Dancing in the Dark: 
Optimal liquidity search under portfolio constraints

ABSTRACT

One of the core responsibilities of many Institutional traders is managing cash and risk constraints of a portfolio. Traders often do not take advantage of dark and block trading due to the risk of an unpredictable and unbalanced change to the composition of the executing list. Said differently, the randomness of dark fills makes it very difficult to constrain an optimization using dark as the only source of liquidity. In this paper, we offer a solution to this problem using stochastic programming to create linear constraints for a quadratic optimization. We believe this research can be used by algorithm designers to bridge the gap between two dissimilar, yet useful products: dark aggregation and portfolio trading algorithms.

THE CHALLENGE OF PORTFOLIO TRADING IN DARK VENUES

Dark aggregation algorithms use a mixture of tactics to execute orders in dark pools while trying to avoid the pitfalls of gaming and information leakage. These tools are traditionally used on a single-stock basis and cannot easily be applied to multi-security trading. Foremost among the reasons for this: the random arrival of fills makes it difficult to maintain cash neutrality and to manage the risk of the residuals.

On the other side of the spectrum, portfolio trading algorithms minimize the competing effects of cost and execution risk while maintaining cash neutrality. They typically operate with two components. The top layer is responsible for generating an optimal trading schedule which then sends out smaller orders to a schedule-based [e.g. VWAP] or implementation-shortfall algorithm to trade. The optimization layer believes that the schedules can always be followed, which inherently assumes all orders can be completed. If we try to apply these concepts to dark trading, this assumption becomes problematic.

The unsystematic nature of non-displayed venues forces us to take a different path for two reasons:

- Dark Fills arrive inconsistently so we need a mechanism to control cash neutrality.
- A substantial part of the portfolio costs will lie in the unrealized cost. The most obvious challenge is in illiquid stocks which trade much slower and with less predictability than liquid ones. The higher cost (less liquid) names should be the first priority in portfolio strategies, rather than allowing only the liquid stocks to trade, reducing hedging material for the trader.
To summarize, traders face two challenges when trading lists on dark venues. First, they need to maintain the cash neutrality of the list while maximizing their exposure to potential liquidity. Second, they need a framework to determine the priority in which they choose what securities to execute.

**Maintaining cash neutrality**

Maintaining cash neutrality means that the total executed value of both the sell and buy sides of a list is kept in line or at a set ratio as specified. It is crucial for the following reasons:

- Cash risk: Highly correlated with market risk for most trade lists
- Settlement risk: end-of-the-day imbalances can be costly

When working to maintain cash neutrality in non-displayed marketplaces, we face two difficulties: we have no guarantee of executing the orders and we can potentially fill more than expected when we discover deep liquidity in a name.

A naïve workaround would be to submit no more than a tolerance threshold [a small percentage of the overall list value]. On the one hand, we would never violate the constraints; however, we would under-expose our orders to the available liquidity in the marketplace, especially to potential dark blocks which usually represent low impact natural liquidity. Or we could do the contrary and expose everything to dark pools. When an imbalance is created, we could go to lit venues and execute orders on the slow side. This isn’t very efficient as the hedging trades will likely cause more impact than is saved trading opportunistically. Additionally, this approach would probably result in the easy-to-trade stocks being executed quickly, leaving many of the high cost names to work through over time.

Our solution is to probabilistically control the imbalance so that we can maximize exposure to dark liquidity, while staying within the cash bounds that we have set.

**Defining objectives**

In the model we have created, traders can have two objectives: trading as much as possible in dark [while keeping risk low] or improving the liquidity profile of the list, making the residuals more manageable to trade. In both cases, we assume that dark pools lower trading costs but at the expense of lower fill rates. Additionally, it is essential to consider that traders may eventually switch to non-discretionary strategies like VWAP.

- **Minimizing Total Cost of Residual:** With this objective, the ambition is to trade the entire list as quickly as possible. It is crucial to note that this differs from trading as much value as possible in dark. In the latter, we would prioritize the liquid names given that they are easy to trade, but we would under-expose the hardest names which would likely result in an increase in the total trading time of the list, especially given cash constraints. Instead, we expose, in priority, the least liquid names given that they will take the longest to trade. Mathematically, this formulation is equivalent to minimizing the total cost of trading the residuals in non-discretionary algorithms. We assume the time it takes to fill an order is proportional to the percentage of ADV [Average Daily Volume]. This metric is also a reasonable proxy for the cost of an order.

