Ex-Post Optimal Strategy for the Trading of a Single Financial Asset : Definition of an Absolute Distance to the Best Behavior

Olivier BRANDOUY[1], Philippe MATHIEU[2], and Iryna VERYZHENKO[3]

[1] LEM, UMR CNRS-USTL 8179 olivier.brandouy@univ-lille1.fr
[2] LIFL, UMR CNRS-USTL 8022 philippe.mathieu@lifl.fr
[3] LEM & LIFL, iryna.verizhenko@univ-lille1.fr

Obtaining good (even acceptable) performances with active management strategies in finance is a fairly hard challenge. Theoretically speaking, tenants of the efficient market hypothesis claim, with strong arguments, that a rational investor should stick to a simple *"Buy and Hold"* strategy for a correctly diversified portfolio (see for example Sharpe (1991) or Malkiel (2004)). Said differently, an active management, linked to hypothetical managerial skills for *market timing* or *stock picking* would mostly generate transaction costs without real benefit. Nevertheless, this debate is far from being closed (see for example Brock, Lakonishok, and LeBaron (1992), Shen (2003)) as well as the whole discussion on the profitability of active *versus* passive portfolio management styles.[4] In this paper, we do not discuss the opportunity of such active strategies based on *market timing* nor we describe an operational process allowing fund managers to find out how to identify states in the market where "buying" or "selling" is particularly appropriate. We neither propose a method that ranks various active strategies in terms of risk-return performance (although our framework might be extended to this bi-criteria framework). We rather propose a theoretical analysis of some absolute limits concerning the profits one can expect from the trading of financial assets.
First of all, one reason for doubting that active strategies can earn excess risk-adjusted returns is linked to the high level of stochasticity of financial markets. In other terms, these markets are hardly predictable. On the other hand, we stress that the realization of these excess returns is complex task due to an additional computational issues. We especially show that *even if* future prices were perfectly predictable, determining how to behave optimally with respect to this knowledge can be extremely complicated, and, in many cases, intractable. Therefore, the bottomline of this research is to identify an optimal strategy $S*$ providing the maximum profit one can obtain in trading some financial commodity, under a predefined set of constraints and with a complete knowledge of its price motion.
This question will be called the $S*-$*determination* problem. We show that this latter problem is far from being trivial, even if this target immediately evokes many popular models[5]. We provide a new algorithm that decreases the complexity class of this problem and propose a new method de-

[4] See for example Jensen (1968) for an argumentation about why actively managed funds should be avoided and, among others, Elton, Gruber, and Blake (1996) or Carhart (1997) for empirical arguments or explanations that do not match exactly this argumentation
[5] That most frequently prove to be completely inefficient

livering *an absolute performance indicator* geared towards the evaluation of a wide range of trading strategies. This algorithm establishes, for any series of prices, a boundary in terms of maximum profit that has not been proposed before to our knowledge.
We restrict our analysis to the evaluation of strategies involving the trading of a single asset (for example, a market index tracker). One potential application for this algorithm could be to estimate the *ex-post* performance of investment strategies or fund management principles that are formulated *ex-ante* the realization of prices over which these are deployed. It also provides an alternative to the relative rankings of investment strategies delivered by traditional methods.
We show that this best investment behavior can be defined using a linear programming framework and solved with a Simplex approach. Nevertheless, if this method is theoretically correct, it suffers from severe limitations in terms of computability (the underlying algorithm being non-polynomial in the worst case). We therefore propose to embed this question in a graph theory framework and show that the determination of the best investment behavior is equivalent to the identification of an optimal path in an oriented, weighted, bipartite network. We illustrate these results with real data as well as simulated algorithmic trading methods.
This paper is organized as follow. We first formalize the framework we start from and define explicitely the $S*-$*determination* problem. In a second section we present the mathematical frameworks related to these questions as well as a new algorithm geared at identifying the $S*$ strategy. In a last section we illustrate this latter algorithm and provide a some practical implementations to gauge the absolute performance of a few trading strategies.

