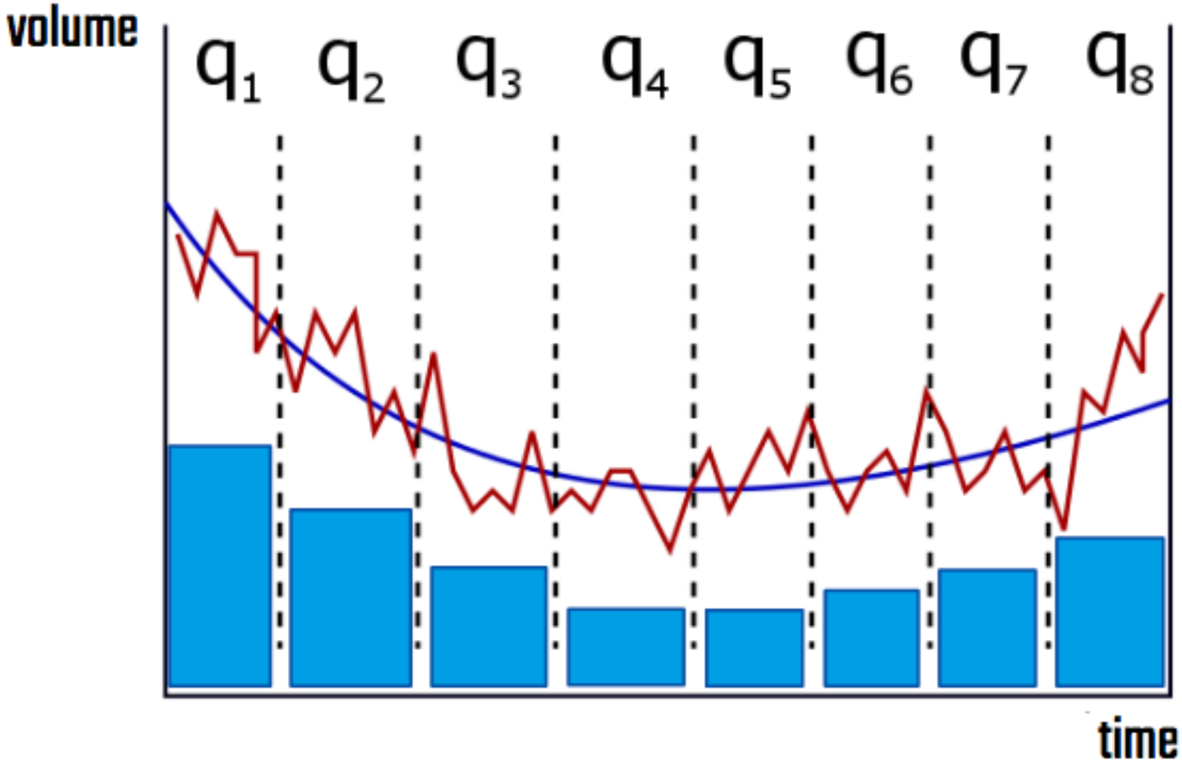
# Algorithmic Trading


**Dr. Javier Sabio González**
*Head of Advanced Analytics & Algorithmic Trading, BBVA C&IB*

# Topics

1. Market Macrostructure
2. Market Microstructure
3. Algorithmic Trading Fundamentals
4. **Algorithmic Execution**
5. Algorithmic Market-Making
6. Algorithmic Investment
7. The Future of Algorithmic Trading

# 4. Algorithmic Execution



# Algorithmic execution

**Aim**: execute (buy or sell) a (large) volume in a financial instrument, typically in markets based on a limit order book trading mechanism

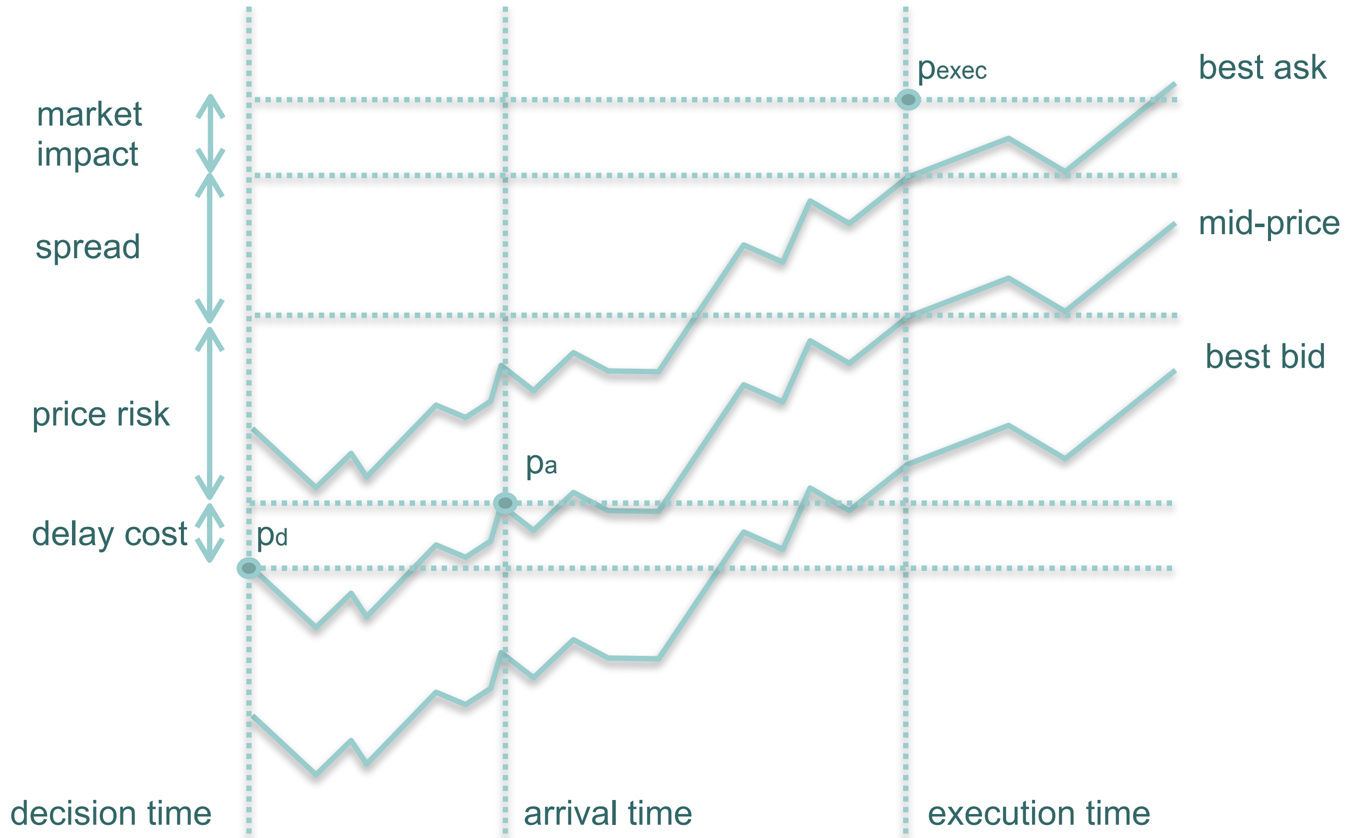
Why is it important?

- **To make trading strategies profitable**. If execution costs are high, trading strategies (algorithmic or not) that are seemingly profitable might really not be it. This affects:
  - **Banks** that are hedging their market risk
  - Investors like **mutual and hedge funds** that need to get returns from their portfolios
- **As a source of revenues:** banks and brokers usually perform execution of block trades on behalf of clients, who rely on their expertise and low fees access to markets.
  - In a competitive market, these companies need to reduce execution costs to attract clients

# Transaction costs for trading algorithms

- **Commissions**: payments made to broker-dealers for executing trades, order routing, risk management, etc
- **Fees**: payments to the exchange per order executed, for clearing, settlement, etc
- **Taxes**: payments to the Government based on realised earnings
- **Rebates**: some exchanges reduce or exempt fees depending on the type of orders. In a market-taker model, liquidity providing orders get the rebate. In a taker-maker model, liquidity consuming orders get it.
- **Spreads**: a cost for liquidity taking orders, is the difference between best offer and best bid. It compensates market makers for inventory and information asymmetry risks.
- **Delay cost**: costs due to the difference between the price when a decision is taken and the price when the order arrives to the market, e.g. due to latencies
- **Market impact**: adverse movements in the price caused by a particular order
- **Timing risk**: costs due to the uncertainty in market conditions during the execution of an order, like price volatility and liquidity changes
- **Opportunity cost**: forgone profit or avoided loss of not being able to transact the entire order in the intended time window

# Transaction costs for trading algorithms



## Market impact

Adverse movements in the price caused by a particular order

*Mathematically*, market impact is the difference between the price trajectory of the instrument with the order and what the price trajectory would have been had the order not been released to the market. However, this is impossible to measure in practice.