- **Minimizing Average Cost of Residual:** In this objective, our goal is to minimize the average cost of the residuals. There can be three different motivations behind this. First, this strategy can be used to prepare the list for more traditional VWAP-type trading. It can also potentially decrease the price of a risk bid when submitting the residual list to dealers. Finally, it can be used in parallel with traditional lit market portfolio trading algorithms.
Let’s take the example of List 1 which contains $10,000 of Stock A (sell, ADV≈34MM shares) and Stock B (buy, ADV≈40K shares). We assume that the Stock A and B orders have respective costs of 2 bps and 100 bps. If we send both at the same time, there is a good chance that we will execute Stock A but not Stock B pushing the average cost up. Therefore, in this simplistic example, to minimize average cost, we only send Stock B initially.

This trading strategy can be referred to as legging. The optimization prefers the difficult names and when they are completed, or an imbalance is created, the optimizer will select the easier names.

<table>
<thead>
<tr>
<th>Exhibit 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent Quantity</td>
</tr>
<tr>
<td>Stock A</td>
</tr>
<tr>
<td>Stock B</td>
</tr>
</tbody>
</table>

Source: ITG

Exhibit 1: Comparison of trading List 1 with different objectives and with cash neutrality off – If the goal is to minimize total cost, we send both names whereas if the goal is to minimize average cost, we only send the name that takes the average cost down.

**HOW TO PROBABILISTICALLY CONTROL THE IMBALANCE?**

One could argue that, since dark fills are random, there is a risk that we will execute too much of certain names. Following that direction and not sending more than the tolerance would make life easier, but we would miss out on a substantial amount of liquidity and incur larger unrealized costs. For lists with many names, it is possible to send much more than the tolerance while taking a limited risk of violating the cash neutrality constraints. Below, we explain how we probabilistically control the imbalance by stochastically designing linear constraints to a quadratic optimization. We start by introducing a model on the probability of different orders getting fills in dark pools. Then, we define the nature of the scenario. Finally, we demonstrate how we link the scenarios with the optimization problem.
Execution model in dark pools

Exhibit 2 represents the distribution of fill quantities when we send a 10,000 shares order into a dark pool aggregator. We see that the distribution is polarized at its two extremes: no fill and full fill. This observation leads us to introduce a tri-state model in which, when one sends an order to dark pools, one can expect to receive a zero fill, a partial fill or a full fill. The probabilities of each state depend on the individual stock and on the quantity submitted.

Defining a scenario

A scenario is a generation of the fill rates for all the names in the list. Each stock in the list gets assigned a fill rate that follows the probability distributions according to the execution model described above. Based off this, the imbalance becomes a linear function of the sent quantities.
**Stochastic constraints**

Due to the randomness of the fills, we cannot generate all scenarios, since the number is an exponential function of the number of names in the list, and imposing all of them in optimization would be costly for a list of reasonable size. We generate N scenarios (i.e. N constraints) that we add to the optimization problem. N is a critical parameter: we don’t want it to be too small because the out-of-sample violating rate would be too high but it shouldn’t be too large, because the result would be under-exposing the orders; additionally, it would add unnecessary burden to the optimization.

Even though not all scenarios are generated, the cash bound violating rate can be controlled since the scenarios have synergies. Live trading will differ from the scenarios we used to solve the problem, but it will share common characteristics with them.

---

**EXHIBIT 3**

Workflow of the generation of constraints

1. Generate stochastically the fill rates of each stock
2. Integrate them into the linear imbalance function
3. Add this imbalance function to the constraints
4. Repeat N times

Source: ITG

These constraints are imposed to control the imbalance. The linearization of the stochastic constraints is the main innovation here. Once the linear constraints are set up, the process becomes a standard quadratic optimization.

**RESULTS**

Now that we have defined the method, let’s see how effective the process is. We will check that we control the imbalance effectively while not under-exposing our orders. We will also verify that we are able to achieve both goals separately: lowering total cost or reducing the average cost of the residuals.

We use List 1 previously introduced. We also use List 2 which has 27 names and List 3 which has 96 names. For each list, we generate 20 Monte-Carlo path simulations. A Monte-Carlo path simulation consists of 20 wedges. A wedge is a trading period of two minutes. We run an optimization for every wedge and then simulate the fills from dark pools. Each example is run with bounds equal to 5% of the notional value of the list.

