Executive Summary of the Thesis

A Combinatorial Multi-Armed Bandit algorithm for dollar volume maximization in the dark pool problem

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Stefano Martino

Advisor: Prof. Francesco Trovò

Co-advisors: Martino Bernasconi de Luca, Edoardo Vittori

Academic year: 2020-2021

Abstract

We study the problem of developing a Smart Order Routing algorithm able to maximize the dollar volume, which is the total value of the traded shares gained from selling assets of shares in a multi-venue environment consisting of dark pool venues. Our paper proposes a novel algorithm, namely the DP-CMAB algorithm, that extends the existing solutions by allowing the agent to specify the desired unit selling price to capture the optimal dollar volume from the trading venues. Moreover, we evaluate the DP-CMAB performance in an environment built from real market data and we show that our algorithm is able to outperform state-of-the-art algorithms.

1. Introduction

Smart Order Routing (SOR) refers to a class of automated algorithms used in online trading that aim at finding the best way to execute a trade. Our work extends the basic SOR problem, as it considers allocations across dark pools venues. Dark pools are a type of equity trading platform, characterized by their complete lack of transparency. The Dark Pool Smart Order Routing problem (DPSOR) consists of a sequential decision problem in which, at each time step t, an agent is given a certain volume of V^t units of shares and has to consume as many of those units by allocating them across K dark pools. Dark pools' complete lack of transparency is the main characterizing feature of our problem, due to the *censoring* aspect intrinsic to this type of trades. Indeed, if v shares are allocated to a dark pool and all of them are executed, the investor only learns that *at least* v units were available in the venue but not what would have been the maximum amount that could have been executed. We propose a novel online learning algorithm that frames the problem as a *Combinatorial Multi-Armed Bandit* (CMAB) problem and that handles the censored feedback. Our approach extends [1, 4] to a more general setting in which the agent can choose among a set of available selling prices. This allows the agent to submit an *intra*-venue routing, where more allocations to the same financial venue at different prices are possible.

2. Related Works

In [4], the authors consider an agent that receives a sequence of volumes $V^1, \dots, V^T$, where

$V^t \in [V]$[1], and has to allocate the available units by choosing among K dark pools. Each venue is characterized by a maximum consumption level s_i^t, which represents the available liquidity, which is assumed to be drawn from a fixed, yet unknown, distribution P_i. *Theorem 3* in [4] states that the allocations made by their algorithm are ϵ-suboptimal with probability at most $1-\epsilon$ after seeing a sufficient number of samples. The work of [1] extends [4] to an adversarial scenario, with improvements in the i.i.d. setup as well. The authors assume that the sequence of volumes and available liquidities are chosen by an adversary, i.e., V^t and s_i^t may depend on $\{v_i^1, \dots, v_i^{t-1}\}_{i=1}^K$. The authors propose an exponentiated gradient style algorithm that, as stated by their *Theorem 1*, achieves a regret of $O(V\sqrt{TlnK})$. Both [1, 4] evaluate a problem setting in which the agent is only interested in consuming as many available units as possible. In a realistic scenario, an agent also wants to maximize the dollar volume gained from the allocations, and thus, needs to select among different possible selling prices. We extend [1, 4] by proposing an algorithm that allows the player to choose the desired ask price.

3. Problem Formulation

The DPSOR can be modeled as an online decision problem in which, at each discrete time step $t \in [T]$ over a time horizon $T \in \mathbb{N}$, an agent is provided with a volume $V^t \in [V]$ of units of an asset, where V^t is sampled from an unknown distribution. Given $K \in \mathbb{N}$ dark pool venues and a vector $\mathcal{P} \in \mathbb{R}^{+N}$ of increasingly ordered prices, $p_i < p_j$ for $i < j$, the agent needs to devise a routing strategy that simultaneously specifies how to distribute the given volume across the venues and the desired selling price of each allocated share. Specifically, the learner must produce an allocation $\mathbf{A}^t \in \mathbb{N}^K \times \mathbb{N}^N$ of these units, whose generic element A_{kn}^t is such that $0 \leq A_{kn}^t \leq V^t$, for each $k \in [K]$, $n \in [N]$, $t \in [T]$, and represents the quantity allocated by the agent at round t to the *k-th* dark pool at price p_n. The available units are treated as perishable goods, meaning that all the shares not allocated at round t will not be available anymore during the following time step. Thus, for each $t \in [T]$, the agent's allocation $\mathbf{A}^t$ has to satisfy the following constraint, which formalizes the allocation of the entire volume V^t:

$$\sum_{k=1}^{K}\sum_{n=1}^{N} A_{kn}^t = V^t.$$

Subsequently, the agent receives a feedback from the environment, consisting of the number of units r_{kn}^t consumed at round t by the *k-th* dark pool at price p_n. Here $r_{kn}^t = min\{A_{kn}^t, s_{kn}^t\}$, where s_{kn}^t represents the actual liquidity present at time t in the *k-th* dark pool at the *n-th* price. If $r_{kn}^t = A_{kn}^t$, we denote the feedback as a *censored observation*, because the information the agent gathers is that $r_{kn}^t \leq s_{kn}^t$. Otherwise, if $r_{kn}^t < A_{kn}^t$, we say that the agent has received a *direct observation*, since it must be the case that $r_{kn}^t = s_{kn}^t$. The goal of the agent is to find a strategy that provides an order routing at each time step t that maximizes the captured dollar volume[2], which is the stock's share price multiplied by its volume:

$$R_t(\mathbf{A}) = \sum_{k=1}^{K}\sum_{n=1}^{N} r_{kn}^t \cdot p_n.$$

Since the routing of the whole volume may involve multiple allocations across different dark pools and at different prices, the problem can be formulated using the Combinatorial Multi-Armed Bandit framework introduced by [3]. More specifically, a generic arm (k, n, v), for $k \in [K]$, $n \in [N]$, and $v \in [V]$, represents an allocation of v units to the *k-th* dark pool venue at price p_n. From now on, we will use k_i, p_{n_i}, and v_i to refer to the dark pool, price and quantity denoted by arm $i \in [m]$ respectively. At each round t, the agent needs to choose a super arm S_t subject to a constraint $\mathcal{S} \subseteq 2^{[m]}$, formally defined as follows:

Definition 1. ***DPSOR valid super-arm*** *A valid super-arm S_t for the DPSOR problem at time t is defined as:*

$$S_t = \left\{(k_i, n_i, v_i)_{1 \leq i \leq |S_t|}\right\}, \; s.t. :$$
$$\sum_{i \in S} v_i = V^t,$$
$$\forall a = (k, n, v), a' = (k, n, q), a \in S \implies a' \notin S$$

[1]We use notation [V] to indicate the set {1,...,V}

[2]https://www.investopedia.com/terms/d/dollar-volume-liquidity.asp

Each arm $i \in [m]$, is associated to a random variable X_i, which is distributed as a Bernoulli random variable with unknown parameter μ_i and describes the probability that the allocation specified by arm i is absorbed by the environment. Therefore, the expectation vector $\boldsymbol{\mu} = (\mu_1, \dots, \mu_m)$, contains the estimates of the success probabilities of all the arms. The arm i allocated by the agent results in a sale of $r_{k_i,n_i} = min\{v_i, s_{k_i,n_i}\}$ units, where s_{k_i,n_i} is the actual liquidity present in dark pool k_i at price p_{n_i}. The reward of playing a super-arm S is:

$$R_t(S) = \sum_{i \in S} r_{k_i n_i} \cdot p_{n_i},$$

which corresponds to the dollar volume made by selling the allocated units. Let us define $r_{\boldsymbol{\mu}}(S) = \mathbb{E}[R_t(S)]$ and $r^*_{\boldsymbol{\mu}} = max_{S \in \mathcal{S}} r_{\boldsymbol{\mu}}(S)$. Based on the allocations executed by playing super-arm S_t, for each allocation $i \in S_t$, the agent observes the arm payoff $x_i = \mathbb{1}[r_{k_i n_i} = v_i]$, which is a realization of the Bernoulli random variable X_i. We say that arm i is *successful* if dark pool k_i manages to absorb v_i units at price p_{n_i} ($x_i = 1$), otherwise we denote it as a *failure* ($x_i = 0$). We make the following assumption on the problem setting:

Assumption 1. *If the allocation (k, n, v) was successful, then all the allocations $(k, \hat{n}, \hat{v})$, for $0 \le \hat{v} \le v$ and $\hat{n} \in [N]$ such that $p_{\hat{n}} \le p_n$ are successful. Likewise, if the allocation (k, n, v) was a failure, then every allocation $(k, \tilde{n}, \tilde{v})$, for $\tilde{v} \ge v$ and $\tilde{n} \in [N]$ such that $p_{\tilde{n}} \ge p_n$ are failures.*