It is typically decomposed in two effects:

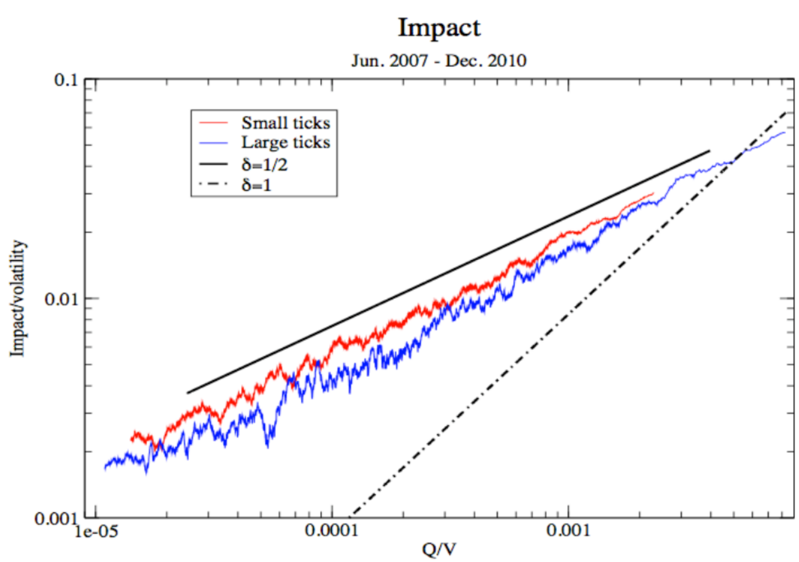
- **Temporary market impact**: when the agent wishing to buy or sell in the market has insufficient counterparties to complete the order at best current prices, so a premium must be paid to execute it whole.
- **Permanent market impact**: information content of the order that is transmitted to the market, which might react readjusting its estimation of fair price

# A model for temporary market impact

A classical model that captures the main dependencies of market impact is that of Grinold and Kahn (1994), aka “the square root market impact model”:

$$\Delta P = \text{Spread cost} + \alpha\,\sigma\sqrt{\frac{Q}{V}}$$

where σ is the average daily volatility, V is average daily volume, Q is the order size, Spread cost is the average bid-offer spread and α is a calibrated constant.




Log-log plot volatility price impact vs % volume. Source: Toth et al

# A model for temporary market impact

Notice:

- The square root formula fits reasonably well empirical data for two to three orders of magnitude in order sizes
- The model can be generalised to fit a power law to the data, and to include intraday information: intraday bid-offer spreads, volatility, predicted volume in a given time window, etc. Calibration is however more complicated.
- This model is used as standard in many theoretical papers and in vendors like Bloomberg
- A linear market impact model is less realistic, but also quite standard in theoretical papers

Applications:

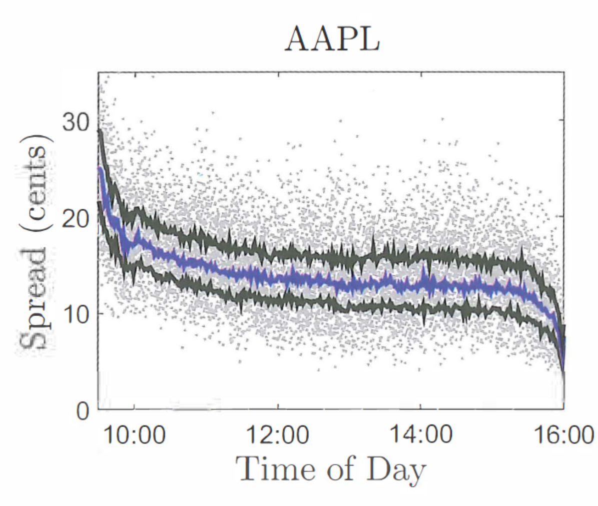
- Backtesting of algorithms: estimation of market impact of algorithmic orders
- Pre-trade analysis: estimation of cost of execution of an algorithm before launching it
- Derivation of execution strategies: to be plugged in objective functions to mathematically derive optimal execution strategies

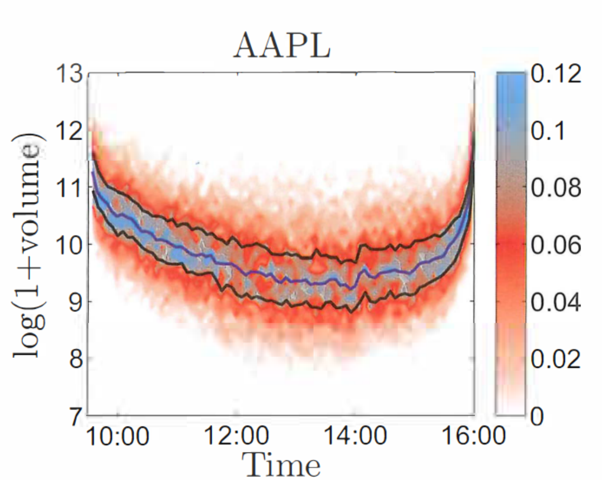
## Timing (price) risk

Costs (but also profits) due to market volatility during the execution of an order

Typically we have

- Price volatility: during the execution window, prices can move away from initial prices, potentially increasing (but also decreasing) the cost of execution with respect to executing at the beginning (even with a market impact)
- Liquidity changes: market impact depends on liquidity (bid-offer spread, for instance), and the latter can change quickly intraday responding to new information coming from external news, block trade signals, etc
- Traded volume changes: volume traded (market orders, aggressive limits) is also volatile, changing probabilities of filling passive orders



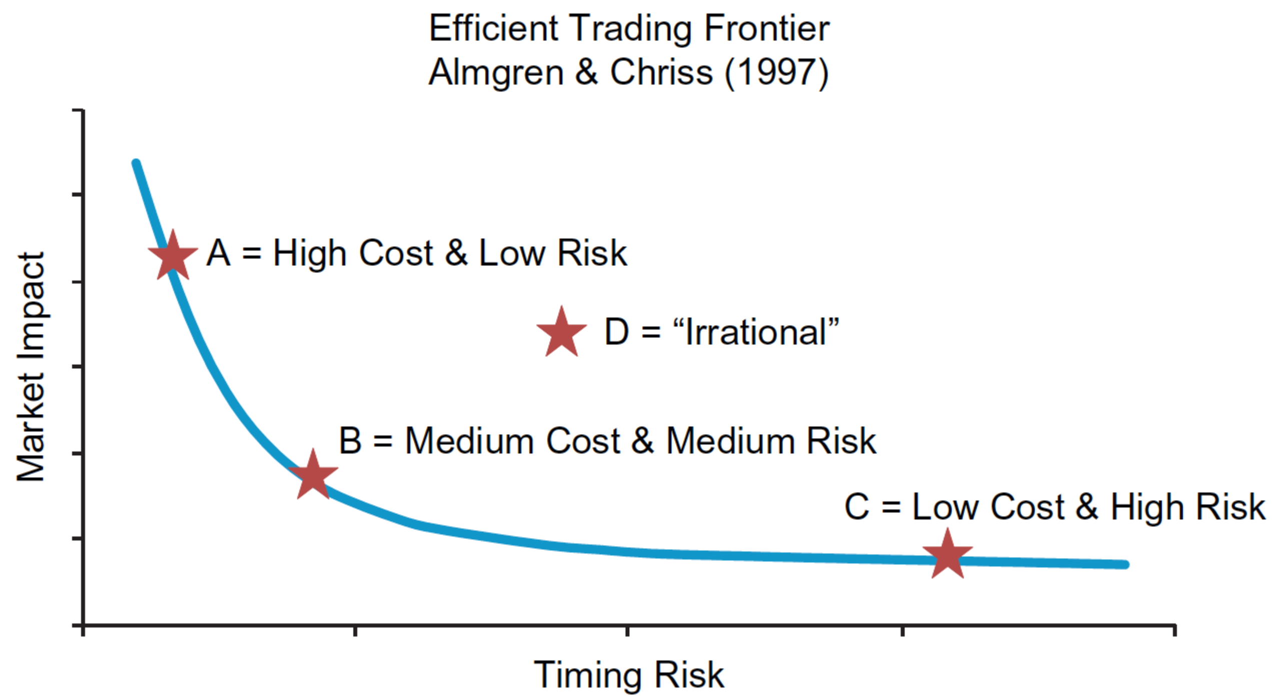




Source: Cartea et al (2015)

