Dark-Pool Smart Order Routing: a Combinatorial Multi-armed Bandit Approach

Martino Bernasconi
martino.bernasconi@polimi.it
Politecnico di Milano

Stefano Martino
stefano.martino@mail.polimi.it
Politecnico di Milano

Edoardo Vittori
edoardo.vittori@intesasanpaolo.com
Intesa Sanpaolo

Francesco Trovò
francesco1.trovo@polimi.it
Politecnico di Milano

Marcello Restelli
marcello.restelli@polimi.it
Politecnico di Milano

Abstract

We study the problem of developing a Smart Order Routing algorithm that learns how to optimize the dollar volume, i.e., the total value of the traded shares, gained from slicing an order across multiple dark pools. Our work is motivated by two distinct issues: (i) the surge in liquidity fragmentation caused by the rising popularity of electronic trading and by the increasing number of trading venues, and (ii) the growth in popularity of dark pools, an exchange venue characterised by a lack of transparency. This paper critically discusses the known dark pool literature and proposes a novel algorithm, namely the DP-CMAB algorithm, that extends existing solutions by allowing the agent to specify the desired limit price when placing orders. Specifically, we frame the problem of dollar volume optimization in a multi-venue setting as a Combinatorial Multi-Armed Bandit (CMAB) problem, representing a generalization of the well-studied MAB framework. Drawing from the rich MAB and CMAB literature, we present multiple strategies that our algorithm may adopt to select the best allocation options. Furthermore, we analyze how exploiting financial domain knowledge improves the agents' performance. Finally, we evaluate the DP-CMAB performance in an environment built from real market data and show that our algorithm outperforms state-of-the-art solutions.

CCS Concepts

• **Theory of computation → Online learning algorithms**; *Online learning theory*; *Sequential decision making*.

Keywords

Dark pools, smart order routing, combinatorial multi-armed bandit

ACM Reference Format:
Martino Bernasconi, Stefano Martino, Edoardo Vittori, Francesco Trovò, and Marcello Restelli. 2022. Dark-Pool Smart Order Routing: a Combinatorial Multi-armed Bandit Approach. In *3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3533271.3561728

1 Introduction

Traditional market structures have been the subject of major changes during the past decades, such as advances in technology and the introduction of new regulations, causing a surge in liquidity fragmentation. In this scenario, electronic trading platforms compete against each other on execution cost and speed, increasing the demand for intelligent order routing software, a.k.a. Smart Order Routing (SOR) [Cont and Kukanov 2017; Laruelle et al. 2011; Maglaras et al. 2012; Pujol and Brueckner 2009]. In particular, SOR refers to a class of algorithms that optimally split an order over multiple venues to minimize transaction fees and market impact, taking into account all the different allocation opportunities and placing orders based on the best available options. In this work, we address the problem of SOR across dark pool venues, namely the Dark Pool Smart Order Routing (DPSOR) problem.

Dark pools are equity trading venues characterized by a complete lack of transparency [Shorter and Miller 2014]. Differently from the so-called *lit* venues, this type of exchange does not reveal any information about the prices and volumes therein, allowing participants to perform trades of considerable amounts without advertising to the general public [Zhu 2013]. Dark pools emerged in the late 1980s and have since experienced rapid growth in popularity. In 2009, more than 40 dark pools were operative in the US alone, with a trading volume growing annually at an average rate of 40% [Degryse et al. 2009; Zhu 2013]. According to the data, the US market share of dark pools increased from about 7.51% in 2008 to 16.57% in 2015 [Ye 2016].

We define the DPSOR problem as a sequential decision problem in which, at each time step t, an agent, given a volume of shares V to execute, has to consume as many shares as possible by allocating them across K dark pools while maximizing the value of the traded assets. Dark pools' complete lack of transparency is the main characterizing feature of our problem due to the *censoring* aspect intrinsic to this type of trade. 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. On the other hand, submitting a really large amount of shares v that ensures that $\tilde{v} < v$ shares are executed ($\tilde{v}$ is the maximum volume that was available at that dark pool) gives an exact picture of the liquidity present, but $v - \tilde{v}$ shares are left unexecuted, which is a loss for the investor. The online learning literature tackled the problem of allocation in the dark pool setting, providing a partial solution to the problem, with the work by Agarwal et al. [2010];

Ganchev et al. [2010]. Although effective and suitable for the DP-SOR problem, the above approaches do not consider the possibility of specifying the desired execution price, thus considering only market orders and no limit orders.[1] However, incorporating this factor is crucial to develop a robust SOR algorithm whose foremost goal is to optimize the trade's dollar volume, which is the trade's execution price multiplied by its volume. Thus, our goal is to devise an algorithm that creates and maintains a dynamic estimate of the hidden liquidity present in each dark pool and uses this information to make optimal joint routing and pricing decisions.

Original Contribution We propose a novel online learning algorithm that frames the DPSOR problem as a *Combinatorial Multi-Armed Bandit* (CMAB) problem [Chen et al. 2013], that handles censored feedback and allows to specify the limit price at which to execute the order in the dark pool. The *Multi-Armed Bandit* (MAB) framework, originally formulated by [Auer et al. 2002], has already proven effective in dealing with financial settings, e.g., in pricing settings [Mussi et al. 2022; Trovò et al. 2018], and in online portfolio optimization [Bernasconi de Luca et al. 2021; Das et al. 2013; Vittori et al. 2020]. The extension of the classical MAB framework to the CMAB one is required by the need to specify the trade's volume as a combination of multiple allocations, each indicating the trading volume, the destination venue, and the desired limit price.

In this paper, we extend the problem formulation presented in the works of Agarwal et al. [2010]; Ganchev et al. [2010] to a more general setting in which the agent can choose among a set of available prices. Furthermore, we conceive a novel set of algorithms with regret guarantees, namely DP-CMAB, designed to solve the DPSOR problem. We also introduce specific learning algorithms exploiting the correlation existing among volumes and prices of the DPSOR problem. Finally, we demonstrate the performance of the proposed algorithms in an extensive experimental campaign on a realistic simulation framework, comparing them with state-of-the-art approaches.

2 Related Works

Order routing problems have been addressed by a stream of research that has gained popularity with the increase in liquidity fragmentation. In particular, works from the economic field try to understand the effects of such fragmentation, such as the one by Hendershott and Mendelson [2000], which analyses interactions between lit exchanges and dark pools, and the one by Foucault and Menkveld [2008] that studies the effects of these algorithms on the markets.