1 Elements of the game and formalizations

Consider the idealised situation in which one investor has the complete knowledge of a finite series of financial prices $\overrightarrow{p} = \{p_t | t \in [0, n]\}$, for example daily closing prices for one given stock, index or portfolio.
Let's admit these prices, defined over a time window $[t = 1, t = n], n \in \mathbb{N}$ are those at which this investor has the opportunity to rebalance his portfolio.
Let's now posit a price-taker framework, *i.e.*, agent' decisions cannot affect these closing prices and sufficient liquidity at these prices is assumed.
We now define the rules of a game for this investor, or said differently, a series of rules constraining her behaviour:

- At the initialisation stage (*i.e.* at $t = 0$), the initial wealth W_0 of any agent is composed of a certain amount of cash (C_0) and no stock ($A_0 = 0$) : $W_0 = A_0 + C_0$. [6]
- Having the knowledge of the entire price series, the idealised investor must decide for each $t \in (1, n)$ one specific action with regards to the composition of her portfolio, "buy", "sell" and "stay unchanged", resp. coded `B`, `S` and `U`. In other terms, the investor has to compose a "sentence" of size n using characters in `B,S,U`. The interpretation of each of these actions is as follows:
  Buy: One can write `B` *if and only if* $W_{t-1} = C_{t-1}$. If `B` is written at date t, all the investor's cash is converted into assets (delivering a new quantity for $A_t \neq 0$). Assuming transaction costs at a $c\%$ rate,

[6] At date $t = 1$ –beginning of the game we posit C_1 to be equal to the first price of the considered time series.

$$A_t = \frac{W_{t-1}}{p_t \times (1+c)}$$

Additionally, the first character in any sentence must be a `B`.

Sell : *if and only if* $A_{t-1} \neq 0$, the investor can write `S` and convert his position into cash. Considering an identical rate of transaction costs c,

$$C_t = A_{t-1} \times (p_t \times (1-c))$$

Stay unchanged: Whatever the nature of W_{t-1} (cash or assets), she can also decide to write `U` and let her position unchanged at date $t : W_t = W_{t-1}$.

- This "sentence" is one investment strategy S_i over $\overrightarrow{p}$ chosen in a set of strategies $\{S\}$.[7]

Each instance S_i can be gauged in terms of relative performance with respect to an other strategy $S_{j,j\neq i}$ (and reciprocally). What we propose here is to determine an absolute performance indicator for each of these instances with respect to the best possible strategy in $\{S\}$ in terms of maximum profit $W_{t+n} - W_t$. As we will show later, this best strategy, denoted $S*$, is relatively easy to identify when transaction costs are not implemented. On the contrary, when transaction costs alter profits, this identification is far more complex.

A trivial method to solve this identification problem when transaction costs are implemented is to generate all possible sentences and to compute the net earning one can obtain with these to identify $S*$. This set is of finite size 2^n, thus exponential. As we will show now, there are at least two ways to improve efficiently the computation of the optimal strategy $S*$, whatever the level of transaction costs is. One is based on a simplex method, the other is based on the search of an optimal path in an oriented bipartite network.

2 Mathematical models : linear programming method and search in graphs

In this section, we show that the identification of $S*$ can be described as a linear programming problem with a classical Simplex solution. Unfortunately, this approach is relatively inefficient since the Simplex algorithm is non-polynomial in the worst case (*i.e.*, one can lack the necessary computing resources to obtain a result as soon as the size of $\overrightarrow{p}$ becomes important.)

2.1 Initial simplification

Before formal results are presented, we introduce two theorems that are necessary to find the solution of the problem. These preliminary elements aims at simplifying the solution we propose.

First simplification : *filtering the price sequence.*

Let's consider the price vector $\overrightarrow{p}$ consisting of three consecutive prices p_t, p_{t+1}, p_{t+2} and the function

$$R(x,y) = y(1-c) - x(1+c) \tag{1}$$

[7] Notice that in this framework, $Card\{S\} = 2^n$

In equation 1, the $R(x, y)$ function computes net earnings of successive buy and sell actions with $c\%$ transaction costs. In this equation, x denotes the price at which one buys and y the price at which one sells. By definition, y appears later in the time sequence than x. We show that $S*$ in $\overrightarrow{p}$, as defined page 2, can be identified in a subset of $\overrightarrow{p}$ denoted $\overrightarrow{fp}$[8] consisting of the extreme points in the price sequence (peaks and troughs) ignoring any intermediary points (here, p_{t+1}).
We assume $p_{t+2} \geq p_{t+1} \geq p_t$. Therefore $R(p_t, p_{t+2}) > R(p_t, p_{t+1})$ and $R(p_t, p_{t+2}) > R(p_{t+1}, p_{t+2})$. In this latter case, p_{t+2} is a "peak" while p_t is a trough.

