Trade Execution and Dynamic Programming

Suppose we want to sell a large number **N** of shares of some stock. Assume the current price of the stock is $\mathbf{P_0}$ – is it reasonable to expect that we will receive, from our trade, a cash flow of $\mathbf{P_0 N}$? The answer, in general, is *no*:

1. If N is indeed large, then the process of selling the stock will affect the price of the stock, with the result that the price will go down.
2. Moreover, the trade will not be executable all at once – today we may be able to sell $\mathbf{k < N}$ shares, and the remaining $\mathbf{N - k}$ will be left for tomorrow. Tomorrow the price will be different from today's, partially due to the natural evolution of the stock price, and of course due to the trading we attempted today. The process will restart tomorrow.

When N is large, the trade might take several days, during which period the price of the stock may evolve substantially, quite likely against our interest. When all is said and done, instead of receiving a cash flow of $\mathbf{P_0 N}$, we will instead receive $\mathbf{P_0 N - M}$, for some $\mathbf{M > 0}$. The quantity is called the "market impact" of our trade – we want to minimize M [note: there are other, routine, smaller magnitude trading costs incurred when one trades. These are smaller and unavoidable, and are not the subject of our discussion].

It seems prudent, therefore, when N is large, to space our trade into more easily digestible bites. If we control the size of these partial trades, we may be able to minimize the market impact. In order to quantify this statement precisely, we need two ingredients:

- A precise numerical model of how market impact arises
- An algorithm that minimizes market impact, given the model we have constructed

Some of the background for the models we will describe can be found in papers and on "folklore" ideas. The algorithmic methodology we will use is based on Dynamic Programming; you can find good descriptions of this in many books on Mathematical Programming and on the web.

A deterministic model

The models we will use are based on the assumption that market impact, on a per-share basis, is a reflection of the lack of liquidity of the stock. Part of this can be observed directly: if the stock is one that is very actively traded, then it should be easy to trade and the market impact should be relatively minor. On the other hand, if the stock is very volatile, then (unless we are lucky) when we attempt our trade many other people are also likely to be attempting trades in the same direction. So, we can construct an admittedly simple model where

Per-share market impact = α =
volatility/(average daily trading volume) **(1)**

Here, "volatility" is the (observed) long-term standard deviation of the return of the stock. Both volatility and ave. daily trading volume are easily computed from time series data. The import of the above formula is that, if at time **i** we try to sell $\mathbf{n_i}$ shares, and if the nominal price of the stock happens to be $\mathbf{P_i}$, then the price will be changed to

$$\mathbf{P_{i+1} = P_i - \alpha\, n_i} \qquad \mathbf{(2)}$$

and so our cash flow at time **i** will equal:

$$\mathbf{P_{i+1}\, n_i = (P_i - \alpha\, n_i)\, n_i} \qquad \mathbf{(2')}$$

This expression, and the analysis that follows, are of course based on the assumption that we are *selling* shares. In the opposite case the sign in (2) and (2′) has to be reversed (and also in the expressions below).

Suppose that we set ourselves the goal of completing our sale of N shares, in at most T days. Here T would be relatively small, say T = 10. We will therefore trade on days 0, 1, 2, …, T – 1. On day i, we will sell some number $\mathbf{n_i}$ of shares. We choose this value. When carrying out this trade, the nominal price of the stock depends on our trade amounts in days 0, 1, …, i-1, (i.e., on the amounts $\mathbf{n_0}$, $\mathbf{n_1}$, … , $\mathbf{n_{i-1}}$) through formula (2). This will dictate the cash flow we receive on day i, and furthermore, through (2) it will dictate the price of the stock on day i+1.

To put it in a different light, our total cash flow (on days 0, …, T-1), for a given choice of the numbers $\mathbf{n_0}$, $\mathbf{n_1}$, … , $\mathbf{n_{T-1}}$, equals (where $\mathbf{P_0}$ = price of the stock at time 0):

$$\mathbf{(P_0 - \alpha n_0)n_0 + (P_0 - \alpha n_0 - \alpha n_1)n_1 + \ldots}$$
$$\mathbf{+ (P_0 - \alpha n_0 - \alpha n_1 - \ldots - \alpha n_{T-1})n_{T-1}} \qquad \mathbf{(3)}$$

Clearly, we must have that

$$n_0 + n_1 + \ldots + n_{T-1} = N, \qquad (4)$$

since N was the total number of shares we wanted to trade. So (3) can be rewritten as:

$$P_0 N - \alpha [n_0^2 + (n_0 + n_1) n_1 + \ldots + (n_0 + n_1 + n_{T-1}) n_{T-1}] \qquad (5)$$

The first term in (5) is the cash flow we would have received had there been no market impact. The **second term** in **(5)** is the market impact, and this we seek to **minimize** by choosing the n_i (the daily trade amounts) appropriately. Our constraints are (4) and the fact that the n_i have to be nonnegative.

In addition, the n_i have to be integral, and there may be a "minimum trade amount" requirement: either $n_i = 0$, or $n_i \geq \mu$, where $\mu > 0$ is a known value. We will ignore this last requirement.

If we forgo integrality, then we simply seek to minimize the expression inside the square brackets in (5). One can show that the optimal solution is simply to set:

$$n_i = N/T, \quad \text{for } i = 0, \ldots, T-1,$$

i.e. the optimal strategy is simply to split the trade evenly! Even in the presence of integrality constraints, this even-split strategy can be used: N is large and T is relatively small, so simply rounding N/T up or down to an integer will not result in large error.