Among the first empirical approaches to order routing is the work by Almgren and Harts [2008], which proposes a heuristic rule to estimate hidden liquidity. It can be used both in the case of lit exchanges (iceberg orders) and in the case of dark pools. Nonetheless, it does not propose a methodology for optimally splitting the orders among different venues. On the other hand, Laruelle et al. [2011] approach the problem from a more rigorous perspective, framing it as an optimization problem with constraints and solving it with also with a reinforcement learning approach. Maglaras et al. [2012] concentrates on lit exchanges and derives a characterization of the market equilibrium. Instead, Cont and Kukanov [2017] propose a stochastic algorithm that computes the optimal routing policy in the case of lit exchanges. Notably, the above-mentioned works cannot be applied to order routing in dark pools where liquidity is completely hidden.

The order routing problem is closely connected to optimal execution, studied by Almgren and Chriss [2001]; Bertsimas and Lo [1998], among others. These works aim at minimizing transaction costs and market impact with temporal slicing but focus on a single venue. In general, optimal execution approaches use market orders, but there also exist extensions that enable choosing the price using limit orders, such as the work by Guéant et al. [2012]. The interested reader can refer to the book by [Guéant 2016] which proposes an in-depth analysis of optimal execution approaches. Finally, the work by Kratz and Schöneborn [2014] analyzes the hybrid problem of liquidating a portfolio using market orders on both a lit exchange and a dark pool. All these works assume perfect knowledge of the dark pool's underlying model and solve the problem analytically. Unfortunately, such approaches cannot be applied in real-life SOR due to the lack of information about the dark pools' liquidities.

Another way of modeling the SOR problem has been provided by the techniques of the online learning field. In particular, the starting point for our model formalization is the well-known *newsvendor problem* [Qin et al. 2011] from the operations research literature. In this setting, each day, the agent needs to choose a quantity of a good (the newspaper) to purchase at a fixed per-unit price and uncertain demand. The work by Huh and Rusmevichientong [2009] extends this formulation by introducing a non-parametric approach and censored demand, due to which, when the agent buys a small quantity of the good, she only observes the sold quantity without having access to the realized demand. However, this work limits its strategy to a *single* selling venue and, therefore, is not suitable for its application to the SOR setting in which one has to deal with a fragmented market.

The works of Ganchev et al. [2010] and Agarwal et al. [2010] are closely related to ours since they extend the framework presented by Huh and Rusmevichientong [2009] by taking into account a multi-venue environment. Specifically, Ganchev et al. [2010] design an algorithm that uses the Kaplan-Meier estimators to produce an estimate of the tail probabilities of the total liquidity available in each dark pool. Based on these estimates, their agent allocates the available units by greedily choosing the dark pools with the highest tail probabilities estimates and receives observations from the environment that are, in turn, used to update the tail probabilities. The work of Agarwal et al. [2010] is an extension of such a work to an adversarial scenario, which obtains improvements in the i.i.d. setup as well. Our work extends the mentioned approaches by allowing the agent to choose the desired price in addition to the destination venue with the objective of optimizing the dollar volume from the trade. As mentioned before, the above works are not taking into account the possibility of specifying the asset price, and therefore their explicit goal is to consume as many available units as possible, while no optimization in terms of dollar value is adopted.

[1]Limit orders are common in dark pools, see, for instance, JPM-X at www.jpmorgan.com/content/dam/jpm/cib/complex/content/markets/aqua/pdf-0.pdf.

3 Problem Formulation

In this section, we formulate the DPSOR problem. At each discrete round $t \in [T]$, over a time horizon $T \in \mathbb{N}$, an agent (also referred to as learner) distributes a total volume $V \in \mathbb{N}$ of units of an asset among $K \in \mathbb{N}$ different dark pool venues.[2] For each unit of the asset assigned to a dark pool, we need to specify a price among a set of ordered prices $\mathcal{P} = \{p_1, \dots, p_N\}$, with $N \in \mathbb{N}$ and s.t. $p_{i-1} < p_i$, i.e., to distribute the given volume across the venues and the desired price for each unit.[3,4]

Formally, the learner has to select an allocation matrix $\mathbf{A}^t \in \mathbb{N}^{K \times N}$, whose generic element A^t_{kn} represents the quantity it allocates at round t to the k-th dark pool at price p_n. Note that, in the considered scenario, the available asset units are treated as perishable goods, meaning that all the shares not allocated at the current round are not available anymore at the following one.

Subsequently, the agent receives feedback from the environment, consisting of the number of units r^t_{kn} consumed at round t by the k-th dark pool at price p_n. Here, $r^t_{kn} = \min\{A^t_{kn}, s^t_{kn}\}$, where s^t_{kn} represents the actual liquidity present at time t in the k-th dark pool at price p_n. Indeed, if $r^t_{kn} = A^t_{kn}$, we denote the feedback as a *censored observation*, because the agent only gathers the information that $r^t_{kn} \leq s^t_{kn}$. Otherwise, if $r^t_{kn} < A^t_{kn}$, we say that the agent has received a *direct observation*, since it must be the case that $r^t_{kn} = s^t_{kn}$.

The goal of the agent is to find an algorithm $\mathfrak{U}$ providing, at each round $t \in [T]$, an allocation $\mathbf{A}^t$ that maximizes the dollar volume of the asset, defined as:

$$R_t(\mathfrak{U}) = \sum_{k=1}^{K} \sum_{n=1}^{N} r^t_{kn}\, p_n. \tag{1}$$

3.1 Formalizing DPSOR as a CMAB

A generic CMAB [Chen et al. 2013] consists in a tuple:

$$(\mathcal{M}, \mathcal{S}, \boldsymbol{\mu}, f_{\boldsymbol{\mu}}(\cdot), \mathrm{Opt}(\cdot)), \tag{2}$$

where $\mathcal{M}$ is a set of arms, $\mathcal{S} \subset 2^{\mathcal{M}}$ is the set of the feasible set of arms (a.k.a. superarms) the learner is allowed to choose at each round, $\boldsymbol{\mu}$ is associating a value μ_i of including arm in $i \in \mathcal{M}$ in the superarm, $f_{\boldsymbol{\mu}} : \mathcal{S} \rightarrow \mathbb{R}^+$ is a function providing the reward associated to a feasible superarm, and $\mathrm{Opt}(V, \boldsymbol{\mu})$ is an oracle providing the arm $S_{\boldsymbol{\mu}} \in \mathcal{S}$ that maximizes the function $f_{\boldsymbol{\mu}}(\cdot)$. We denote with r^* the expected dollar value provided by the optimal superarm, formally $r^* := f_{\boldsymbol{\mu}}(\mathrm{Opt}(\mathrm{V}, \boldsymbol{\mu}))$.