Theorem 1 *Ignoring intermediary points: To identify $S*$, p_{t+1} can be ignored.*

Proof. Reductio ad absurdum / *proof by contradiction*:
If it were not the case, since it is not allowed to buy and sell at the same date : $R(p_{t+1}, p_{t+2}) > R(p_t, p_{t+2})$
Therefore: $p_{t+2}(1-c) - p_{t+1}(1+c) > p_{t+2}(1-c) - p_t(1+c)$
Which can be simplified: $-p_{t+1} > -p_t$
Thus, $p_{t+1} < p_t$ since, by definition $p_{t+1} > p_t$
Q.E.A ⁂

Notice that the same demonstration can be made by analogy in the case where $p_{t+2} \leq p_{t+1} \leq p_t$. As a consequence, if p_{t+1} is an intermediary point as exposed previously, it can be ignored to identify $S*$. In other terms, if one considers a complete price sequence $\overrightarrow{p}$, only peaks and trough should be selected to identify $S*$ (that is, $\overrightarrow{fp}$).

Lemma 1
No inclusion of losses : To identify $S*$, one can ignore all situations in which $R(x, y) < 0$

Litteraly, no trade with negative net earnings can be included in the best strategy which also excludes a situation where the so-called "buy and hold" strategy is not profitable.

Determining two vectors of prices for *potential* "buy" and "sell" actions.

From theorem 1 we know that it is necessary and sufficient for determining $S*$ to focus on extremum points in the price sequence. We now show that $\overrightarrow{fp}$ can itself be sliced in two separate sub-vectors of "peaks" and "troughs" corresponding to two independent potential "buy" and "sell" positions in $\overrightarrow{p}$ (*resp.* denoted $\overrightarrow{fp_B}$ and $\overrightarrow{fp_S}$).
Let's consider four consecutive prices $p_t, p_{t+1}, p_{t+2}, p_{t+3}$ such as $p_{t+1} > p_t$, $p_{t+3} > p_{t+2}$ and $p_{t+2} < p_{t+1}$.[9]

Theorem 2 *To identify $S*$, none of the $\overrightarrow{fp_B}$ can receive a `S` and none of the $\overrightarrow{fp_S}$ can receive a `B`.*

[8] *i.e.* $\overrightarrow{fp}$ for "filtered $\overrightarrow{p}$"
[9] In this latter case, we do not consider the situation in which $p_{t+2} > p_{t+1}$ since it is equivalent to the initial simplification case exposed previously.

Proof. (i) Since $p_{t+1} > p_t$, it is obvious that $R(p_t, p_{t+3}) > R(p_{t+1}, p_{t+3})$. Then $p_t \leftarrow \texttt{B} \succ p_{t+1} \leftarrow \texttt{B}$
(ii) Similarly, since $p_{t+2} < p_{t+1}$ it is obvious that $R(p_{t+2}, p_{t+3}) > R(p_{t+1}, p_{t+3})$. Then $p_{t+2} \leftarrow \texttt{B} \succ p_{t+1} \leftarrow \texttt{B}$
From lemma 1 we know that the situation in which $p_{t+3} < p_t$ can be ignored; Therefore, from (i), (ii) and lemma 1 :
– whether $p_t \leftarrow \texttt{B}$ and $p_{t+1} \leftarrow \texttt{U}$ from (ii); thus $p_{t+2} \leftarrow \{\texttt{U}$ and $p_{t+3} \leftarrow \{\texttt{U}$ or $\texttt{S}\}$
– or $p_t \leftarrow \texttt{U}$ and $p_{t+1} \leftarrow \texttt{U}$; thus $p_{t+2} \leftarrow \{\texttt{U}$ or $\texttt{B}\}$ and $p_{t+3} \leftarrow \{\texttt{U}$ or $\texttt{S}\}$
$(p_t, p_{t+2}) \leftarrow \{\texttt{U}$ or $\texttt{B}\}$; $\overrightarrow{fp_B} = \{p_t, p_{t+2}\}$
$(p_{t+1}, p_{t+3}) \leftarrow \{\texttt{U}$ or $\texttt{S}\}$; $\overrightarrow{fp_S} = \{p_{t+1}, p_{t+3}\}$
Q.E.D ■

This theorem does not state where to buy or to sell in the subsets $\overrightarrow{fp_B}$ and $\overrightarrow{fp_S}$ to identify $S*$. It uniquely states that it is not worth buying in any element of $\overrightarrow{fp_B}$ and selling in any element of $\overrightarrow{fp_S}$.