## The algorithmic execution trade-off (the "trader's dilemma")

A good execution must balance two opposing forces:

- Market impact: which is potentially larger the shorter the window of execution
- Timing risk: which is potentially larger the longer the window of execution



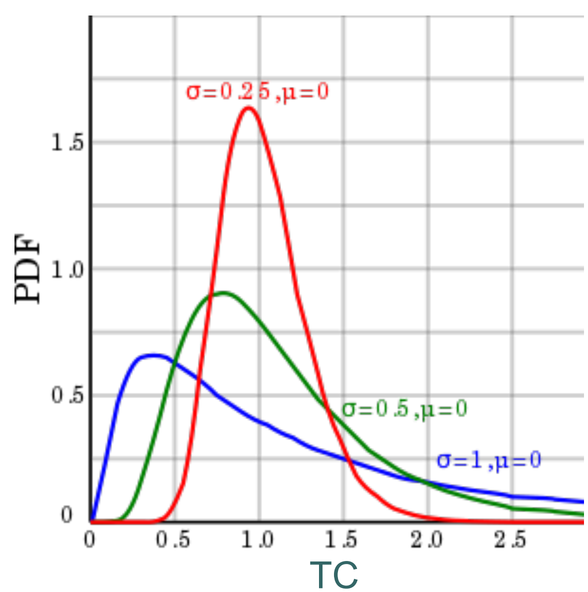

Source: Kissell (2014)

# What is a good execution?

buy/sell, respectively

Essentially, one that minimises transaction costs: $$TC = \pm \sum_i p_i^{exec} q_i + fees$$

- However, transaction costs are not known until the end of execution, since they depend on market conditions
- Therefore an optimal execution tries to minimise expected transaction costs
- But an strategy that minimises expected transaction costs might have scenarios where execution costs are quite high, due to timing risk.
- A good execution therefore requires to be defined at the level of its probability of executing at a given cost: not only means are compared, also standard deviations, high percentiles (tails), etc



# <u>What is a good execution?</u>

In order to evaluate an execution, it is also convenient to compare with <u>benchmarks</u>:

- Simple execution strategies, for example:
  - simply slicing the order in equal parts over the maximum time window admisible for the trader (“Time Weighted Average Price”, TWAP)
  - trading a fix proportion of market volume (“Percentage of Volume”, POV) until finished
- Market-wide benchmarks, for example:
  - Market mid-price when the execution started (“Implementation Shortfall”)
  - Market mid-price when the execution finished (“Target at Close”)
  - Market Volume Weighted Average Price (VWAP) over the execution window

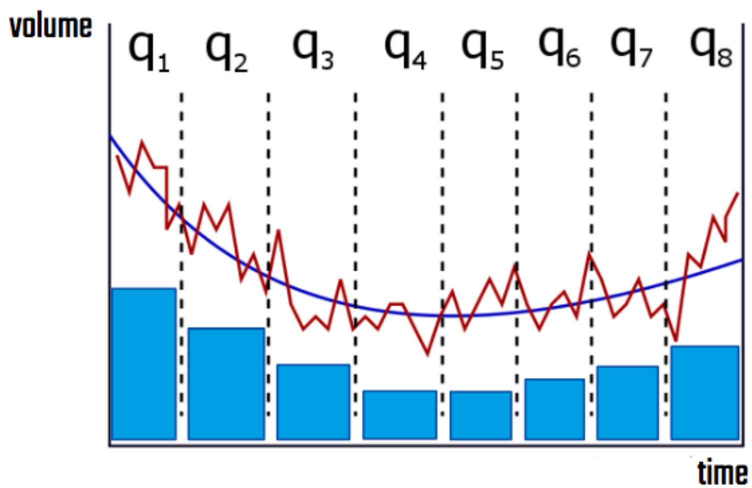
<u>Notice</u>:

- Benchmarks are important to assess the relative performance of our strategy: if we have a very sophisticated strategy that does no better than e.g. TWAP, is not worthy.
- They are also typically used by other agents (e.g. clients) to assess the performance of our strategy, or to determine a pre-agreed price for an execution performed on their behalf
- Careful analysis is required for strategies that don’t execute the whole quantity: in order to assess their quality, an explicit “opportunity cost” should be introduced (for instance charging the remaining quantity at the cost of a market order at close)

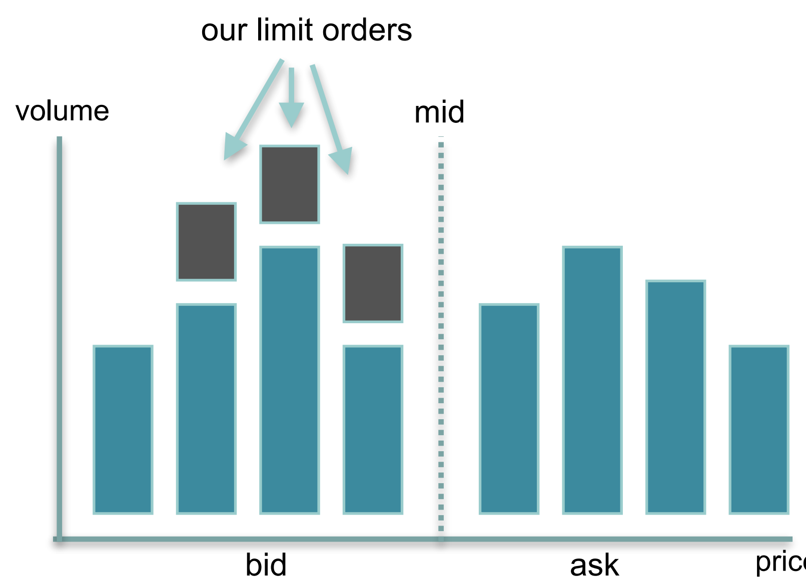
# Designing execution algorithms

We divide the optimisation problem in two sub-parts (not necessarily optimal, but convenient):

- Optimal execution / execution schedule / macro-strategy: slice optimally big orders into smaller ones intra-day to minimise market impact / timing risk and/or to track a benchmark
- Optimal placement / execution tactic / micro-strategy: optimally place the smaller orders in the Limit Order Book to be executed at best price / minimum cost in the time window selected by the execution schedule




1. Divide day in time windows (e.g. 5 minutes) and find optimal sub-orders




2. Place orders optimally in the Limit Order Book, to be executed during the selected time window

# Minimising expected costs of execution: Bertsimas & Lo

Optimal acquisition problem: find the optimal *n* slices of size $q_i$ to buy an order of size *Q* over time windows *T/n,* where *T* is the maximum time allowed for the purchase (e.g. if it is the whole trading day, in BME that would be *T* = 8 hours)

(Optimal liquidation problem is symmetrical, but selling the size Q)

Bertsimas & Lo (1998): the optimal slices minimise the total expected cost

$$min_{q_0,\dots,q_{n-1}} E_0[\sum_{i=0}^{n-1} p_i^{exec} q_i]$$

$$\text{subject to} \sum_{i=0}^{n-1} q_i = Q \ , \ q_i \geq 0$$

Assumptions:

- Prices follow a random walk model: $$p_i = p_{i-1} + \epsilon_i$$ $$\epsilon_i \sim N(0, \sigma^2)$$
- Temporary market impact is a linear function: $p_i^{exec} = p_i + kq_i$

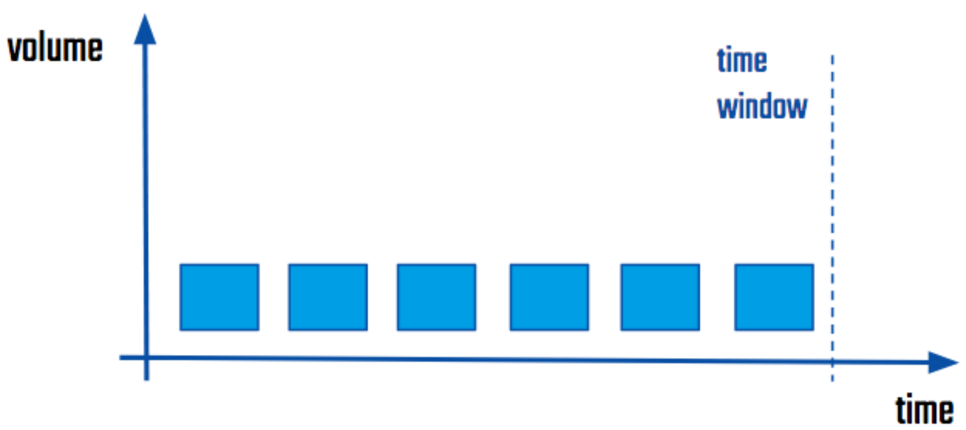
# Minimising expected costs of execution: Bertsimas & Lo