In the DPSOR case, the available arms corresponds to the elements of the matrix A^t and the set of the feasible superarms $\mathcal{S}$ is composed by all the matrices A^t satisfying the following constraint:

$$\sum_{k=1}^{K} \sum_{n=1}^{N} A^t_{kn} = V, \tag{3}$$

i.e., requiring the agent to allocate all the available units. We remark that this formulation, differently from the ones by [Agarwal et al. 2010; Ganchev et al. 2010], allows *intra-venue* routing, where multiple allocations to the same financial venue at different prices are possible.

[2]For a generic number $T \in \mathbb{N}$, we denote the set $\{1, \dots, T\}$ using the symbol $[T]$.
[3]Note that the scenario considered by Agarwal et al. [2010]; Ganchev et al. [2010] is a special case of ours in which you can execute all the dark pools at a single price, the same for all the venues.
[4]For the sake of presentation, we assume all the dark pools allow to set prices in $\mathcal{P}$. The extension to a setting with different price sets is straightforward.

Algorithm 1 DP-CMAB

```
1:  Parameters: number of dark pools $K$, prices set $\mathcal{P}$, volume $V$
2:  Initialize $\alpha^t_{knv} = \beta^t_{knv} = 1 \; \forall k \in [K], n \in [N], v \in [V]$
3:  for $t \in [T]$ do
4:      Let $V$ be the volume given to the agent
5:      for arms $i \in \mathcal{M}$ do
6:          Define $\theta^t_{knv}$ according to Equation (6), (7), or (8)
7:      $A^t \leftarrow \mathrm{Opt}(V, \boldsymbol{\theta})$
8:      Play allocation $A^t$
9:      Receive feedbacks $r^t_{knv}$ on the arms selected in the allocation $A^t$
10:     Update the parameters of the distributions $\alpha^t_{knv}$ and $\beta^t_{knv}$ corresponding to the arms in the allocation $A^t$
```

In this setting $\boldsymbol{\mu} \in \mathbb{R}^{K \times N \times (V+1)}$ is a 3 dimensional matrix whose elements μ_{knv} corresponds to the expected volume returned by the allocation of a volume of $A^t_{kn} = v$ on the k-th dark pool at price p_n. The reward function corresponding to a superarm A^t is defined as follows:

$$f_{\boldsymbol{\mu}}(A^t) := \sum_{k=1}^{K} \sum_{n=1}^{N} \mu_{knA^t_{kn}}\, p_n. \tag{4}$$

Finally, the oracle $\mathrm{Opt}(\cdot)$ is the same presented by [Nuara et al. 2018, 2022] where the budget therein considered corresponds to the asset volume and the reward function is $A^t_{kn} p_n$. Note that this algorithm has been shown to be optimal, i.e., to provide the allocation in $\mathcal{S}$ maximizing $f_{\boldsymbol{\mu}}(\cdot)$.

A CMAB algorithm $\mathfrak{U}$ is evaluated in terms of its expected pseudo-regret, formally defined as:

$$Reg_t(\mathfrak{U}) := t\, r^* - \sum_{h=1}^{t} \sum_{k=1}^{K} \sum_{n=1}^{N} \mathbb{E}[r^h_{kn}] \mathbb{1}\{A^h_{nk} > 0\}\, p_n, \tag{5}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, the expected value $\mathbb{E}[\cdot]$ is taken w.r.t. the stochasticity of the algorithm and of the reward function r^h_{kn}.

4 DP-CMAB Algorithm

In this section, we introduce the DP-CMAB algorithm, designed to solve the DPSOR problem by adapting strategies of general-purpose CMAB algorithms to our setting. The core idea of our approach is to estimate the matrix $\boldsymbol{\mu}$, using either a frequentist or a Bayesian approach. Specifically, we model the random variable X_{knv} whose expected value corresponds to the probability that an allocation of v units of the asset in the k-th dark pool at a price p_n is successful.

Throughout the learning period, we store the quantities α^t_{knv} and β^t_{knv}, denoting the number of successes (with the full quantity executed) and failures (partial executions) after t rounds of an allocation of v units of the asset in the k-th dark pool at price p_n. In this way, we will build proxies for $\boldsymbol{\mu}$, which will be used for learning.

Algorithm 2 No-propagation Update

```
1: for each arm $i \in A^t$ corresponding to allocation $A^t_{kn}$ do
2:     Observe $r^t_{knv}$ corresponding to arm $i$
3:     if $r^t_{knv} = A^t_{kn}$ then
4:         $\alpha^{t+1}_{knv} \leftarrow \alpha^t_{knv} + 1$
5:     else
6:         $\beta^{t+1}_{knv} \leftarrow \beta^t_{knv} + 1$
7: return $\boldsymbol{\alpha}^{t+1}, \boldsymbol{\beta}^{t+1}$
```

The procedure corresponding to the DP-CMAB algorithm is provided in Algorithm 1. At the beginning of the learning period, the algorithm initializes for each arm a Beta distribution with parameters $\alpha^t_{knv} = \beta^t_{knv} = 1$, i.e., sets a Uniform prior over the expected value of the variables X_{knv}. At each time step t, the algorithm generates a value θ_{knv} which is used as proxy of vX_{knv} (Line 6) to be used in the oracle optimization procedure. We propose different strategies to compute such a value, formally:

$$\theta^t_{knv} = v\left(\frac{\alpha^t_{knv} - 1}{\alpha^t_{knv} + \beta^t_{knv} - 2} + \sqrt{\frac{2\log(t)}{\alpha^t_{knv} + \beta^t_{knv} - 2}}\right), \tag{6}$$

$$\theta^t_{knv} \sim v\ Beta\left(\alpha^t_{knv}, \beta^t_{knv}\right), \tag{7}$$

$$\theta^t_{knv} = v\ Q\left(1 - \frac{1}{t(\log T)^5}; Beta(\alpha^t_{knv}, \beta^t_{knv})\right), \tag{8}$$

where $Q(\eta, D)$ represents the quantile of order η of a generic distribution D, and $Beta(\alpha, \beta)$ is the beta distribution with parameters α and β. From now on, the three different flavors of the DP-CMAB algorithm will be denoted with DP-CUCB, DP-TS, and DP-Bayes UCB, respectively. The first two strategies are the adaptation to our problem of the algorithms presented by Chen et al. [2013] and Wang and Chen [2018], respectively. Instead, the third version of the DP-CMAB is inspired by the Bayes-UCB algorithm, designed by Kaufmann et al. [2012], which combines the frequentist and Bayesian approaches by selecting the value of the arms by constructing upper confidence bounds on the Beta posterior distributions.