After running these simulations, we observe how the imbalance and the cost evolve.
### Exhibit 4.1: Imbalance control

<table>
<thead>
<tr>
<th>List</th>
<th># of Names</th>
<th>Objective</th>
<th>Imbalance</th>
<th>Values in K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>List 1</td>
<td>2</td>
<td>Minimize Average Cost</td>
<td><img src="image1" alt="Graph" /></td>
<td>-1, -0.5, 0, 0.5, 1</td>
</tr>
<tr>
<td>List 2</td>
<td>27</td>
<td>Minimize Average Cost</td>
<td><img src="image2" alt="Graph" /></td>
<td>-400, -200, 0, 200, 400</td>
</tr>
<tr>
<td>List 3</td>
<td>96</td>
<td>Minimize Average Cost</td>
<td><img src="image3" alt="Graph" /></td>
<td>-4000, -2000, 0, 2000, 4000</td>
</tr>
</tbody>
</table>

Source: ITG
Exhibit 4.1: These charts represent the cash imbalance as a function of the wedges. Each colored line shows a potential path based off the exposure and dark fill rates. The black lines represent the tolerance. Maintaining a cash balance consists of staying between these 2 lines. These are worst-case scenarios because they assume that we don’t do any real-time monitoring of the imbalance. For clarity reasons, we only show a subset of the paths in Exhibit 4.1.

<table>
<thead>
<tr>
<th>Exhibit 4.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
</tr>
<tr>
<td>List 1</td>
</tr>
<tr>
<td>List 2</td>
</tr>
<tr>
<td>List 3</td>
</tr>
</tbody>
</table>

Source: ITG

Exhibit 4.2: Table shows how much of the value we expose with respect to the cash imbalance tolerance when we minimize total cost. We see that the more names in the list the bigger the multiplier.

As shown in Exhibits 4.1 and 4.2, for a list with 2 names, we obviously don’t have the potential to expose more than the tolerance given the low probability of simultaneous buy and sell fills. But, as the list grows, the numerous buys are more likely to offset the sells and vice versa so we can safely expose more. For example, if the bounds are equal to 5% of notional value, for a list with 27 names, we can send 3.5 times the bounds; furthermore, as the list expands to 96 names we are able to send 4.6 times the threshold values. In the rare case when the imbalance is violated to a small extent, the real-time monitoring of realized imbalance can reduce the risk of violations to near zero.
Exhibit 5: Minimizing Total Cost or Average Cost

<table>
<thead>
<tr>
<th>List</th>
<th># of Names</th>
<th>Objective</th>
<th>Total Cost ($)</th>
<th>Total Cost (bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>List 1</td>
<td>2</td>
<td>Minimize Average Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List 1</td>
<td>2</td>
<td>Minimize Total Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List 2</td>
<td>27</td>
<td>Minimize Average Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List 2</td>
<td>27</td>
<td>Minimize Total Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List 3</td>
<td>96</td>
<td>Minimize Average Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List 3</td>
<td>96</td>
<td>Minimize Total Cost</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: ITG
Exhibit 5: These charts represent the total costs in dollar and the average cost in bps for each simulated path. The thick black lines represent the average path.

The objectives are clearly met. When we attempt to minimize total cost, the total cost goes down faster than when we minimize average cost. Conversely, when the objective is to minimize average cost, the average cost goes down. When we minimize total cost, the average cost goes down on average but in occasional cases there are paths when it moves up slightly as trading some relatively easy names would change the composition of the list and move up average cost of the remaining list.

CONCLUSION

Trading lists in dark can be rewarding. Those who strategically manage risk can find a substantial amount of opportunistic liquidity in dark, reducing unrealized costs efficiently. One can also benefit from the tactics of advanced dark aggregators which find natural liquidity while limiting information leakage, thereby also reducing realized costs. Our research on list trading in dark pools can be used in two different ways. First, traders can use our approach as a self-sufficient execution strategy. In this case, the objective is to minimize the total cost of the residuals, which will finish the list as quickly as possible using dark aggregation strategies. Second, traders can expose the list to dark in advance of a risk bid or in parallel with traditional VWAP or IS algorithms. In this case, the objective is to minimize the average cost of the residuals, trading mostly the illiquid names and always improving the liquidity profile of the residuals.

In either scenario, this new approach to self-directed portfolio trading in dark removes inefficiencies that have historically limited the ability of portfolio traders to find natural liquidity in dark pools. We hope this research can contribute to a market that better matches heterogeneous counterparties, improving performance for both list and single-stock investment strategies.
REFERENCES