Solution: optimal slices are of equal size:

$$q_0 = \ldots = q_{n-1} = \frac{Q}{n} \qquad\qquad Q_{j,n-1} = \sum_{i=j}^{n-1} q_i = \frac{Q(n-j)}{n}$$

Remarks:

- Stochastic Optimal Control Problem: typically solved using Dynamic Programming
- Solution is a "Time Weighted Average Price" (TWAP) strategy
- It is deterministic, and does not depend on the market impact or the price dynamics



# Including risk aversion and the trader's dilemma: Almgren & Chriss

Almgren & Chriss (1999, 2001): the optimal slices minimise the total expected cost, for a given tolerance of total cost variance (which is directly related to timing risk)

$$min_{q_0,\ldots,q_{n-1}} \left( E_0[\sum_{i=0}^{n-1} p_i^{exec} q_i] + \lambda Var_0[\sum_{i=0}^{n-1} p_i^{exec} q_i] \right)$$

$$\text{subject to} \sum_{i=0}^{n-1} q_i = Q \,,\; q_i \geq 0$$

Notice:

- As in Bertsimas & Lo, prices follow a random walk model with linear temporary market impact:

$$p_i = p_{i-1} + \epsilon_i \qquad p_i^{exec} = p_i + kq_i$$
$$\epsilon_i \sim N(0, \sigma^2)$$

- It can also include a linear permanent market impact (we won't consider it here)
- λ is a Lagrange multiplier that can be interpreted as a risk aversion parameter, since the problem can be mapped to maximising an exponential utility function

# Including risk aversion and the trader's dilemma: Almgren & Chriss

Solution: optimal slices are of size

$$q_j = Q_{j,n-1} - Q_{j+1,n-1}\ ,\ \ j = 0, \ldots, n-1$$

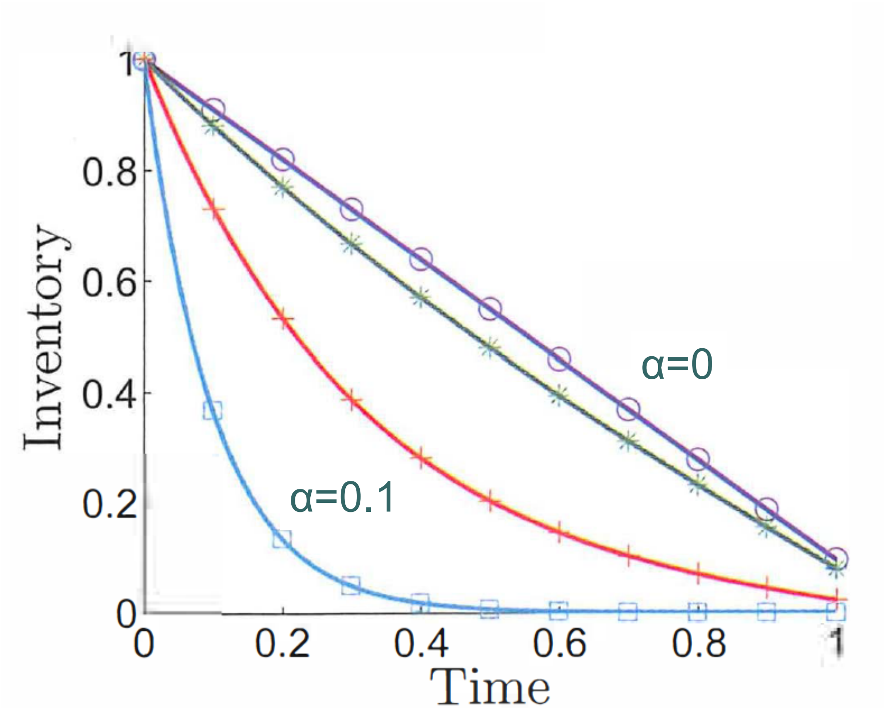
$$Q_{j,n-1} = Q\frac{\sinh(\alpha(T - t_j))}{\sinh(\alpha T)}$$

$$t_j = \frac{T}{n} j\ ,\quad \alpha^2 \simeq \lambda \frac{\sigma^2}{k}$$

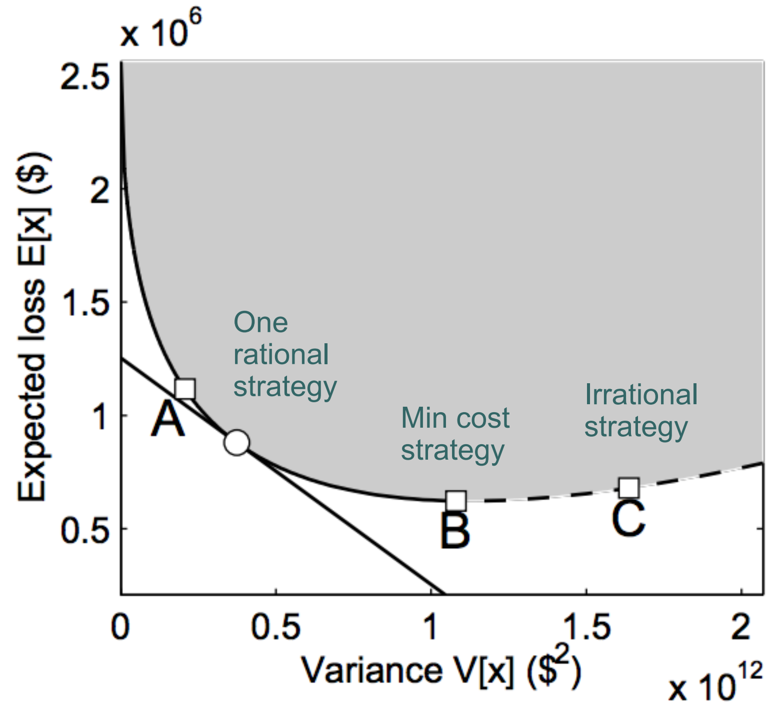
Remarks:

- Again, is the solution to a Stochastic Optimal Control Theory problem
- Deterministic solution. Depends on price volatility, market impact and risk aversion
- Interesting limits:
  - For very small volatility and/or risk aversion, or for very large market impact, the solution tends to a TWAP, since market impact dominates timing risk:
  $$q_0 = \ldots = q_{n-1} = \frac{Q}{n}$$
  - For very large volatility and or risk aversion, or very small market impact, the order is executed full at the first time window:
  $$q_0 = Q$$
  $$q_j = 0\ ,\ j > 0$$

# Including risk aversion and the trader's dilemma: Almgren & Chriss




Example of execution schedules for different values of α
Source: Cartea et al (2015)




Efficient frontier of optimal execution
Source: Almgren & Chriss (2000)

## Algorithms tracking benchmarks

Execution algorithms are measured against benchmarks:

- To assess their performance
- To reference the cost of an execution when the client delegates it to a specialist

There are execution algorithms that try to replicate as close as possible a given benchmark:

- As we have seen, there is a tradeoff between cost of execution and its variance (risk)
- If beating the benchmark has less upsides than missing it (the execution cost is going to be associated to the benchmark, the client will judge the performance using the benchmark, …), then it makes sense to try to stay close to the benchmark if that reduces the variance with respect to it

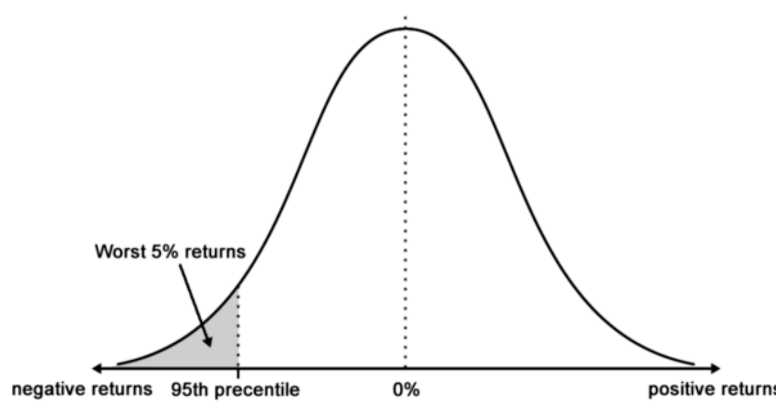
What are typical benchmarks? Implementation Shortfall (IS), Volume Weighted Average Price (VWAP), Time Weighted Average Price (TWAP), Percentage of Volume (PoV)

# Pricing an execution using a benchmark

Typical situation: a client delegates the execution of a block order to a specialised broker/dealer. A benchmark is set to reference the price that the broker-dealer will charge the client for the execution.