Once θ^t_{knv} has been computed, it is used to provide the allocation A^t (Line 7) to use in the next round (Line 8). This allocation is determined using $\text{Opt}(\cdot)$ on the matrix $\boldsymbol{\theta}$. After allocating the shares, the agent receives feedback from the environment (Line 9), that is used to update the Beta distributions of the arms. A standard update rule changes only the values of the parameters of the arms which were included in the current allocation, i.e., the arms corresponding to the entries of the allocation s.t. $A^t_{nk} > 0$. Such an update strategy is detailed in Algorithm 2, in which, depending if it is a success (Line 4) or failure (Line 6) the α^t_{knv} or β^t_{knv} parameters are updated, respectively. In the following section, we will present other approaches for the update which exploit the correlations present in the DPSOR problem.

From a theoretical point of view, the DP-CUCB and DP-CTS algorithms provide guarantees on the upper bound of the regret provided over a specific time horizon T. First, we need to define some quantities related to the specific problem that characterize the regret. Let us denote with $\mathcal{S}_B := \{S \in \mathcal{S} | S \neq \text{Opt}(V, \boldsymbol{\mu})\}$ as the set of *bad* super arms. For any given underlying arm $i \in \mathcal{M}$, we define:

$$\Delta^i_{\min} := r^* - \max_{S \in \mathcal{S}_B | i \in S} \{f_{\boldsymbol{\mu}}(S)\}, \tag{9}$$

$$\Delta^i_{\max} := r^* - \min_{S \in \mathcal{S}_B | i \in S} \{f_{\boldsymbol{\mu}}(S)\}, \tag{10}$$

$\Delta_{\max} := \max_{i \in \mathcal{M}} \Delta^i_{\max}$, and $\Delta_{\min} := \min_{i \in \mathcal{M}} \Delta^i_{\min}$.

Moreover, some assumptions are required for the DP-CUCB and DP-CTS algorithms to have theoretical guarantees: monotonicity and bounded smoothness of the reward function. Monotonicity means that given two super arms $S, S' \in \mathcal{S}$ such that for all $i \in \mathcal{M}$, $\mu_i \leq \mu'_i$, the expected reward obtained by playing S is smaller or equal to the one obtained by playing S', i.e., $f_{\boldsymbol{\mu}}(S) \leq f_{\boldsymbol{\mu}'}(S)$ for all $S \in \mathcal{S}$. This assumption is trivially satisfied by the DP-SOR problem thanks to the linear dependence between the reward function, and the expected volume returns μ_{knv}. Instead, bounded smoothness consists in the existence of a strictly increasing function $g(\cdot)$, such that for any two expectations $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$, we have $|f_{\boldsymbol{\mu}(S)} - f_{\boldsymbol{\mu}'(S)}| \leq g(\Lambda)$ if $\max_{k,n,v} |\mu_{knv} - \mu'_{knv}| \leq \Lambda$. It is easy to show that in the DP-SOR problem the bounded smoothness function is $g(\Lambda) = V\ p_n\ \Lambda$.

Thanks to the above definitions and assumptions, we have:

THEOREM 4.1 (PSEUDO-REGRET OF THE DP-CUCB ALGORITHM [CHEN ET AL. 2013]). *The Pseudo-regret for the DP-CUCB applied to a DPSOR problem over a time horizon of T satisfies:*

$$R_T(\textit{DP-CUCB}) \leq \sum_{i \in \mathcal{M}, \Delta^i_{\min} > 0} \left(\frac{12\ln(T)\ (V\ p_n)^2}{\Delta^i_{\min}}\right) + \left(\frac{\pi^3}{3} + 1\right) \cdot VKN \cdot \Delta_{\max}. \tag{11}$$

Note that the proof of this theorem follows from the application of Theorem 1 in [Chen et al. 2013] to the DPSOR setting.

Similarly, the DP-CTS agent, implements a strategy matching the one proposed by the Combinatorial Thompson Sampling (CTS) introduced by Wang and Chen [2018].

THEOREM 4.2 (PSEUDO-REGRET OF THE DP-CTS ALGORITHM [WANG AND CHEN 2018]). *For any $0 < \epsilon < \frac{\Delta_S}{2Vp_n(k^{*2}+2)}$ and $T \in \mathbb{N}$, the pseudo-regret for the DP-CTS algorithm satisfies:*

$$R_T(\textit{DP-CTS}) \leq \sum_{i \in \mathcal{M}} \frac{[2\ln(KV) + 6]V^2\ p_n^2 \ln(2^{2|\mathcal{M}|}T)}{\min_{S \in \mathcal{S}} \Delta_S - 2Vp_n(k^{*2} + 2)\epsilon} + o(\ln(T)), \tag{12}$$

where $\Delta_S := r^* - f_{\boldsymbol{\mu}}(S)$ and k^* is the number of arms contained in the optimal superarms (assuming it is unique).

Similarly, this results is derived applying Theorem 1 by [Wang and Chen 2018] to the DP-SOR setting. Summarizing, both these results are providing no-regret problem-dependent guarantees with order $O(\ln T)$. This consists in a guarantee that the algorithms are converging asymptotically to the optimal solution of the DP-SOR problem. We remark that, to the best of our knowledge, similar results for the DP-Bayes UCB algorithm in a general setting have not yet been provided in literature.

However, applying the algorithms provided by the bandit literature does not exploit the specific structure existing among the volumes and prices of dark pools. In the following section, we will

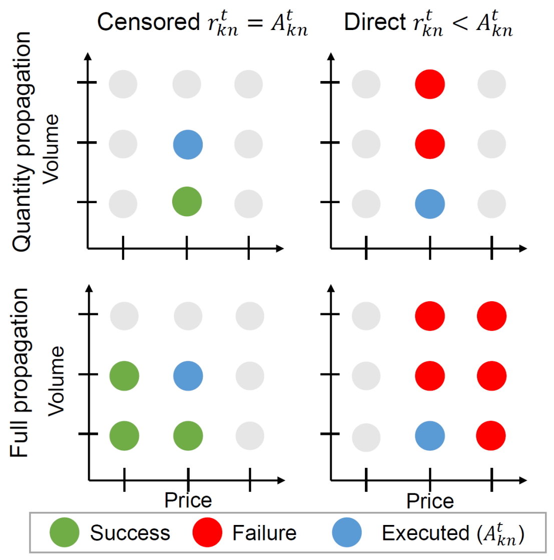

Figure 1: Propagation schemes for quantity (top) and full (bottom) depending if the outcome of a dark pool allocation was successful (right) or failure (left).

propose some modifications to the update rule of the Beta parameters that exploit the dependence among the available arms.