As a consequence, when a super-arm S_t is played, not only the outcomes of the arms contained in S_t are revealed, but also other arms (allocations) might reveal their payoffs, and the reward the agent receives depends on the outcomes of these arms as well. In fact, suppose that the allocation (k_i, n_i, v_i) is successful ($x_i = 1$). Then, as stated by Assumption 1, we also infer all the payoffs corresponding to the arms $(k, \hat{n}, \hat{v})$, for $0 \le \hat{v} \le v_i$ and $p_{\hat{n}}$ such that $p_{\hat{n}} \le p_{n_i}$. The complementary case, in which an allocation is a failure ($x_i = 0$), applies as well.
Now we proceed to check that our formulation satisfies the assumptions required by the CMAB framework of [3], namely the monotonicity property and the existence of a bounded smoothness function $f(\cdot)$. Given two super arms $S \in \mathcal{S}$ and $S' \in \mathcal{S}$, such that for all $i \in [m]$, $\mu_i \le \mu'_i$, the expected dollar volume made by playing S is not larger than the expected dollar volume obtained by playing S'. More specifically, if for all $i \in [m]$, $\mu_i \le \mu'_i$, $r_{\boldsymbol{\mu}}(S) \le r_{\boldsymbol{\mu}'}(S)$ for all $S \in \mathcal{S}$. Thus, the monotonicity constraint of [3] is satisfied. Finally, the bounded smoothness function is $f(x) = V \cdot p_{max} \cdot x$, i.e., increasing all probabilities of arms by x can increase the expected dollar volume by at most $V \cdot p_{max} \cdot x$, where p_{max} is the highest ask price. Suppose that, as stated by [3], we have a (γ, σ)-Approximation Oracle able to compute the sub-optimal super arms. We define the (γ, σ)-Approximation Pseudo-Regret for the DPSOR problem as follows:

Definition 2. ***((γ, σ)-Approximation Pseudo-Regret) for the DPSOR problem*** *The (γ, σ)-Approximation Pseudo-Regret for the DPSOR problem after N rounds of a given policy $\mathcal{U}$ that picks the super-arm S_t at round t is defined as:*

$$Reg_N(\mathcal{U}) = N \cdot \gamma \cdot \sigma \cdot r^*_{\boldsymbol{\mu}} - \mathbb{E}\left[\sum_{t=1}^{N} r_{\boldsymbol{\mu}}(S_t)\right].$$

Algorithm 1 DP_CMAB(dark pools K, prices $\mathcal{P}$, bound on volume V)

```
1: Initialize $\alpha^1_{knv} = \beta^1_{knv} = 1\ \forall k \in [K]$, $n \in [N]$, $v \in [V]$
2: for $t \in [T]$ do
3:     Let $V^t$ be the volume given to the agent
4:     Define $\theta^t_{k_i n_i v_i}$ according to Equation (2); let $\boldsymbol{\theta} = (\theta_1, ..., \theta_m)$
5:     $\mathbf{A}^t \leftarrow$ Allocation($V^t, K, \mathcal{P}, \boldsymbol{\theta}$)
6:     Submit the allocations according to $\mathbf{A}^t$
7:     Observe $r^t_{kn}$
8:     $\boldsymbol{\alpha}^{t+1}$, $\boldsymbol{\beta}^{t+1} \leftarrow$ Update($\boldsymbol{\alpha}^t$, $\boldsymbol{\beta}^t$, $r^t_{kn}$, $A^t_{kn}$)
9: end for
```

4. DP-CMAB Algorithm

We estimate the vector of expectations of all arms, $\boldsymbol{\mu} = (\mu_1, ..., \mu_m)$, using a Bayesian approach. More specifically, we consider the random variable X_i, describing the random outcome of the generic arm i, as distributed according to a Bernoulli distribution with parameter μ_i. Therefore, we estimate the expectation probability μ_i of the arm (k_i, n_i, v_i) by keeping the quantities $\alpha^t_{k_i,n_i,v_i}$ and $\beta^t_{k_i,n_i,v_i}$, that indicate the number of successes and the number of failures of arm i up to round t respectively. A high-level pseudo-code of the DP-CMAB algorithm is provided in Algorithm 1. At the beginning of the algorithm, all entries of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are set to 1, which reduces the Beta distribution corresponding to each arm to a Uniform distribution. At each time step t, the agent is given a volume V^t to allocate. *Line 4* defines $\theta^t_{k,n,v}$, which represents the agent's current estimate of the probability that at least v units will be consumed by dark pool k at price p_n. Algorithm 1 can choose between three different strategies to compute $\theta^t_{k,n,v}$:

$$\theta^t_{knv} \sim Beta\left(\alpha^t_{knv}, \beta^t_{knv}\right), \tag{2a}$$

$$\theta^t_{knv} = \frac{\alpha^t_{knv}}{\alpha^t_{knv} + \beta^t_{knv}} + \sqrt{\frac{2\log(t)}{\alpha^t_{knv} + \beta^t_{knv} - 2}}, \tag{2b}$$

$$\theta^t_{knv} = Q_{Beta}\left(1 - \frac{1}{t(\log T)^5}\right), \tag{2c}$$

where Q_{beta} denotes the quantile of the Beta distribution. We refer to DP-CTS as the agent following the Thompson Sampling strategy (Equation (2a)), DP-CUCB as the agent implementing the UCB strategy (Equation (2b)) and DP-Bayes UCB as the agent using the Bayes UCB strategy (Equation (2c)). *Line 5* invokes the Allocation subroutine, that computes the allocation matrix defining the routing of the volume. As the agent receives feedbacks from the environment, it uses the gathered evidence to update its beliefs (*Line 8*).

Algorithm 2 Allocation(Volume V^t, dark pools K, prices $\mathcal{P}$, estimates $\boldsymbol{\theta}$)

```
1: Initialize $A^t_{kn} = 0 \; \forall k \in [K], n \in [N]$
2: for $u \in [V^t]$ do
3:     $k^*, n^* = \text{argmax}_{k,n}\{p_n \cdot \theta^t_{k,n,A^t_{kn}+1}\}$
4:     $A^t_{k^*n^*} \leftarrow A^t_{k^*n^*} + 1$
5: end for
6: return $\mathbf{A}^t$
```

Line 3 of Algorithm 2 implements the *Greedy maximization step*, responsible for the choice of the best allocations to submit based on the maximization of the expected dollar volume.
Suppose that the agent has already allocated $q \in [V^t - 1]$ units to the *k-th* dark pool at price p_n ($A^t_{kn} = q$). Then, $\theta^t_{k,n,A^t_{kn}+1}$ represents the agent's current estimate of the probability that an additional unit will be consumed by that dark pool and price pair. The quantity $p_n \cdot \theta^t_{k,n,A^t_{kn}+1}$, thus, represents the dollar volume the agent expects to make from allocating such additional unit to the $(k,\, n)$ pair of dark pool and price. For each unit $u \in [V^t]$ to allocate, the Greedy Maximization step finds the best pair (k^*, n^*) of dark pool and price that maximizes the expected dollar volume gained by adding the *u-th* unit to the allocation of the k^**-th* dark pool at price p_{n^*}. Note that Algorithm 2 performs the role of the oracle introduced by [3], in that it is responsible of building the super arms representing the allocations. The construction of the arms is achieved in a time polynomial with respect to the number of dark pools K, the number of prices N and the volume V.

4.1. Update

Each time the algorithm receives a feedback from the environment, it uses the observation to improve its estimates and make increasingly accurate predictions about the environment. The algorithm can choose among three different types of updates, namely the *no propagation* update, the *quantity propagation* update and the *full propagation* update. The no propagation update (Algorithm 3) does not exploit the existing correlations between the arms stated by Assumption 1 and decides whether to increment $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$ based on the fact that that allocation was successful (censored observation) or a failure (direct observation).

Algorithm 3 No_propagation(successes $\boldsymbol{\alpha}^t$, failures $\boldsymbol{\beta}^t$, units sold r^t_{kn}, units allocated A^t_{kn})

```
1: if $r^t_{kn} = A^t_{kn}$ then
2:     $\alpha^{t+1}_{k,n,A^t_{kn}} \leftarrow \alpha^t_{k,n,A^t_{kn}} + 1$
3: else
4:     $\beta^{t+1}_{k,n,A^t_{kn}} \leftarrow \beta^t_{k,n,A^t_{kn}} + 1$
5: end if
6: return $\boldsymbol{\alpha}^{t+1}, \boldsymbol{\beta}^{t+1}$
```

Instead, the quantity propagation update (Algorithm 4) uses Assumption 1 and propagates the feedback to non-played arms, exploiting the quantity correlations between the allocations.