2.2 A linear programming method for the identification of $S*$

A first way to solve the $S*$ determination problem is to use a linear programming method. The basic idea here is to maximize an objective function subject to a set of constraints formalizing the rules in which this problem is embedded. We now expose how this program should be written.
Let denote $a(i,j)$ the potential benefit one can obtain if $p_i \in \overrightarrow{fp_B}$ and $p_j \in \overrightarrow{fp_B}$. Notice $a(i,j)$ is computed using equation 1. Let $x(i,j)$ be a dummy variable coding 0 or 1 that will be used to ignore (resp. to identify) transitions between any two prices p_i and p_j. If $p_i \leftarrow (\texttt{S}$ or $\texttt{U})$ or $p_j \leftarrow \texttt{U}$ then $x(i,j) = 0$, else $x(i,j) = 1$. Using these notations, the identification of $S*$ can be done solving the following linear problem:

$$max \sum_{(i,j) \in \overrightarrow{fp_B} \cup \overrightarrow{fp_S}} a(i,j)x(i,j) \tag{2}$$

$$\sum_{(i,j) \in S*} x(i,j) \leq n \tag{3}$$

$$\sum_{j} x(j,i) + x(i,j) \leq 1, \qquad \forall i \in \overrightarrow{fp_B} \tag{4}$$

$$x(i,j) + \sum_{k=1}^{j} + x(i+1,k) \leq 1, \qquad \forall i \in \overrightarrow{fp_B}, j \in \overrightarrow{fp_S} \tag{5}$$

$$x(i,j) + \sum_{k=j+1}^{n} x(i+1,k) \leq 2, \qquad \forall i \in \overrightarrow{fp_B}, j \in \overrightarrow{fp_S} \tag{6}$$

$$0 \leq x(i,j) \leq 1, \qquad \forall i \in \overrightarrow{fp_B}, j \in \overrightarrow{fp_S} \tag{7}$$

Literally, the objective function (2) states one seeks to maximize the total benefits in trading (that is, to identify $S*$). Constraint (3) implies that $S*$ cannot be composed of more than n prices while constraint (4) imposes the uniqueness of the solution. Constraints (5)- (6) do not allow backwards in the price series with respect to their sequential ordering. Constraint (7) requires that $x(i,j) = 1$

if a trade occurs between position i and j in $\overrightarrow{f_p}$, otherwise, $x(i,j) = 0$. This latter constraint means that the problem can be solved by simplex method.

However, it is virtually impossible to explicitly enumerate all these constraints when $\overrightarrow{f_p}$ is of moderate size. It is also recognized that the simplex algorithm is exponential even if it can be solved for certain cases in polynomial time.

We now propose to develop an alternative approach for this problem allowing an efficient solution. We tackled the $S*$ determination problem as the identification of an optimal path in an oriented bipartite network.

2.3 Embedding the identification of $S*$ in a Graph structure

Let each price in $\overrightarrow{fp}$ be depicted as a vertex in a network. $Card(\overrightarrow{fp} = k)$. Each vertex is indexed with an integer with respect to its place in the price series. We show now how to construct a bipartite, oriented and weighted network $\mathcal{N}\big(E, \overrightarrow{fp_B}, \overrightarrow{fp_S}\big)$ connecting points in $\overrightarrow{fp_B}$ and $\overrightarrow{fp_S}$.

Definition: Let $\aleph_X$ the subset of vertices succeeding vertex X. The network $\mathcal{N}$ is defined by the successors of each vertex.