4.1 Exploiting the DPSOR Problem Characteristics: the Propagation Updates

To make the learning algorithm more efficient, we exploit some properties of the dark pool environment. Specifically, we use the fact that by setting a volume and a price, we might infer information on other price/volume pairs on the same dark pool once the reward r^t_{knv} is revealed.[5]

Let us focus on the asset quantities A^t allocated at each round, considering we want to sell a certain asset.[6] If the allocation A^t_{kn} was successful ($r^t_{knv} = A^t_{kn}$), then all the allocations $A'_{kn} < A_{kn}$ would have been successful too. Conversely, if the allocation A_{kn} returned a failure ($r^t_{knv} < A^t_{kn}$), then each allocation $A'_{kn} > A_{kn}$ would have also failed. This inference is implemented in the update presented in Algorithm 3, in which the parameters of the Beta distributions are updated according to the above-described strategy. A graphical representation of the quantity propagation update carried out in a single dark pool is provided in Figure 1 on the two top pictures. The blue dots represent the value of volume and price of the order sent to a specific dark pool, while the green ones are the volume/price pairs whose distribution is updated, thanks to the propagation, with a success (update of the α^t_{knv} parameter, Line 4) and the red ones correspond to the volume/price pairs whose distribution are updated with a failure (update of the β^t_{knv} parameter, Line 7).

[5]Note that a different approach would consider each darkpool/price pair as an independent entity. However, such an approach would not take advantage of the information that can be extracted from other prices coming from the same darkpool.

[6]Note that we focus on the case in which we need to sell an asset, but the case in which we are buying is symmetrical.

Algorithm 3 Quantity propagation

```
1:  for each arm $i \in A^t$ corresponding to allocation $A^t_{kn}$ do
2:    if $r^t_{knv} = A^t_{kn}$ then
3:      for each $v' \le A^t_{kn}$ do
4:        $\alpha^{t+1}_{knv'} \leftarrow \alpha^t_{knv'} + 1$
5:    else
6:      for each $v' \ge A^t_{kn}$ do
7:        $\beta^{t+1}_{knv'} \leftarrow \beta^t_{knv'} + 1$
8:  return $\boldsymbol{\alpha}^{t+1}, \boldsymbol{\beta}^{t+1}$
```

Algorithm 4 Full propagation

```
1:  for each arm $i \in A^t$ corresponding to allocation $A^t_{kn}$ do
2:    if $r^t_{knv} = A^t_{kn}$ then
3:      for each $v' \le A^t_{kn}$ do
4:        for each $n' \le n$ do
5:          $\alpha^{t+1}_{kn'v'} \leftarrow \alpha^t_{kn'v'} + 1$
6:    else
7:      for each $v' \ge A^t_{kn}$ do
8:        for each $n' \ge n$ do
9:          $\beta^{t+1}_{kn'v'} \leftarrow \beta^t_{kn'v'} + 1$
10: return $\boldsymbol{\alpha}^{t+1}, \boldsymbol{\beta}^{t+1}$
```

The reasoning applied to the volumes can also be applied to the prices. Indeed, given an allocation A^t_{kn}, if a success occurred, we would have also succeeded for lower prices, i.e., for $A^t_{kn'}$ with $n' < n$. Similarly, with a failure on a specific arm, the same would have occurred for larger prices, i.e., for all $A^t_{kn'}$ with $n' > n$. This allows combining such an update on the prices with the one on the volumes to maximize the amount of information provided within a single round. The corresponding update of the Beta distribution modeling the probability that an allocation is successful is provided in Algorithm 4, while an example of the full propagation in a single dark pool is provided in Figure 1 (bottom). Note that, even from the provided toy example in the figures mentioned above, the number of distributions that are updated by the full propagation is far larger than the number of distributions updated with the classical and quantity propagations.

5 Experiments

In this section, we analyze the empirical performance of the DP-CMAB algorithms developed in the previous sections. To have a realistic evaluation of what has been presented, we developed a realistic simulation environment, i.e., based on real-world data, to test such approaches. First, we analyze how the domain knowledge helps the DP-CMAB agents to achieve a better performance by evaluating how the different types of propagation affect the regret. Subsequently, we compare them to the approaches proposed by Agarwal et al. [2010]; Ganchev et al. [2010]. Finally, we evaluate the performance of the proposed methods on five different asset models to evaluate the robustness of our method when the underlying asset is changing.

5.1 Experimental setting

In our experimental setting, the task consists in selling $V = 10$ units, each of which is composed by 20,000 shares, across $K = 10$ dark pools, and the prices span in the set $\mathcal{P} = \{90, 91, \dots, 100\}$. This process is repeated over a time horizon of $T = 1,000$ rounds, where each round is assumed to occur every 10 minute for a total trading time of one week. The results are averaged over 20 independent runs (semitransparent areas in the plots represent the 95% confidence intervals of the empirical mean value).

Since the baselines do not allow to specify the price, we devised the following approaches to apply such algorithms to the dark pool scenario:

- The *Random price selection* the strategy selects a random price in $\mathcal{P}$ at the beginning of every run $r \in [R]$ and continues to sell the units at the fixed price for each $t \in [T]$ of that run;
- The *Oracle price selection* selects the *oracle price* $p^* \in \mathcal{P}$, which is the best single price that maximizes the expected cumulative dollar volume across all runs R. Formally, let $\lambda_p^{r,t}$ be the sum of the total liquidity available across all the K dark pools for the run r at time t and price p, the optimal price is defined as:

$$p^* := \operatorname*{argmax}_{p \in \mathcal{P}} \frac{1}{R} \sum_{r=1}^{R} \sum_{t=1}^{T} (p \ \min\{V, \lambda_p^{r,t}\}). \tag{13}$$

We remark that this strategy represents an oracle since an agent would select p^* only 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 performances of the different algorithms have been tested by means of empirical average regret $\widehat{Reg}_t(\mathfrak{U})$ over the time horizon T, for $t \in [T]$ (the lower, the better), averaged over the different runs, which is the empirical counterpart of the expected pseudo-regret $Reg_t(\mathfrak{U})$ defined in Equation (5). Moreover, we also report the averaged (over the runs) per-round dollar volume $R_t(\mathfrak{U})$ over the rounds t (the larger, the better).

The following experiments were obtained by running Python 3.7 multi-core, with NumPy and Pandas as the main necessary libraries. The data generated for the experiments took ≈ 4 hours on a 64-core Linux machine with Intel(R) Xeon(R) E5-4610 v2 @ 2.30GHz CPU.

5.2 Liquidity Generation

The generation of the underlying dark pools liquidity is a fundamental and essential aspect of our problem since it determines the environment our algorithms will interact with. In what follows, we describe the procedure to generate the liquidity of the dark pools we used in the experiments.