How does the broker-dealer calculate the price?

- Use an execution algorithm to handle the execution, typically one specialised in matching or beating the chosen benchmark
- Calculate historically an histogram of the performance of the algorithm in similar operations against the benchmark (in basis points)
- Select a percentile of the loss tail of the histogram enough to make the business profitable on average: aggressive is around 50, something between 5 - 35 typical
- Price = benchmark + percentile



## Implementation Shortfall (IS)

Implementation Shortfall benchmark: cost of execution against executing the whole order at the arrival price

- For algorithms that complete the whole order in the maximum time allowed:

$$IS = \sum_i q_i p_i^{exec} + fees - Qp_0 = Q(p_{avg} - p_0) + fees$$

- For algorithms that **don't** complete the whole order in the maximum time allowed:

$$IS = \underbrace{\sum_i q_i (p_i^{exec} - p_0)}_{\text{only the executed part}} + \underbrace{(Q - \sum_i q_i)(p_T - p_0)}_{\text{opportunity cost}} + fees$$

- The profit & loss (p&l) of the executed part is usually expressed as a relative measure in basis points:

$$P\&L_{IS} = side * \frac{p_0 - p_{avg}}{p_0} 10^4$$

$$side = +1(buy), -1(sell)$$

# Implementation Shortfall Algorithm

Algorithm that tries to minimise the Implementation Shortfall for a given level of risk aversion/tolerance to the deviation wrt to the benchmark

$$min_{q_0,\dots,q_{n-1}} \left(E_0[IS] + \lambda Var_0[IS]\right)$$

$$\text{subject to} \sum_{i=0}^{n-1} q_i = Q \,, \; q_i \geq 0$$

If, for simplicity, consider fees as deterministic, then:

$$E_0[IS] = E_0[\sum_{i=0}^{n-1} p_i^{exec} q_i] - Qp_0 + fees$$

$$Var_0[IS] = Var_0[\sum_{i=0}^{n-1} p_i^{exec} q_i]$$

This is the Almgren - Chriss execution algorithm!! The AC algorithm automatically tracks the IS benchmark. This is due to the arrival price being deterministic at $t_0$

# Implementation Shortfall Algorithm

## Example: HSBC’s IS for FX

**Instrument**

Spot/Forward

**Currency Pair**

€$¥

**Direction**

Trade direction (Buy/Sell Currency).

**Order Notional**

Total quantity to trade in Currency.

**Value Date**

Instrument value date, defaults to the Spot value date of the selected Pair.

**Start time**

Execution of the order commences immediately by default but can be postponed until this time.

**End time**

Execution of the order ends by default at date roll time of the selected Pair (typically 5 p.m. EST) but an earlier End Time can be specified. The Algorithm will continue working the order with the aim to complete by the specified End Time.

**Limit price**

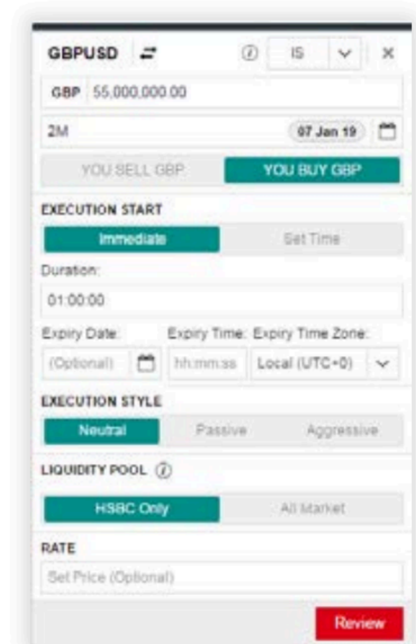
The Algorithm will consume liquidity at prices no worse than the Limit Price.

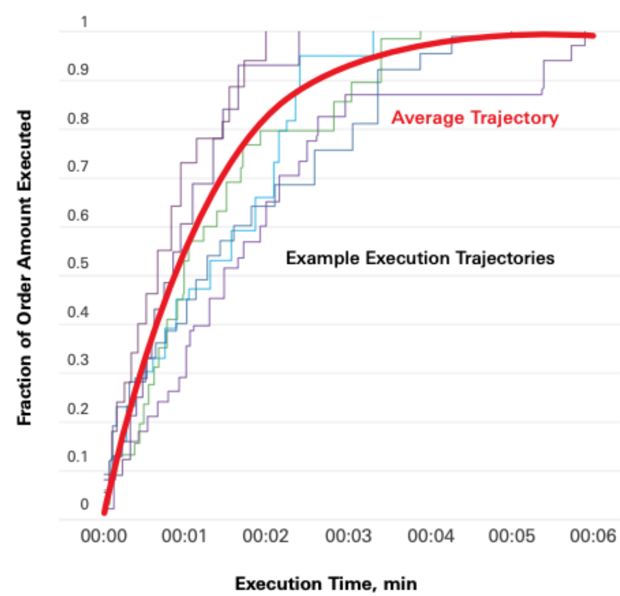
**Execution Style**

Passive, Neutral or Aggressive, depending on the client’s level of market volatility risk aversion (see next page for details).

**Liquidity Pool**

There is a choice to either use market liquidity, HSBC liquidity or both. Market liquidity sources are detailed in the Appendix. For client algorithmic execution, HSBC liquidity consists of top-tier streaming for aggressive matching and Internal Exchange for both passive and aggressive matching.



# Time Weighted Average Price (TWAP)

Time Weighted Average Price benchmark: average market price over a time window (calculated using all the trades within the time window)

$$twap = \frac{1}{n}\sum_{i=0}^{n-1} p_i^{twap}$$

$$p_i^{twap} = \frac{1}{N_{trades(i)}} \sum_{j \in trades(i)} p_j^{avg}$$

As a performance benchmark we use:

$$P\&L_{twap} = side * \frac{p_{twap} - p_{avg}}{p_{twap}} 10^4$$

$$side = +1(buy), -1(sell)$$

As a cost measure we use:

$$C_{twap} = Q(p_{avg} - p_{twap}) + fees$$

## Time Weighted Average Price Algorithm

Algorithm that tries to minimise the TWAP cost for a given level of risk aversion/ tolerance to the deviation with respect to the benchmark

$$min_{q_0,\ldots,q_{n-1}} \left(E_0[C_{twap}] + \lambda Var_0[C_{twap}]\right)$$

$$\text{subject to} \sum_{i=0}^{n-1} q_i = Q \,,\ q_i \geq 0$$

Solution: execute an amount proportional to the time window. Since in our case, time windows are of equal size, then:

$$q_0 = \ldots = q_{n-1} = \frac{Q}{n}$$

# Time Weighted Average Price Algorithm

## Example: HSBC's TWAP for FX

| Instrument | Currency Pair | Direction | Order Notional |
|---|---|---|---|
| Spot/Forward | €$¥ | Trade direction (Buy/Sell Currency). | Total quantity to trade in Currency. |

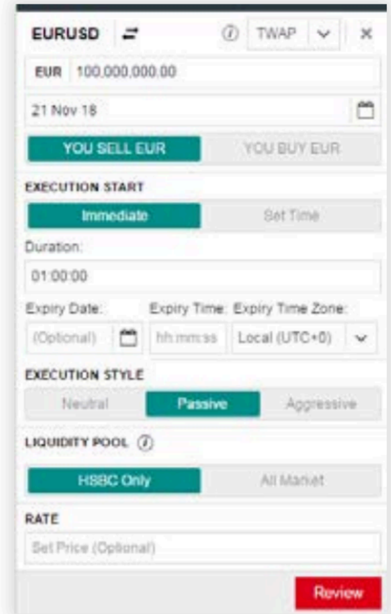
| Value Date | Start time | End time |
|---|---|---|
| Instrument value date, defaults to the Spot value date of the selected Pair. | Execution of the order commences immediately by default but can be postponed until this time. | Execution of the order ends by default at date roll time of the selected Pair (typically 5 p.m. EST) but an earlier End Time can be specified. The Algorithm will continue working the order with the aim to complete by the specified End Time. |

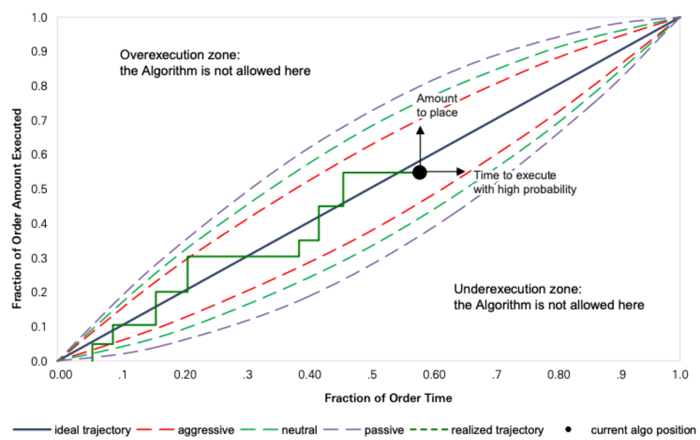
| Limit price | Execution Style | Liquidity Pool |
|---|---|---|
| The Algorithm will consume liquidity at prices no worse than the Limit Price. | Passive, Neutral or Aggressive, depending on the client's level of market volatility risk aversion (see next page for details). | There is a choice to either use market liquidity, HSBC liquidity or both. Market liquidity sources are detailed in the Appendix. For client algorithmic execution, HSBC liquidity consists of top-tier streaming for aggressive matching and Internal Exchange for both passive and aggressive matching. |

When the **Passive Style** selected, the Algorithm will tolerate market volatility risk and larger deviations from the "ideal" trajectory.