Graph construction:

The initial situation from which we start is : $\forall X \in \overrightarrow{fp}, \aleph_X = \emptyset$. From this situation, two different kind of edges can be build :

Trading edge ($TE_{i,j}$): for any two vertices $i \in \overrightarrow{fp_B}$ and $j \in \overrightarrow{fp_S}$, vertex $j \in \aleph_i$ if and only if :

1. $j > i$ (which ensure temporal consistency)
2. c being the rate of transaction costs,

$$R_{i,j} = p_j(1-c) - p_i(1+c) \geq 0 \tag{8}$$

Forward edge ($FE_{m,n}$): for any two vertices $m \in \overrightarrow{fp_S}$ and $n \in \overrightarrow{fp_B}$, $n \in \aleph_m$ if and only if :

1. $n > m$ (which ensure temporal consistency)
2. $\aleph_n \neq \emptyset$

Notice we impose a "time consistency rule"[10] to avoid backward connections in this bipartite oriented graph. This means that a starting vertex p_{t+k} cannot be connected to a ending vertex p_{t+l} with $k \geq l$.
The rule presented in equation 8 obviously determines a profit as in equation 1. For any two vertices, these profits[11] can be analyzed as weights for the corresponding edges of $\mathcal{N}$.
Consequently, we receive a balanced, bipartite, weighted and directed network. We propose to interpret weights computed with 8 as distances between two vertices in the following proposition:

Proposition 1
$S*$ in this framework is a longest path problem.

[10] Similar to equations 5 end 6
[11] Provided these are positive.

Theorem 3 *For $c > 0$ and any 4 consecutive prices p_t, p_{t+1}, p_{t+2}, p_{t+3} in a filtered price series such as $\overrightarrow{fp}$ (see section 2.1) with $R(t, t+1) > 0, R(t, t+3) > 0, R(t+2, t+3) > 0$:*
$R(t, t+1) + R(t+2, t+3) > R(t, t+3)$

Proof. $-p_t(1+c) + p_{t+1}(1-c) - p_{t+2}(1+c) + p_{t+3}(1-c) > -p_t(1+c) + p_{t+3}(1-c)$
$p_{t+1}(1-c) > p_{t+2}(1+c)$
$p_{t+1}/p_{t+2} > (1+c)/(1-c)$
$c > 0 \Rightarrow (1+c)/(1-c) > 1 \Rightarrow p_{t+1}/p_{t+2} > 1$ by definition.
Q.E.D ■

In the construction of $\mathcal{N}$, one can notice that the number of edges depends upon the level of transaction costs c:

- The greater c the fewer the number of edges in $\mathcal{N}$ and the easier the solution of the problem as well.
- When $c \rightsquigarrow 0$, the graph tends to be more and more connected. For a specific threshold θ, $\mathcal{N}$ is fully connected (with respect to the time consistency rule). θ can be computed linearly; for any two consecutive prices in $\rightarrow fp$, $p_t \in \overrightarrow{fp_B}$ and $p_{t+1} \in \overrightarrow{fp_S}$:

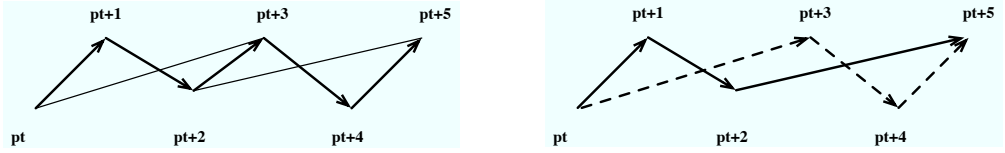
$$\theta = min(p_{t+1} - p_t)/(p_{t+1} + p_t) \tag{9}$$

  In the example provided section **??** (see table **??**), this threshold is 3%.

When $c < \theta$, $\mathcal{N}$ in fully connected. In this situation, we can derive from theorem 3 the following corollary:

Corollary 4
No backtracking: $\forall c < \theta, S* = \sum_{i=1}^{k-1} TE_{i,j=(i+1)}$

In other terms, when $c < \theta$, it is proved that $S*$ is the path connecting all the edges as they appear in sequential order (see figure 1(a)). $S*$ connect all the vertices. When $c > \theta$, this result cannot be established. For example, in figure 1(b), we posit c such as $R(t+2, t+3) < 0$; one



(a) Complete Bipartite Network : no backtracking

(b) Uncomplete Bipartite Network : backtracking

Fig. 1. Illustration of the "No Backtracking Theorem"

cannot follow a path in the price series connecting all vertices : many potential interesting paths can be discovered (see figure 1(b)) [12] and therefore must be compared to determine $S*$. One way

[12] Two of these paths are presented, one with plain lines, the other with hash lines. As soon as a TE between consecutive vertices is missing this kind of situation may occur

to tackle this *backtracking* problem could be to compute all possible paths, which would deliver an exponential algorithm.