Algorithm 4 Quantity_propagation(successes $\boldsymbol{\alpha}^t$, failures $\boldsymbol{\beta}^t$, units sold r^t_{kn}, units allocated A^t_{kn})

```
1: $\alpha^{t+1}_{k,n,\hat{v}} \leftarrow \alpha^t_{k,n,\hat{v}} + 1,\ 0 \leq \tilde{v} \leq r^t_{kn}$
2: if $r^t_{kn} \neq A^t_{kn}$ then
3:     $\beta^{t+1}_{k,n,\tilde{v}} \leftarrow \beta^t_{k,n,\tilde{v}} + 1,\ \forall \tilde{v} > r^t_{ij}$
4: end if
5: return $\boldsymbol{\alpha}^{t+1}, \boldsymbol{\beta}^{t+1}$
```

In the full propagation update (Algorithm 5), the agent uses Assumption 1 to propagate the feedback to non-played arms, exploiting both the quantity and price correlations between the allocations.

Algorithm 5 Full_propagation(successes $\boldsymbol{\alpha}^t$, failures $\boldsymbol{\beta}^t$, units sold r^t_{kn}, units allocated A^t_{kn})

```
1: $\alpha^{t+1}_{k,\hat{n},\hat{v}} \leftarrow \alpha^t_{k,\hat{n},\hat{v}} + 1,\ \forall\ 0 \leq \hat{v} \leq r^t_{kn},\ \hat{n} \in [N]$
   s.t. $p_{\hat{n}} \leq p_n$
2: if $r^t_{kn} \neq A^t_{kn}$ then
3:     $\beta^{t+1}_{k,\tilde{n},\tilde{v}} \leftarrow \beta^t_{k,\tilde{n},\tilde{v}} + 1,\ \forall \tilde{v} > r^t_{kn},\ \tilde{n} \in [N]$ s.t
       $p_{\tilde{n}} \geq p_n$
4: end if
5: return $\boldsymbol{\alpha}^{t+1}, \boldsymbol{\beta}^{t+1}$
```

4.2. Regret Analysis

The DP-CUCB agent implements a strategy equivalent to the one adopted by the CUCB algorithm introduced by [3]. Similarly, our DP-CTS agent uses a strategy matching the CTS algorithm introduced by [5]. Their algorithms exploit an oracle, that, given all the terms $(\theta_i)_{1 \leq i \leq m}$, computes the super arm S to play. Algorithm 2 can be seen as such an oracle, that, given $(\theta_i)_{1 \leq i \leq m}$ along with the knowledge of the problem instance, computes the super arm corresponding to the best allocation in terms of dollar volume. The applications of [3, 5] help us finding an upper bound for the expected regret of the DP-CUCB and DP-CTS agents without propagation:

Theorem 4.1. *The expected regret of the DP-CUCB and DP-CTS algorithms implementing the no propagation update for the DPSOR problem with a max volume V, K dark pools and N prices with a max price p_{max} after T rounds is at most:*

$$\mathbb{E}[\mathcal{R}(T)] \leq O\Big((V \cdot p_{max})^2 \cdot \ln(T) + VKN\Big)$$

5. Experiments

In this section we analyze the empirical performance of the DP-CMAB algorithms, comparing them to the baselines represented by the agents studied in [1, 4]. In our experiments, every agent plays R independent runs, each consisting of T time steps, in which the agent has to gain the best possible dollar volume from allocating V^t units, each consisting of 20000 shares of an asset across K dark pools. Since [1, 4] do not take into account the possibility to specify the price to sell the shares to, we devised two strategies their algorithms can implement:

- The *Random price selection* strategy selects a random price at the beginning of every run $r \in [R]$, and continues to selling the units to that fixed price for each $t \in [T]$ of that run.
- The *Oracle price selection* plays each run $r \in [R]$ and time step $t \in [T]$, by selling the units to the *oracle price* p^*, which is the best single price that maximizes the expected cumulative dollar volume across all runs R. In practice, p^* is the unique price the agent would select if it had perfect knowledge of the environment, *i.e.* if it could have access to the actual underlying liquidity present in the dark pools.

The experiments were made by setting the following parameters:

- Time steps $T = 1000$;
- Runs R = 20;
- Number of dark pools $K = 10$;
- Volume to be sold $V^t = 10$ for $t \in [T]$;
- $\mathcal{P} =< 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100 >$.

We generated the dark pools liquidity using real historical market data, extracted from the Apple (AAPL) order books messages of the NASDAQ exchange, generated by following the steps detailed by [2] on the data found here: `ftp://emi.nasdaq.com/ITCH/Nasdaq%20ITCH`. Since our agent's purpose is to sell the units it is provided with, we considered only the messages marked as *Executed* on the *Buy* side, consisting of all the accepted and completed buy orders on behalf of a client.