First, we started from the historical market data of different stocks in the form of order book messages of the executed bid orders.[7] Then, the historical data is extracted from the data feed of the NASDAQ exchange.[8] From this data, the order book messages were generated following the steps detailed by Bernasconi-De-Luca et al. [2021]. Subsequently, we removed the outliers by looking at the distribution of the number of orders and total volume exchanged every 10 minutes, using the DBSCAN algorithm [Ester et al. 1996]. Finally, we partitioned the entire dataset into independent batches of messages taken every 10 minutes. Based on a random subset of these independent batches, for each dark pool, we generated the liquidity matrix from which the dark pool liquidity is sampled. Therefore, each dark pool provides different volumes corresponding to the same price. This process has been repeated for five different stocks: Apple (AAPL), Facebook (FB), Amazon (AMZN), Microsoft (MSFT), and Google (GOOGL).

[7]The order book messages we used implements the FIX protocol language. For more details, see https://www.fixtrading.org/what-is-fix/.
[8]The historical data can be found at ftp://emi.nasdaq.com/ITCH/Nasdaq%20ITCH.

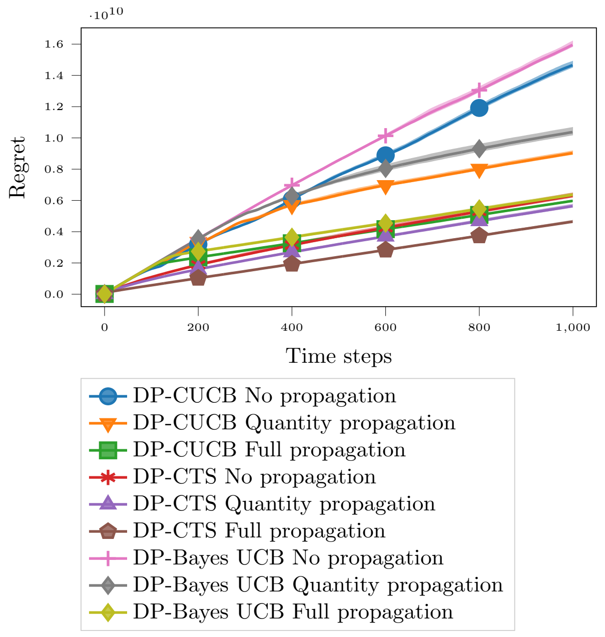

Figure 2: Regret of the different flavours of the DP-CUCB, DP-CTS and DP-Bayes UCB algorithms.

5.3 Experiments on the AAPL asset

In this section, we show the results of the experiments on the liquidity generated from the data for the Apple (AAPL) order books from the following days: 2019-1-30, 2019-3-27, 2019-7-30, 2019-8-30, 2019-10-30, 2019-12-30, and 2020-1-30.[9]

Figure 2 provides the results in terms of the empirical regret of our algorithms. Overall, the ones using the full propagation update strategy achieve the best performance in terms of regret, i.e., these flavors of the DP-CMAB algorithms exhibit a lower regret than the corresponding versions exploiting the no propagation or quantity propagation. This is due to the fact that this approach can gather more information in a single round, converging faster to the optimal arm. As experienced in many other settings in literature, the TS-like approach is the one providing the best performances for each typology of propagation. This is because it uses all the information available about the distribution of the rewards, modeling

[9]The results on each other asset are in line with the ones we reported here. In what follows, we also present results summarizing them over the different assets.

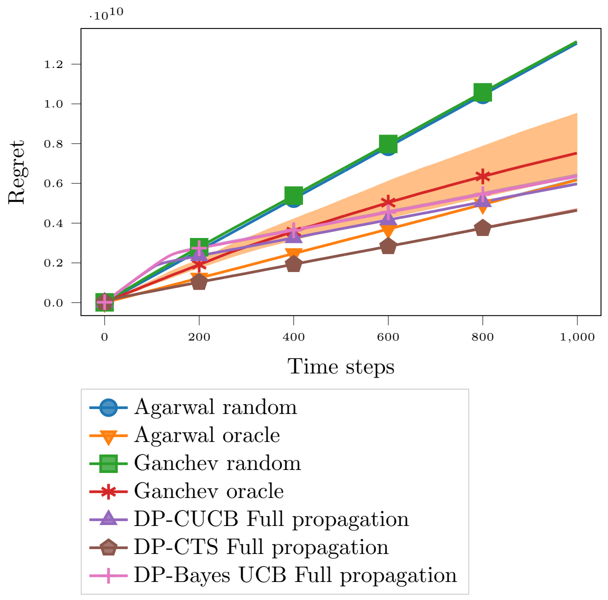

Figure 3: Regret of the DP-CMAB with full propagation update and baselines.

Table 1: Regret ($\cdot 10^{10}$) of DP-CMAB agents at round $t = 1,000$.

	No prop.	Quantity prop.	Full prop.
DP-CUCB	1.51	0.95	0.60
DP-CTS	0.63	0.59	0.55
DP-Bayes UCB	1.59	1.05	0.65

the expected value as a Beta distribution and using it to generate samples to feed to the DP-CTS algorithm.

Table 1 summarizes the average regret obtained by the DB-CMAB agents after $t = 1,000$ rounds, i.e., at the end of the time horizon. The reduction in terms of regret of the DP-CTS without any propagation strategy w.r.t. the other approaches without propagation is $> 58\%$, meaning that we are considerably reducing the losses, due to the lack of information, by using such an algorithm. The use of the quantity propagation is most effective for the DP-CUCB, and the DP-Bayes UCB approaches, leading to a decrease of $\approx 35\%$ of the regret, while it only provides a reduction of $\approx 6\%$ for DP-CTS. Similarly, the use of the full propagation (over the quantities and prices) reduces the regret of the DP-CUCB and the DP-Bayes UCB of $\approx 60\%$ and that of DP-CTS of $\approx 12\%$ w.r.t. their counterparts with no propagation. Overall, these results justify empirically the introduction of the propagation approach, which significantly reduces the regret of the proposed algorithms. In the following experiments on the AAPL asset, we will only report the results using the full propagation since they are consistently better than those obtained with the other flavors of our algorithm.