We now show how we decrease the complexity class of this problem using a new algorithm to determine $S*$ in this graph formalism.

2.4 The $S* - determination$ algorithm

In this section we develop a new algorithm adapted to the determination of $S*$ in the graph framework. This algorithm derives from a technique exposed by Floyd (1969). Floyd originally developped this algorithm for finding a shortest path between every pair of vertices in a graph. This algorithme is transformed and adapted to fit our constraints.

We first introduce some notations and present the Floyd shortest-path algorithm; then we expose the $S* - determination$ algorithm itself.

Identifying the shortest path in $\mathcal{N}$ with the Floyd algorithm

Let ${d^k}_{ij}$ denote the length of a shortest path from vertex i to vertex j, where only the first k vertices are allowed to be intermediate vertices. If no such path exists, then let ${d^k}_{ij} = \infty$. Using this definition of ${d^k}_{ij}$, it follows that ${d^0}_{ij}$ denotes the length of a shortest path from i to j that uses no intermediate vertices.
Let ${d^0}_{ii} = 0$ for all vertices i. Furthermore,${d^n}_{ij}$ represents the length of a shortest path from i to j. Let D^k denote the $n \times n$ matrix whose i, j^{th} element is ${d^k}_{ij}$. If we know the length of each edge in the graph, then we can determine matrix D^0. Ultimately, we wish to determine D^n, the matrix of shortest path lengths. The Floyd shortest-path algorithm starts with D^0 and computes D^1 from D^0. Then, the algorithm calculates D^2 from D^1. This process is iterated until D^n is computed from D^{n-1}. Notice that only the elements of matrix $D^k - 1$ are needed to compute the elements of D^k.Moreover, these computations can be performed without reference to the underlying graph (see Minieka (1978)).
Therefore, the Floyd shortest-path algorithm can be expressed in pseudo-code as in Figure 2.

```
for k=1 to n
    for i=1 to n
       for j=1 to n
           path[i][j]=min ( path[i][j], path[i][k]+path[k][j])
```

Fig. 2. Floyd Algorithm

It is well known that the total amount of computation required by the Floyd algorithm is therefore proportional to $2n^3$, which means that the Floyd algorithm requires $O(n^3)$ running time.

Operating the $S*-determination$ algorithm

If the Floyd algorithm is performed with a maximisation procedure instead of a minimisation operation, this latter algorithm will produce the *maximum longest path* which corresponds, in our formalism, to $S*$.
The pseudo-code of the $S*-determination$ algorithm is presented Figure 3.

```
for k=1 to n
    for j=k to n
        path[0][j]=max ( path[0][j], path[0][k]+path[k][j])
```

Fig. 3. $S*-determination$ algorithm

We now present in details how the $S*-determination$ algorithm can be used for finding the longest path between the initial edge in $\overrightarrow{fp_B}$ to any other edge in $\mathcal{N}$.

1. Setting-up D^0
   (i) Number the vertices of $\mathcal{N}$ $1, ..., n$.
   (ii) Determine the matrix D^0 whose i, jth element equals the length of the longest arc from vertex 1 to vertex j if any.
   (iii) If no such arc exists, let $d_i^0 j = -\infty$.
   (iv) Let $d_i^0 i = 0$ for each i.
2. Recursive computations of D^k
   (i) For $k = 1, ..., n$ successively determine the elements of D^k form elements of D^{k-1} using the following recursive formula:

$$d_i^k j = \max\{d_{ik}^{k-1} + d_{kj}^{k-1}, d_{ij}^{k-1}\} \tag{10}$$

   (ii) As each element is determined, record the path that is represents.
3. Upon termination, the i, jth element of matrix D^n represents the length of a longest path from vertex i to vertex j.

The optimality of this algorithm follows inductively from the fact that the length of a longest path from i to j allowing only the first k vertices to be intermediate vertices must be the bigger of (i) the length of a longest path from i to j allowing only the first $k-1$ vertices to be intermediate vertices and (ii) the length of a longest path from i to j that allows only the first k vertices as intermediate vertices and uses the k^{th} vertex once as an intermediate vertex.