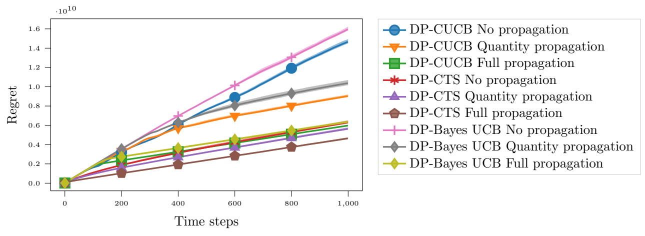

Figure 1: Regret of the DP-CMAB agents.

Figure 1 shows the mean and the 95% confidence intervals of the regret of the DP-CMAB agents. The full propagation update strategy achieves a better performance in terms of regret than the quantity propagation update, which, in turn, enjoys a lower regret than the agents not implementing any kind of propagation.

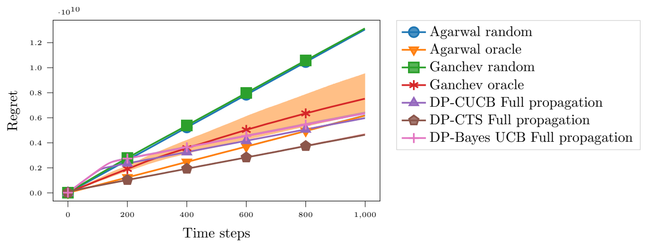

Figure 2: Regret of the DP-CMAB agents implementing the full propagation update and the baselines.

Figure 2 compares the regret of the DP-CMAB agents exploiting the full propagation update and the baseline agents implementing the random and oracle price selection strategies. Note that the confidence intervals of the baseline agents using the random price selection have been omitted for clarity reasons. The baseline agents implementing the random price selection achieve a very high regret, while the baseline agents implementing the oracle price selection strategy manage to perform better thanks to the a priori knowledge about the liquidity distribution.

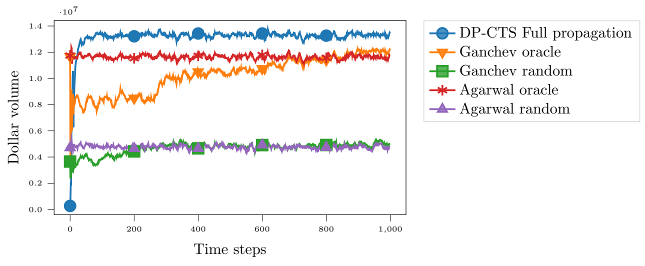

Figure 3: Average dollar volume of the DP-CTS full propagation agent against the baselines.

Figure 3 shows the average dollar volume made at each time step by the DP-CTS Full propagation agent compared to the baselines. Our agent manages learn an allocation policy that is better than the one produced by the baselines using the oracle strategy, without having access to any prior information about the environment.

6. Conclusions

The main focus of our work is to design a Smart Order Routing algorithm to maximize the trader's dollar volume gained from the allocation of given units of assets across dark pool venues. We accomplished this result by introducing the DP-CMAB algorithm which overcomes the limitations of the existing works in the dark pool literature. Indeed, [1, 4] consider a scenario in which an agent cannot specify the selling price of the shares. In our paper, we compared the empirical performance of our agents with the baselines of [1, 4], showing how the DP-CMAB agents can empirically outperform state-of-the-art algorithms by leveraging the knowledge of the problem setting. Further developments could involve the extension to a scenario in which the trader is not limited to only sell the shares, but has the possibility to buy assets from the dark pools as well.

References

[1] Alekh Agarwal, Peter Bartlett, and Max Dama. Optimal allocation strategies for the dark pool problem. PMLR, 13–15 May 2010.

[2] Martino Bernasconi-De-Luca, Luigi Fusco, and Ozrenka Dragić. martinobdl/itch: Itch50converter, 2021.

[3] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. 28(1):151–159, 17–19 Jun 2013.

[4] Kuzman Ganchev, Michael J. Kearns, Yuriy Nevmyvaka, and Jennifer Wortman Vaughan. Censored exploration and the dark pool problem. *CoRR*, abs/1205.2646, 2012.

[5] Siwei Wang and Wei Chen. Thompson sampling for combinatorial semi-bandits. PMLR, 2018.