When the **Neutral Style** is selected, the Algorithm will be moderately risk-averse and tolerate deviations from the "ideal" trajectory in a balanced manner.

When the **Aggressive Style** is selected, the Algorithm will be highly risk-averse and try to stay very close to the "ideal" trajectory.



# Volume Weighted Average Price (VWAP)

Volume Weighted Average Price benchmark: average prices of market traders weighted by the volume of each trade

$$vwap = \frac{\sum_{i=0}^{n-1} v_i p_i^{vwap}}{\sum_{i=0}^{n-1} v_i}$$

$$p_i^{vwap} = \frac{\sum_{j \in trades(i)} v_j p_j^{avg}}{\sum_{j \in trades(i)} v_j}$$

As a performance benchmark we use:

$$P\&L_{vwap} = side * \frac{p_{vwap} - p_{avg}}{p_{vwap}} 10^4$$

$$side = +1(buy), -1(sell)$$

As a cost measure we use:

$$C_{vwap} = Q(p_{avg} - p_{vwap}) + fees$$

# Volume Weighted Average Price (VWAP)

Remarks:

- VWAP is one of the most common benchmarks for broker-dealers that execute orders on behalf of clients
- It is not without issues as an execution benchmark: since our own trades are part of the benchmark, optimal strategies that execute large enough quantities could have behaviours which are tantamount to market abuse

$$vwap = \frac{\sum_{i=0}^{n-1}(v_i p_i^{vwap} + |q_i| p_i^{exec})}{\sum_{i=0}^{n-1}(v_i + |q_i|)}$$

## Volume Weighted Average Price Algorithm

Busseti & Boyd (2015): They derive algorithms that track the VWAP benchmark under the following conditions

- Small volumes compared to the market volumes: this way we can exclude our own trades from the VWAP benchmark
- A linear market impact model. However, in our case we will consider that volumes are small enough to neglect their market impact
- Price process is a drift-less Geometric Random Walk
- Some technical conditions on the market volume distribution:
  - Market volumes at a specific time window are independent of total market volumes over the trading period $$E[\frac{v_i}{\sum_j v_j}] \equiv E[\frac{v_i}{V}] \simeq E[v_i]E[\frac{1}{V}]$$
  - No convexity adjustment for the inverse of total market volume

$$E[\frac{1}{V}] \simeq \frac{1}{E[V]}$$

## **Volume Weighted Average Price Algorithm**

Objective function: Busseti & Boyd use a relative cost measure:

$$S = side * \frac{p_{avg} - p_{vwap}}{p_{vwap}}$$

Then the objective function has the usual form:

$$min_{q_0,\dots,q_{n-1}} \left(E_0[S] + \lambda Var_0[S]\right)$$
$$\text{subject to} \sum_{i=0}^{n-1} q_i = Q \,,\ q_i \geq 0$$

However under the assumption of negligible market impact, the variance term disappears, resulting in a minimum expected cost problem *à la* Bertsimas & Lo:

$$min_{q_0,\dots,q_{n-1}} E_0[S]$$
$$\text{subject to} \sum_{i=0}^{n-1} q_i = Q \,,\ q_i \geq 0$$

# Volume Weighted Average Price Algorithm

Static solution: by restricting the information set to that available at $t_0$ (i.e. the strategy does not use new information observed in the market later on), the problem can be reduced to a standard deterministic quadratic program. The solution is:

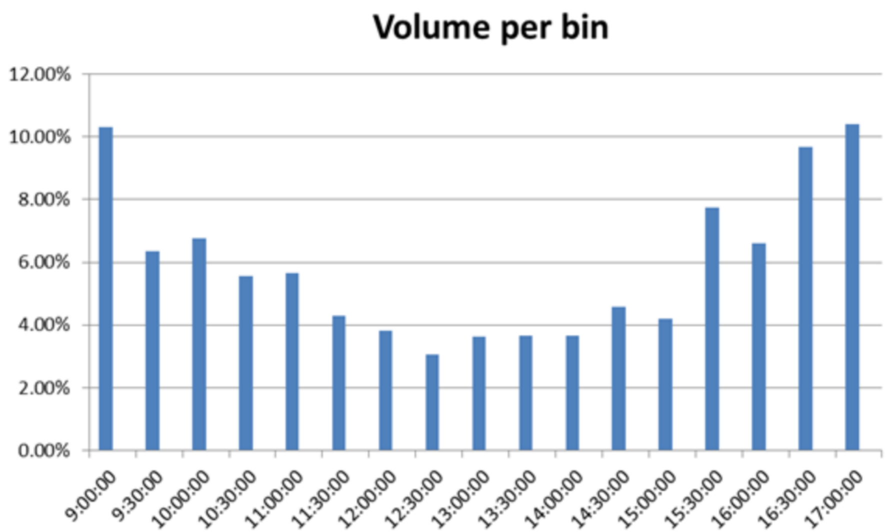
$$q_i = Q\, E_0[\frac{v_i}{V}]$$

Remarks:

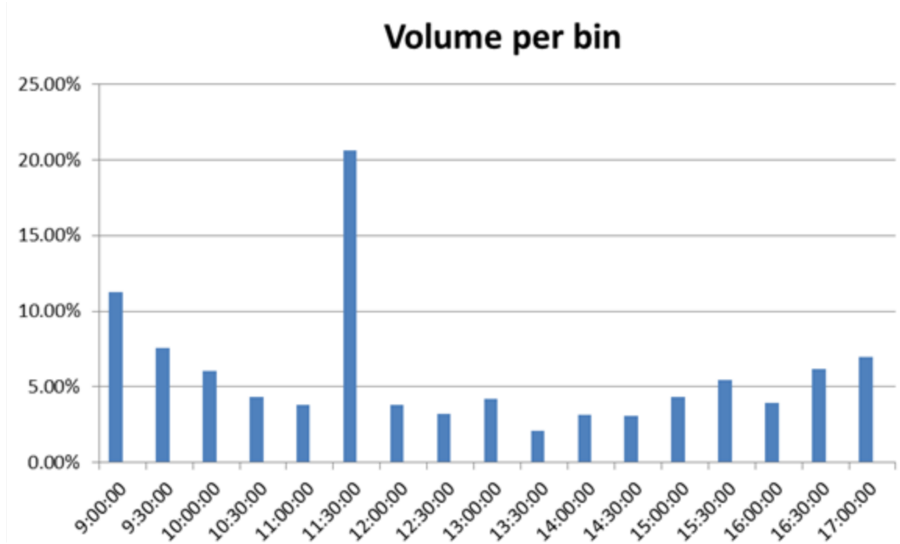
- The strategy is written in terms of a relative volume curve, the “static volume curve”
- The optimal static strategy consists on executing at each interval the same proportion of our order than the expected market one over its total over the execution window
- The static volume curve can be calculated calibration prediction models using historical data from previous days
  - It is important to take into account that volume profiles might change radically in certain special days like option and index expirations, ECB and FED days, etc
  - Outliers should be handled carefully: for instance, with medians instead of means
- Data vendors like Bloomberg and certain brokers provide their clients a static curve per instrument every day

# Volume Weighted Average Price Algorithm

Examples:




A typical day: the curve is "U" or "J" shaped




An option expiration day

# Volume Weighted Average Price Algorithm

Dynamic solution: if the strategy can use information arriving after $t_0$, the optimal solution can be found using Dynamic Programming techniques:

$$q_0 = Q\frac{E_0[v_0]}{E_0[V]}$$

$$q_i = side * \max\left(|Q|\frac{V_{0,i-1} + E_i[v_i]}{E_i[V]} - |Q_{0,i-1}|, 0\right)$$