The complexity of the $S*-determination$ algorithm has now to be established. One must remember the Simplex solution is exponential and so is the simple enumeration of all possible paths in the network. In our case, the longest path from vertex 1 to every other vertex is searched. During the first iteration one must go over $n-1$ vertices. Hence, $n-1$ additions and $n-1$ minimisations have to be processed. Thereby, the first iteration consists of $2(n-1)$ operations. Similarly, it is possible to show that the second iteration consists of $2(n-2)$ operations and so on.

$$\sum_{i=1}^{i=n} 2(n-i) = n(n-1) \tag{11}$$

Thereby, the $S*-determination$ algorithm requires $O(n^2)$ running time and therefore $\in PSPACE$.

Notice we should also build other longest-path algorithms able to take into account the constraints we face on solutions such as the one proposed by Dantzig (1966) or Shier (1973). The first solution is similar to Floyd (1969) although the order in which the calculations are performed is different. The second algorithm, known as the *double-sweep algorithm*, finds the k shortest path lengths between a specified vertex and all other vertices in the graph and can also be tuned to our problem.

3 Numerical Illustrations and conclusive remarks

We now propose one application of the $S*-determination$ algorithm with the daily Dow-jones index at the daily frequency level from 2/12/1980 to 20/02/2009 (*i.e.* 7156 observations). No one can seriously defend the idea that one particular economic agent could be able to predict with some accuracy the next 7156 closing prices of the Dow-Jones Index by Dec., 2nd, 1980. Notice that even if it were possible (which is most imporbable), taking advantage of this knowledge under the constraints enumerated in section 1 would also be extremely difficult if simply possible without using the $S*-determination$ algorithm. With this algorithm, we determine the best behavior with transaction costs c respectively at 0% and 5%. The maximum wealth one should obtain in these two cases is bigger than 1.10E+015 in the first case and bigger than 1.83E+010 in the second case. These figures seem extraordinarily high : one must keep in mind they are simply impossible to obtain because of the global impredictibility of the market motions at date t with regards to the available information at this date. In figure 4 we present the evolution of an investor's wealth adopting $S*$ in both contexts.

Nevertheless, on shorter horizons, some agents claim they can produce such predictions or at least detect specific dates where it is worth enterring the market or shorting their positions. For example, technical traders claim they can detect signals in past prices (based on patterns) associated with potential market reversals.

Among others, one popular model for technical traders consists in comparing two moving averages based on past prices.[13] One is computed over a long range period L, the other on a short time window s. If MM_s crosses MM_L from the top to the bottom, technical traders would predict a further decrease in stock prices and try to sell immediately their holdings. On the contrary, if these moving average cross from the bottom to the top, the signal will be interpretated as "buy" signal. In Figures 5 we generated such signals using the same data as previously; we also computed portfolios managed with respect to the signals. For this purpose the artificial investor is endowed with an amount of cash equal to the Dow-Jones index value at date 1 (974.40). Notice that the "moving averages" strategies provide an example of the "rules of the game" presented in section 1. Concerning the signals sub-figures, we only present a limited time window for graphical clarity reasons. The portfolio subfigures report the evolution of an investor's wealth using these signals in context of 0% transaction costs.

[13] The moving average with i lags MM_i is equal to $(1/i)\sum_i(p_{(t-i+1)})$

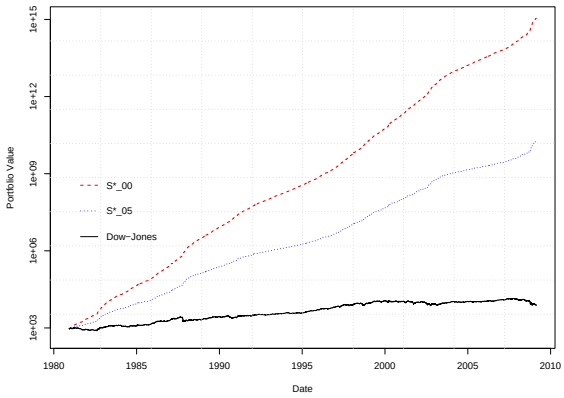
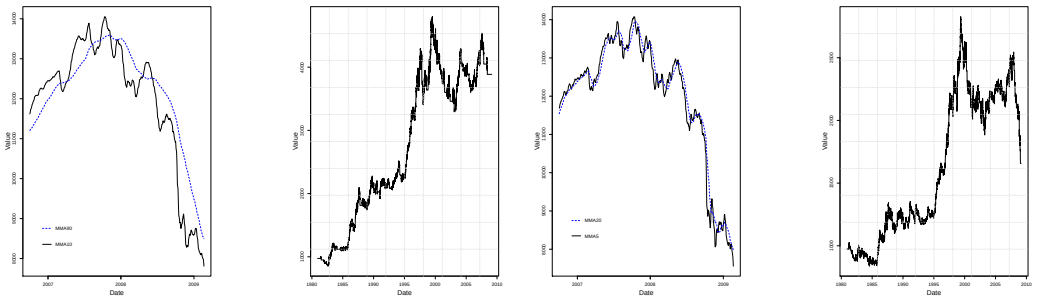