A different deterministic model

An issue with the model given by (2) is that the drop in price depends on the number of shares traded, but not in the actual price itself. One might imagine that, e.g. if P_i were small, then the drop in price itself should be smaller (the stock price is more "sluggish") for a given number n_i of sold shares, and if P_i is high then the drop should be higher. This leads to a model of the form

$$P_{i+1} = P_i - \alpha P_i n_i = P_i(1 - \alpha n_i) \qquad (6)$$

as opposed to (2). In this model, the price at time i+1 will equal

$$P_i = P_0(1 - \alpha n_0)(1 - \alpha n_1)\ldots(1 - \alpha n_{i-1}) \qquad (7)$$

Now, the problem becomes: choose $n_0, n_1, \ldots, n_{T-1}$, nonnegative, integral, with $n_0 + n_1 + \ldots + n_{T-1} = N$, such that

$$P_0 N - P_1 n_0 - P_2 n_1 \ldots - P_T n_{T-1} \qquad (8)$$

is **minimized** (where the P_i are given by (7)) or, what is equivalent, such that

$$P_1 n_0 + P_2 n_1 \ldots + P_T n_{T-1} \qquad (9)$$

is **maximized**. Note that this expression is simply the cash flow that we will receive. To tackle this problem we can use dynamic programming.

The dynamic programming approach hinges on the notion of *states*. Here, a state is a triple **(K, P, i)**, where $0 \le K \le N$, $P \ge 0$ and $0 \le i \le T-1$. These states have the following interpretation: at day **i**, we find ourselves in state **(K, P, i)**, if we still have **K** shares left to trade and the current stock price is **P**. For each state **(K, P, i)**, define

V(K, P, i) =
Maximum cash flow that we can obtain, by selling **K** shares on days **i**, …, **T-1**, if on day **i** we start in state **(K, P, i)**

Why are we interested in these values? Because our original problem, that of maximizing the expression in (9) can simply be rephrased as computing $V(N, P_0, 0)$. The dynamic programming approach embeds this problem into that of computing every parameter **V(K, P, i)**.

Note that there are many states – because on any day **i** the price of the stock could take many possible values between the initial price P_0 and 0. In effect, we would need to enumerate all possible sequences of values n_i that add up to **N**.

What 'saves the day' is the relatively simple nature of equations (6) and (7). Note that:

$$V(K, P, T-1) = P[1 - \alpha K]K, \qquad (10)$$

since by time T we must sell all our shares. For i < T-1, we have that

$$V(K, P, i) = \qquad (11)$$
$$= \max\{ P[1 - \alpha n_i]n_i + V(K - n_i, P[1 - \alpha n_i], i+1) \mid 0 \le n_i \le K \}$$

In this expression, the blue term is our immediate cash flow should we sell n_i shares at time **i**. The red term is what happens starting at time **i+1**: we have $K - n_i$ shares, and the price has

dropped to $P[1 - \alpha n_i]$, and, of course, we should proceed optimally from time $i+1$ onwards. We need n_i to choose so as to maximize the expression in the right-hand side of (11).

But before we proceed, notice that for any K, P and i,

$$V(K, P, i) = P\, V(K, 1.0, i) \qquad (12)$$

Here, to repeat the definition, V(K, 1.0, i) is the maximum cash flow that can be obtained by selling K shares in days i through T-1, assuming that the price on day i equals 1.0. Why is (12) true? That is because (6) is multiplicative in -- if we double P_i, then P_{i+1} doubles as well, etc. [In order to obtain a rigorous proof of (12), note that (12) holds at i = T-1 (because of equation (10)) and it also holds for i = T-2, and T-3, etc., because we can apply equation (11), which is also multiplicative in P – this is a proof by induction on i].

Given that (12) holds, we can streamline (11). Let's use the notation, for any K and i:

$$W(K, i) = V(K, 1.0, i).$$

Then

$$W(K, T-1) = [1 - \alpha K]K, \qquad (10')$$

and for i < T-1:

$$W(K, i) = \qquad (11')$$
$$= \max\{[1 - \alpha n_i]n_i + [1 - \alpha n_i]W(K - n_i, i+1) \mid 0 \le n_i \le K\}$$

Thus, we can use dynamic programming to compute all values W(K,i), and then, using (12), set $V(N, P_0, 0) = P_0\, W(N, 0)$.

A more general class of models

The impact model discussed above is frequently modified as follows. Rather than using (6) to represent the price change, we use

$$P_{i+1} = P_i(1 - \alpha n_i^{\pi}) \qquad (6')$$

Here, $0 \le \pi$ is a modeling parameter. Using $\pi < 1$ will slow down the impact.

The random setting

In the random setting, instead of equation (6) we have

$$P_{i+1} = P_i (1 - \alpha n_i + \varepsilon_i) \qquad (6)$$

where ε_i is a "reasonable" random variable (for example, log-normal).

We are interested in handling the case where ε_i does not have zero mean. This enables us to model the independent process (separate from market impact) by which the stock price is changing.

Here we will discuss a numerical approach – we will rely again on dynamic programming. We will use the same states as above, but now we define, for each state **(K, P, i)**,

V(K, P, i) =
Maximum ***expected*** cash flow that we can obtain, by selling **K** shares on days **i**, … **T-1**, if on day **i** we start in state **(K, P, i)**

We have

$$V(K, P, T-1) = P[1 - \alpha K + E(\varepsilon_{T-1})]K,$$

(where **E** denotes expectation) again since by time T we must sell all our shares. We can then proceed using backwards recursion as above. Again we have to handle the fact that the state space is very large – in principle, it could be infinitely large.

Question: how do we adapt the deterministic algorithm to this setting?