Figure 3 compares the regret of the DP-CMAB agents exploiting the full propagation update and that of the baselines. For the sake of presentation, we omitted the confidence intervals of the baseline agents using the random price selection since they are

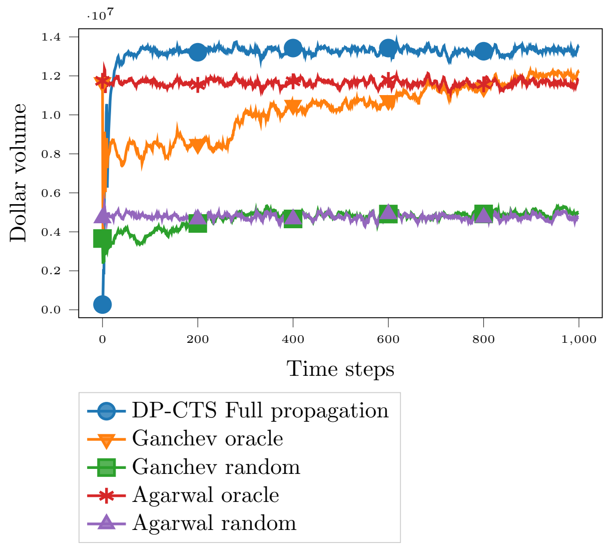

Figure 4: Average dollar volumes of the DP-CTS agent with full propagation and the baselines.

orders of magnitude larger than that of other methods. This provides evidence that such a strategy might lead to results having high variance over the different runs, therefore making such an approach not reliable in practice. Moreover, such baselines also have larger regret than the other considered algorithms. This is due to the fact that these algorithms may choose to play with an excessively low price, thus missing the opportunity to gain a large dollar volume, or with an excessively large price, thus being able to allocate only a small fraction of the available liquidity V.

The baseline agents implementing the oracle price selection strategy perform better than their random counterparts thanks to the a priori knowledge about how the liquidity is distributed across the dark pools. However, the full propagation DP-CMAB algorithms achieve a lower regret than the baselines, even without requiring such prior information concerning the distribution of the liquidity across the dark pools. The DP-CUCB and DP-Bayes UCB algorithms provide a larger regret than those of the baselines in the first $t = 200$ rounds, but they manage to reduce the overall regret at the end of the period at $t = 1,000$. This suggests that such algorithms should be adopted over long time periods, while one may use simpler strategies in the case the number of transactions is expected to be small. However, the use of the DP-CTS algorithms seems to be the best option over arbitrary time horizons since the expected value of the regret is consistently lower than that of the other algorithms.

Figure 4 shows the dollar volume per round of the different algorithms. It provides empirical evidence that the DP-CTS algorithm accumulates most of the regret in the early stages of the time horizon ($t < 100$), due to the initial need to perform exploratory allocations to gather enough knowledge about the environment. Once the learning phase progresses and the algorithm collects information on the environment, the dollar volume converges to the optimal value. This behaviour is also experienced by the Ganchev with oracle chosen price, which is constantly increasing its performance over time. However, it seems that, after $t = 1,000$ rounds, it has not yet reached the optimum, implying that its regret is larger than the one of DP-CTS. The other baseline algorithms seem to converge quickly to a specific solution which, unfortunately, is far from the one selected by DP-CTS, e.g., the Agarwal random strategy is providing, on average, a dollar value of $\approx 35\%$ of the one provided by DP-CTS. In conclusion, in this setting, we have that the DP-CTS algorithm with the full propagation update is providing the best results in terms of regret.

5.4 Experiments on all the assets

In this experiment, we run our algorithms and the baselines on the five assets mentioned in Section 5.2. The experimental setting is the same as the one described in Section 5.1 and has been repeated for each different asset. As a performance index, we reported the average ranking Ra (and the corresponding standard deviation σ_{Ra}) of each algorithm (the smaller, the better) we analysed in the previous sections since the value of the regret over different assets cannot be compared directly. Specifically, an algorithm having the smallest regret for a specific asset has a rank 1, and the one having the largest regret has a rank 13.

Table 2 reports the ranking of the different algorithms over different rounds of the time horizon T. The DP-CTS algorithm is the one obtaining the best performance since it is able to provide the lowest regret in each one of the five different settings. Similarly, the sole use of the quantity propagation allows the DP-CTS to be the second-best consistently. Finally, the third place is consistently occupied by DP-CTS without propagation update for $t > 200$. This strengthens the idea that this algorithm is the one preferred to deal with the problem of allocating volumes in the DP-SOR problem. The other algorithms closely following the DP-CTS are the DP-CUCB and DP-Bayes UCB in their flavor with full propagation. Therefore, this provides further empirical evidence that the application of the full propagation is providing a significant improvement to the developed CMAB techniques. Following these strategies, we have the two baselines having the information on the price. However, such algorithms are not applicable in real-world settings since the information about the optimal price p^* is commonly unknown.

6 Conclusions

The focus of this paper is the design of a Dark Pool Smart Order Routing algorithm that learns the optimal allocation strategy among multiple dark pools. We accomplish this by introducing the DP-CMAB family of algorithms, which extends the currently available bandit literature by working with limits orders (thus specifying also the price) rather than market orders. The DP-CMAB algorithms consist of the application of state-of-the-art CMAB approaches to the SOR framework. We show that the theoretical guarantees on the Regret of the CMAB algorithms also hold in this context. Furthermore, we exploit the financial properties of the environment to make our algorithms more efficient. Finally, we compare the empirical performance of our algorithms with two baselines, i.e., the works by Agarwal et al. [2010]; Ganchev et al. [2010], by leveraging real data of lit exchanges. The results show that, in this scenario, the DP-CMAB agents outperform the baseline algorithms by exploiting the knowledge of the problem characteristics.

The next steps include the extension of the regret bounds for the algorithms implementing the propagation updates and the development of techniques able to optimize SOR for different kinds of venues at the same time.

References

Alekh Agarwal, Peter Bartlett, and Max Dama. 2010. Optimal Allocation Strategies for the Dark Pool Problem. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Vol. 9. 9–16.

Robert Almgren and Neil Chriss. 2001. Optimal execution of portfolio transactions. *Journal of Risk* 3 (2001), 5–40.

Robert Almgren and Bill Harts. 2008. A dynamic algorithm for smart order routing. *White paper StreamBase* (2008), 1–11.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47 (05 2002), 235–256.

Martino Bernasconi-De-Luca, Luigi Fusco, and Ozrenka Dragić. 2021. martinobdl/ITCH: ITCH50Converter. https://doi.org/10.5281/ZENODO.5209267

Martino Bernasconi de Luca, Edoardo Vittori, Francesco Trovò, and Marcello Restelli. 2021. Conservative Online Convex Optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 19–34.

Dimitris Bertsimas and Andrew W Lo. 1998. Optimal control of execution costs. *Journal of Financial Markets* 1, 1 (1998), 1–50.

Wei Chen, Yajun Wang, and Yang Yuan. 2013. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the International conference on Machine Learning (ICML)*. 151–159.

Rama Cont and Arseniy Kukanov. 2017. Optimal order placement in limit order markets. *Quantitative Finance* 17, 1 (2017), 21–39.

Puja Das, Nicholas Johnson, and Arindam Banerjee. 2013. Online lazy updates for portfolio selection with transaction costs. In *Proceedings of the conference on Artificial Intelligence (AAAI)*. 202–208.