The optimal strategy tracks the fraction of total market volume up to the current time interval:

- The quantity $V_{0,i-1} + E_i[v_i]$ represents how much market volume we expect to have been traded when the time interval *i* finishes $E_i[V]$ represents how much market volume we expect to see over the whole execution window
- Therefore the fraction of these two is the expected fraction of total market volume executed when the time interval *i* finishes
- The optimal strategy tries to keep the same proportion with respect to our order *Q*

## Volume Weighted Average Price Algorithm

Remarks:

- The strategy relies on two volume predictions per time interval *i:* total market volume over the execution window $E_i[V]$ and market volume at the interval i $E_i[v_i]$
- If these **predictions are perfect** at every interval *i*, $E_i[V] = V,\ E_i[v_i] = v_i$ **and the execution price at the time interval equals the market VWAP of this interval**, then the strategy tracks the running VWAP benchmark and therefore the VWAP over the execution window:

$$q_0 = Q\frac{E_0[v_0]}{E_0[V]} = Q\frac{v_0}{V}$$

$$q_i = side * \max\left(|Q|\frac{V_{0,i-1} + E_i[v_i]}{E_i[V]} - |Q_{0,i-1}|, 0\right) = Q\frac{V_{0,i}}{V} - Q_{0,i-1}$$

$$vwap_Q = \frac{1}{Q}\sum_i p_i^{vwap}\left(Q\frac{V_{0,i}}{V} - Q_{0,i-1}\right) = \sum_i p_i^{vwap}\left(\frac{V_{0,i}}{V} - \frac{Q_{0,i-1}}{Q}\right)$$

$$= \sum_i p_i^{vwap}\left(\frac{V_{0,i}}{V} - \frac{V_{0,i-1}}{V}\right) = \frac{1}{V}\sum_i p_i^{vwap} v_i = vwap_{market}$$

- The performance of the strategy depends critically on the **quality of the volume predictions and the execution tactic** that executes the target volume in a LOB

# Volume Weighted Average Price Algorithm

Remarks:

- Volume predictions are typically done using models (time series, machine learning) trained with historical data. Examples:
  - Busseti & Boyd: link
  - Salish et al: link
  - Bloomberg model: link
- Errors in local volume predictions can be corrected at the next time interval, when we know the exact market volume. If the price at the next time interval didn't change too much with respect to the previous one, then the strategy keeps tracking the VWAP. Therefore, **good local volume predictions are not absolutely critical to track the VWAP benchmark over the execution window**
- Errors in total volume prediction are more difficult to correct, since the total market volume is not known until the end. Therefore, **a good total volume predictor is critical** for the strategy to track the VWAP benchmark
- To track the VWAP, the execution tactic must achieve at least the VWAP at each time interval. **Therefore, a good execution tactic is also critical to track the VWAP.**

# Volume Weighted Average Price Algorithm

Remarks:

- In practice, in order to ensure the VWAP is tracked, the strategy must be enriched to exploit market opportunities at some degree (“**alpha VWAP**”), for instance:
  - trying to beat the VWAP with the execution tactic
  - predicting the final VWAP over the execution window, and identifying regions where spot prices are high (in a sell) or low (in a buy) with respect to it, to execute relatively more volume than the strategy indicates

# Volume Weighted Average Price Algorithm

Another view of VWAP: “*optimal strategy that minimises Implementation Shortfall*”

- We have seen that TWAP is the optimal strategy for a risk-neutral trader
- However this relies on having a market impact function whose parameters are time independent, like the linear: $p_i^{exec} = p_i + kq_i$
- For a market impact that is linear in the percentage of market volume, and where market volume is random, static VWAP is optimal for a risk-neutral trader
- Proof: see (Kato, 2017)
    - The market impact function reads: $p_i^{exec} = p_i + k\frac{q_i}{v_i}$
    - An the expected IS:

$$E_0[IS] = E_0[\sum_{i=0}^{n-1} p_i^{exec} q_i] = E_0[\sum_{i=0}^{n-1}(p_i q_i + k\frac{q_i^2}{v_i})]$$

- Proof: for details see paper. Highlights: 1) Use continuous time representation and integrate by parts:

$$E_0[IS] = E_0[\int_0^T (p_t q_t + k\frac{q_t^2}{v_t})dt] = p_0 Q_0 + kE_0[\int_0^T \frac{q_t^2}{v_t} dt]$$

2) Change variables from time to cumulated market volume. 3) The argument of the integral is convex so one can use Jensen’s inequality to prove:

$$E_0[IS(q_t^{stat})] \geq E_0[IS(q_t^{vwap})]$$

where $q_t^{stat}$ are the family of static strategies (use t = 0 information) and $q_t^{vwap} = Q\frac{1}{E_0[\frac{1}{v_t}]}E_0[\frac{1}{V_T}]$ is the static VWAP strategy

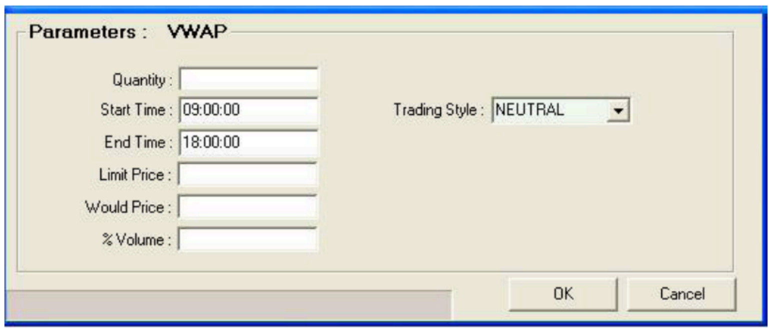
# Volume Weighted Average Price Algorithm

## Example: Natixis's VWAP

A strategy that releases waves into the markets (Primary exchange and MTFs) using stock specific historical volume profiles in order to execute the order close to the **Volume Weighted Average Price (VWAP)** over a chosen period of time, with some randomization to reduce gaming risks.

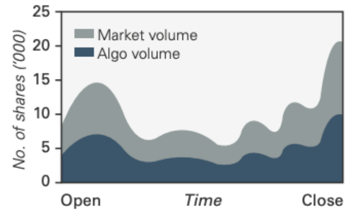
The **VWAP** algorithm aims to execute a global quantity at an average price close to the Market Volume Weighted Average Price over a defined period of time.

### VWAP Parameters




### CHARACTERISTICS

**VWAP is to be favoured in the following cases**




- To limit the market impact by executing a large quantity not too quickly.
- To execute totally an order (without guarantee if the order is limited).
- Suitable for a liquid security with a stable volume profile from one day to the next.

**Main Parameters**

| | Description |
|---|---|
| **Start / End time** | By default, start time is at reception of the order, potentially including market open if received before the opening auction. |
| **Trading style** | Conservative, neutral and aggressive. The conservative option suits a favorable price variation while Aggressive option applies better to an adverse price trend. |

**Optional Parameters**

| | Description |
|---|---|
| **%Volume** | The strategy will limit the participation rate to this maximum volume constraint. Order may not be completed. As soon as the limit constraint is respected, the strategy spreads the volume over a short period in order to reduce market impact. |
| **Price limit** | The strategy will apply the price limit to orders. Orders may not be completed. As soon as the limit constraint is respected, the strategy will spread the volume over a short period in order to reduce market impact. |
| **Would price** | The strategy will aim to complete the order if the stock trades at the Would price or better. The use of the Would option may significantly deviate the execution price from the benchmark price. |

# Percentage of Volume (PoV)

Percentage of Volume benchmark: rate of volume participation of the algorithm with respect to the market volume for a given period of time

$$POV_i = \frac{q_i}{v_i}$$

Remarks:

- It is another popular benchmark in executions, specially when the focus is on market impact and there is no need to end the order before a certain time
- Normally, the client or trader will specify a constant POV for the order to be executed, and then an algorithm will try to execute at the constant rate

$$POV_i = \rho$$

## Percentage of Volume Algorithm

An optimal execution algorithm that tries to track a given POV rate, is the one that minimises the following objective function:

$$E_0[\sum_i (q_i - \rho v_i)] + \lambda Var_0[\sum_i (q_i - \rho v_i)]$$

However, a simple strategy (albeit not necessarily optimal) that works sufficiently well and is typically used by practitioners is the following one:

$$q_i = \rho v_{i-1}$$

i.e., just execute a constant proportion of the market volume traded at the previous interval, which is already known. Notice:

- This algorithm tracks the POV but cannot guarantee that a given quantity will be executed, unless forced to do so (e.g. with a market order at the end time)
- In the limit of very small intervals, the algorithm basically sends market orders every time there is a trade in the market, with the proportional volume.
- Some versions of the POV use limit orders whose size is adjusted as new trades are closed in the market. The trade-off is a risk to deviate from the benchmark.

