[10] Z. Feldman and C. Domshlak, 'On MABs and separation of concerns in Monte-Carlo planning for MDPs', in *ICAPS*, (2014).
[11] Y. Gai, B. Krishnamachari, and R. Jain, 'Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation', in *IEEE Symposium on New Frontiers in Dynamic Spectrum*, pp. 1–9, (2010).
[12] Z. S. Karnin, T. Koren, and O. Somekh, 'Almost optimal exploration in multi-armed bandits', in *ICML*, pp. 1238–1246, (2013).
[13] T. Keller and M. Helmert, 'Trial-based heuristic tree search for finite horizon MDPs', in *ICAPS*, pp. 135–143, (2013).
[14] L. Kocsis and C. Szepesvári, 'Bandit based Monte-Carlo planning', in *ECML*, pp. 282–293, (2006).
[15] A. Kovarsky and M. Buro, 'Heuristic search applied to abstract combat games', in *Canadian Conference on AI*, volume 3501 of *LNCS*, pp. 66–78, (2005).
[16] S. Ontañón, 'The combinatorial multi-armed bandit problem and its application to real-time strategy games', in *AIIDE*, (2013).
[17] A. Saffidine, H. Finnsson, and M. Buro, 'Alpha-beta pruning for games with simultaneous moves', in *AAAI*, (2012).
[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.