Fig. 4. $S*$ with resp. $c = 0\%$ and $c = 0.5\%$ and Dow-Jones Index (y axis in log scale)

In Figure 5(a), MM_s is based on 10 days while MM_L is based on 90 days. With these values we can generate 135 signals in the complete time window, which delivers the portfolio evolution. In Figure 5(c), these moving averages are respectivly based on 5 and 20 trading days which delivers 469 signals. Notice none of these strategies is interesting in any manner



(a) Excerpt of the signals, MM_{10} v.s. MM_{90} (b) Counter-part portfolio, MM_{10} v.s. MM_{90} (c) Excerpt of the signals, MM_5 v.s. MM_{20} (d) Counter-part portfolio, MM_5 v.s. MM_{20}

Fig. 5. Two investment strategies based on moving averages techniques

One can easily rank these strategies in term of overall profitability : MM_{10} v.s. MM_{90} seems to perform better than MM_5 v.s. MM_{20} in this price sample since the first one bears an overall profitability of +299% (terminal value of the portfolio = 3886.36) against +70% for the second (terminal value : 1657.18). In any case, one can also measure how far these two strategies ar from the optimum $S*$. In other terms, whatever the relative performance of any trading strategy, $S*$ can

be used to gauges its absolute performance.[14]
Resolving the $S*-determination$ problem does not give insights on the kind of signals one should feed automatic trading systems with, nor indicate a plausible behavior for any real-world investor. It simply establishes a boundary that was, to our opinion, largely unknown, and proposes a reference in terms of maximum-profit trajectory against which any population of investment trajectories can be gauged.

References

Brock, W., J. Lakonishok, and B. LeBaron (1992): "Simple Technical Trading Rules and the Stochastic Properties of Stock Returns," *Journal of Finance*, 47(5), 1731–1764.

Carhart, M. M. (1997): "On Persistence in Mutual Fund Performance," *Journal of Finance*, 52(1), 57–82.

Dantzig, G. (1966): "All Shortest Routes in a Graph," in *Theory of Graphs, International Symposium, Rome*, pp. 91–92. Gordon and Breach, New York.

Elton, E. J., M. J. Gruber, and C. R. Blake (1996): "The Persistence of Risk-Adjusted Mutual Fund Performance," *Journal of Business*, 69(2), 133–57.

Floyd, R. (1969): "Algorithm 97, Shortest Path Algorithms," *Operations Research*, 17, 395–412.

Jensen, M. C. (1968): "The performance of mutual funds in the period," *Journal of Finance*, 23, 389–416.

Malkiel, B. (2004): "Can Predictable Patterns in Market Returns be Exploited Using Real Money?," *Journal of Portflio Management*, 30, 131–141.

Minieka, E. (1978): *Algorithms for Networks and Graphs*. Marcel Dekker.

Sharpe, W. F. (1991): "The Arithmetic of Active Management," *The Financial Analysts' Journal*, 47(1), 7–9.

Shen, P. (2003): "Market timing strategies that worked – based on the E/P ratio of the S&P 500 and interest rates.," *Journal of Portfolio Management*, 29, 57–68.

Shier, D. (1973): "Iterative Methods for Determining the k Shortest Paths in a Network," *Networks*, 6, 205–230.

[14] In our example, both MM_{10} v.s. MM_{90} and MM_5 v.s. MM_{20} were poor performing strategies. A simple Buy and Hold behavior "buying" the market at date 1 and selling it at date 7156 perfoms far better than these two moving average techniques. Nevertheless, one can suppose some automatic trading strategies could perform better than this B&H strategy, especially in the context of high frequency data.