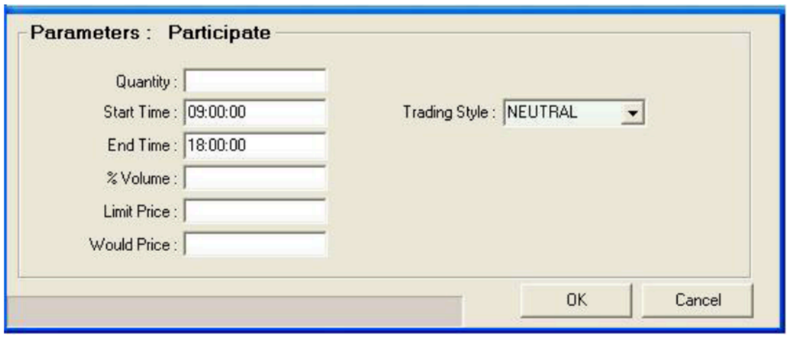
# Percentage of Volume Algorithm

## Example: Natixis's POV ("Participate")

The **Participate** strategy, also known as Percentage of Volume (POV) trades at a user defined percentage of the current market volumes on primary exchanges and MTFs until the order is completed or market closes. This strategy can be strict or dynamically adapted to market conditions.

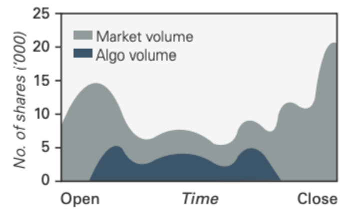
**Participate** algorithm aims to follow (live) the exchange volumes on the market by respecting a target level of participation.

### Participate Parameters




## CHARACTERISTICS

### Participate is to be favoured in the following cases




- Satisfied with current prices.
- Willing to limit the market impact on the execution period.

### Main Parameters

| | Description |
|---|---|
| **Start / End time** | By default, start time is at reception of the order, potentially including market open if received before the opening auction. End time will apply if the order has not been completed before. |
| **Trading style** | Conservative, neutral and aggressive. The conservative option suits a favorable price variation while the aggressive option applies better to an adverse price trend. |
| **%Volume** | Targeted participation rate. |

### Optional Parameters

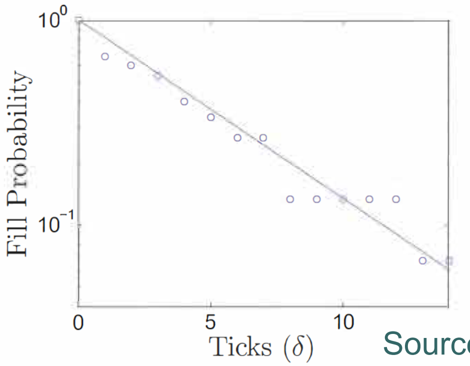
| | Description |
|---|---|
| **Price limit** | The strategy will apply the price limit to orders. Orders may not be completed. Volume traded outside the price limit is not taken into account by the strategy. |
| **Would price** | The strategy will attempt to complete the order if the stock trades at the Would price or better. The use of the Would option may significantly deviate the final participation rate from the targeted participation rate. |

# Execution tactics: optimal placement in an order book

Optimal placement problem: buy or sell a volume $q$ using orders available in the limit order book, in a maximum time $T$. We consider only limit and market orders. If there is a remaining quantity at time $T$, we execute it with a market order

Simple model:

- We place the full quantity using a limit order at a distance $\boldsymbol{\delta}$ ticks wrt the mid-price
- The optimal $\boldsymbol{\delta}$ is calculated using a model of historical fill probabilities (conditional to the arrival of a market order) fitted with an exponential distribution: $P(\delta) = e^{-a\delta}$
- Market orders arrival are modelled with a Poisson distribution of intensity $\lambda$
- Our final market order at $T$ has a linear market impact with parameter $k$
- At every time we reevaluate market conditions, check remaining time and quantity to execute, and recalculate optimal $\boldsymbol{\delta}$




Source: Cartea et al (2015)

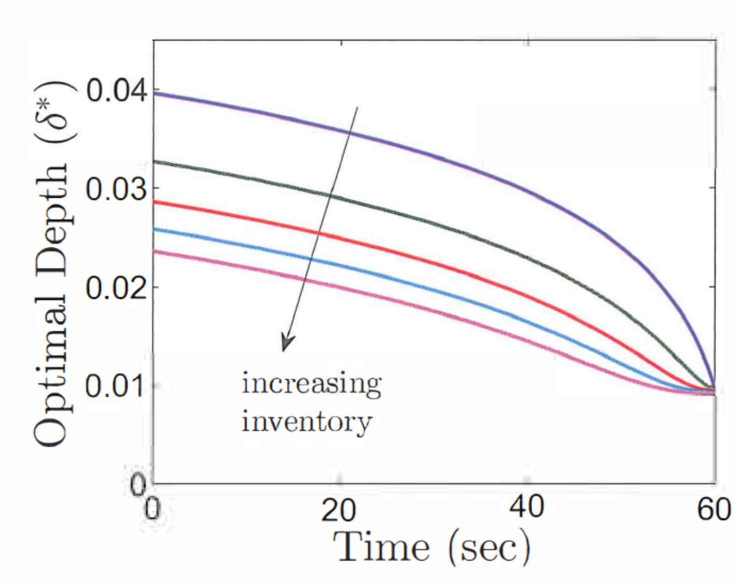
# Execution tactics: optimal placement in an order book

Solution: optimal number of ticks from the mid-price to place a limit order, for a given remaining quantity *q* and time *t*

$$\delta(t,q) = \frac{1}{a}\left[1 + \log \frac{\sum_{n=0}^{q} \frac{\lambda^n}{n!} e^{-ak(q-n)^2}(T-t)^n}{\sum_{n=0}^{q-1} \frac{\lambda^n}{n!} e^{-ak(q-1-n)^2}(T-t)^n}\right]$$

Remarks:

- The more the time and the less the remaining quantity, the more passive (farther from mid) the limit order
- The higher the market impact *k* the more aggressive (closer to mid), to avoid executing a final market order
- The higher the probability of filling (smaller *a*) and the higher the rate of market order arrivals λ the more passive the order






Source: Cartea et al (2015)

## Other execution scenarios: portfolio execution

Optimal portfolio execution problem: for a portfolio of *M* instruments, find the optimal *n* slices of size $q_{i,m}$ to buy an order of sizes $Q_m$ over time windows *T/n,* where *T* is the maximum time allowed for the purchase. For example, for optimal acquisition and the implementation shortfall benchmark:

$$IS = \sum_{m=1}^{M} \left( \sum_{i=0}^{n-1} q_{i,m} p_{i,m}^{exec} - Q_m p_{0,m} \right) + fees$$

$$min_{\{q_{i,m}\}} \left(E_0[IS] + \lambda Var_0[IS]\right)$$

$$\text{subject to} \sum_{i=0}^{n-1} q_{i,m} = Q_m \ , \ q_{i,m} \geq 0$$

Remarks:

- In principle, the execution can be done using M separated single-instrument executions
- However, the idea is to minimise the overall tradeoff cost vs risk of the portfolio:
  - If the instruments are correlated, one can potentially achieve an efficient frontier that is globally better than that of independent executions
  - Further optimisation can be achieved if there is flexibility in the instruments / sizes to be executed, typically in hedging application where the target is to neutralise the risk of a portfolio (e.g. and index future) with a given tolerance

# <u>Other execution scenarios: multiple markets</u>

<u>Smart Order Routing (SOR) problem</u>: find the optimal slices $q_k$ , $k = 1,\ldots,K$ of a single instrument order of size Q to be routed to a set of K different trading venues / liquidity pools, in order to minimise the execution costs (market impact, fees, …)

<u>Remarks:</u>

- This is a very relevant problem in multiple assets given the increasing fragmentation of markets, with multiple venues where an instrument can be traded
- These algorithms are typically inserted as an extra layer between execution strategies and execution tactics:
  - First an execution strategy slices the order in time, considering the trade-off between execution cost and timing risk
  - Then each slice is further split by the SOR algorithm and routed to the different trading venues / liquidity pools
  - Finally an execution tactic executes at each trading venue / liquidity pool the selected target amount
- These algorithms rely critically on good pre-trade transaction cost estimations at each trading venue / liquidity pool: probabilities of filling, market impact, …