Hans Degryse, Mark Van Achter, and Gunther Wuyts. 2009. Shedding Light on Dark Liquidity Pools. *Trading* 1 (2009), 147–155.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the conference on knowledge discovery and data mining (SIGKDD)*. 226–231.

Thierry Foucault and Albert J Menkveld. 2008. Competition for order flow and smart order routing systems. *The Journal of Finance* 63, 1 (2008), 119–158.

Kuzman Ganchev, Yuriy Nevmyvaka, Michael Kearns, and Jennifer Wortman Vaughan. 2010. Censored exploration and the dark pool problem. *Commun. ACM* 53, 5 (2010), 99–107.

Olivier Guéant. 2016. *The Financial Mathematics of Market Liquidity: From optimal execution to market making*. Vol. 33. CRC Press.

Olivier Guéant, Charles-Albert Lehalle, and Joaquin Fernandez-Tapia. 2012. Optimal portfolio liquidation with limit orders. *SIAM Journal on Financial Mathematics* 3, 1 (2012), 740–764.

Terrence Hendershott and Haim Mendelson. 2000. Crossing networks and dealer markets: Competition and performance. *The Journal of Finance* 55, 5 (2000), 2071–2115.

Woonghee Huh and Paat Rusmevichientong. 2009. A Nonparametric Asymptotic Analysis of Inventory Planning with Censored Demand. *Math. Oper. Res.* 34 (02 2009), 103–123.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. 2012. On Bayesian upper confidence bounds for bandit problems. In *Proceedings of the International Confererence on Artificial intelligence and Statistics (AISTATS)*. PMLR, 592–600.

Peter Kratz and Torsten Schöneborn. 2014. Optimal liquidation in dark pools. *Quantitative Finance* 14, 9 (2014), 1519–1539.

Sophie Laruelle, Charles-Albert Lehalle, and Gilles Pages. 2011. Optimal split of orders across liquidity pools: a stochastic algorithm approach. *SIAM Journal on Financial Mathematics* 2, 1 (2011), 1042–1076.

Costis Maglaras, Ciamac Moallemi, and Hua Zheng. 2012. Optimal Order Routing in a Fragmented Market. *Preprint* (05 2012), 1–5.

Marco Mussi, Gianmarco Genalti, Francesco Trovò, Alessandro Nuara, Nicola Gatti, and Marcello Restelli. 2022. Pricing the Long Tail by Explainable Product Aggregation and Monotonic Bandits. In *Proceedings of the conference on knowledge discovery and data mining (SIGKDD)*. 3623–3633.

Alessandro Nuara, Francesco Trovò, Nicola Gatti, and Marcello Restelli. 2018. A combinatorial-bandit algorithm for the online joint bid/budget optimization of pay-per-click advertising campaigns. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 2379–2386.

Alessandro Nuara, Francesco Trovò, Nicola Gatti, and Marcello Restelli. 2022. Online joint bid/daily budget optimization of internet advertising campaigns. *Artificial*

	$t = 200$		$t = 400$		$t = 600$		$t = 800$		$t = 1000$	
	Ra	σ_{Ra}	Ra	σ_{Ra}	Ra	σ_{Ra}	Ra	σ_{Ra}	Ra	σ_{Ra}
Ganchev Random	8.4	0.49	10.0	1.09	11.6	0.80	11.8	0.98	11.8	0.98
Ganchev Oracle	5.6	0.49	6.0	0.89	6.4	0.80	6.6	1.85	6.8	2.04
Agarwal Random	6.4	1.2	8.8	1.17	10.2	0.75	10.6	0.80	10.6	0.80
Agarwal Oracle	3.4	0.49	4.2	0.40	4.6	0.80	5.8	0.40	6.2	0.75
DP-CUCB No propagation	9.6	0.49	11.8	0.98	11.8	0.98	11.8	0.40	12.0	0.63
DP-CUCB Quantity propagation	11.4	0.49	10.2	1.16	9.2	0.75	9.0	0.63	8.8	0.75
DP-CUCB Full propagation	8.6	1.49	6.4	0.49	6.0	0.63	5.6	0.80	5.4	0.49
DP-CTS No propagation	3.6	0.49	3.0	0.0	3.0	0.0	3.0	0.0	3.0	0.0
DP-CTS Quantity propagation	2.0	0.0	2.0	0.0	2.0	0.0	2.0	0.0	2.0	0.0
DP-CTS Full propagation	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
DB-B-UCB No propagation	13.0	0.0	12.8	0.40	12.0	1.55	11.6	1.74	11.4	1.62
DP-B-UCB Quantity propagation	11.6	0.49	9,4	1.02	8.2	0.4	7.6	0.49	7.4	0.8
DP-B-UCB Full propagation	6.4	0.8	5.4	1.02	5.0	1.09	4.6	1.2	4.6	1.2

Table 2: Mean and standard deviation of the ranking of the algorithms applied to the 5 different assets.

Intelligence 305 (2022), 103663.

Gregor Pujol and Alexander Brueckner. 2009. Smart Order Routing and Best Execution.. In *Proceedings of the Americas Conference on Information Systems (AMCIS)*, Vol. 3. 155.

Yan Qin, Ruoxuan Wang, Asoo J. Vakharia, Yuwen Chen, and Michelle M.H. Seref. 2011. The newsvendor problem: Review and directions for future research. *European Journal of Operational Research* 213, 2 (2011), 361–374.

Gary Shorter and Rena S. Miller. 2014. Dark Pools in Equity Trading: Policy Concerns and Recent Developments. In *University of North Texas Libraries, UNT Digital Library*. 1–18. https://digital.library.unt.edu/ark:/67531/metadc461960/

Francesco Trovò, Stefano Paladino, Marcello Restelli, and Nicola Gatti. 2018. Improving multi-armed bandit algorithms in online pricing settings. *International Journal of Approximate Reasoning* 98 (2018), 196–235.

Edoardo Vittori, Martino Bernasconi de Luca, Francesco Trovò, and Marcello Restelli. 2020. Dealing with transaction costs in portfolio optimization: online gradient descent with momentum. In *Proceedings of the International Conference on AI in Finance (ICAIF)*. 1–8.

Siwei Wang and Wei Chen. 2018. Thompson Sampling for Combinatorial Semi-Bandits. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vol. 80. 5114–5122.

Linlin Ye. 2016. Understanding the Impacts of Dark Pools on Price Discovery. (2016), 1–73.

Haoxiang Zhu. 2013. Do Dark Pools Harm Price Discovery? *The Review of Financial Studies* 27, 3 (12 2013), 747–789.