UNIVERSITA' DEGLI STUDI DI TORINO

FACOLTA' DI ECONOMIA

CORSO DI LAUREA IN ECONOMIA E COMMERCIO

TESI DI LAUREA

Eventi di Rilevanza Economica e Reazioni degli Agenti in un Mercato Borsistico Simulato

Relatore:

Prof. Pietro Terna

Correlatore:

Prof. Sergio Margarita

Candidato:

Marco Canavesio

Indice

Capitolo 1
Complessità e simulazione

1.1 Complessità

Uno dei tanti obiettivi che l'umanità si è proposta di perseguire riguarda la comprensione della realtà. La realtà di ogni giorno è una realtà complessa caratterizzata da un elevato numero di elementi che interagiscono continuamente tra loro generando quello che la società e l'uomo definiscono evoluzione. La società di oggi non è altro che un punto, collocato su una retta dei tempi, appartenente ad un grande sistema complesso in continua evoluzione: ciò che si colloca prima è definito con il termine di storia, ciò che si colloca dopo è noto con il termine di futuro. La società è solo uno dei tanti sistemi complessi che formano la realtà; può essere considerata come un sistema al quale appartengono migliaia di altri sottosistemi altrettanto complessi e la quale appartiene a sua volta ad altri sistemi ancora più grandi. Un chiaro esempio è costituito dall'intero universo.

Il desiderio di imparare e comprendere una realtà complessa, ha spinto l'umanità alla semplificazione, là dove fosse possibile, dei vari sistemi complessi con lo scopo di comprenderne meglio il loro funzionamento. Questa operazione è di estrema difficoltà e presenta innumerevoli ostacoli. Un'attenta analisi dei sistemi conduce alla seguente classificazione strutturale:

- *Sistemi semplici*
- *Sistemi complessi*.

Data la loro importanza, è bene analizzarle separatamente specificando per ciascuna categoria le relative caratteristiche, pregi e difetti.

1.1.1 I sistemi semplici

Un sistema è definito semplice quando è caratterizzato da una sola causa e da un solo effetto. La relazione è univoca e ciascuna causa è in grado di produrre uno ed un solo effetto. In un sistema di questo tipo, data una certa causa, è possibile prevedere a priori la rispettiva conseguenza: se questa sarà ritenuta desiderabile, se ne favorirà il verificarsi altrimenti, nel caso la conseguenza non risultasse gradita, si cercherà di

evitare il verificarsi della causa generatrice. Il notevole vantaggio attribuibile a questi sistemi riguarda l'assenza di interazione reciproca; grazie a questa caratteristica, gli effetti derivanti da ogni causa possono essere isolati e trattati singolarmente. Le cause che generano gli effetti possono essere individuate all'interno del sistema e le eventuali manipolazioni che si effettuano sul sistema hanno lo scopo di ottenere effetti sempre differenti. I sistemi semplici dispongono di alcune caratteristiche che meritano particolare attenzione (*Parisi, 2001*):

- Quando un sistema semplice è dinamico nel tempo, le modalità di cambiamento che lo governano sono determinabili o comunque prevedibili. Grazie a questa caratteristica è possibile determinare con una certa precisione l'evoluzione di un sistema purché siano noti gli stati precedenti.
- Due sistemi semplici, caratterizzati da diverse condizioni iniziali, avranno sviluppi differenti proporzionali alle diversità che li caratterizzano.
- Un sistema semplice può essere isolato dal proprio contesto senza che si verifichino particolari distorsioni a carico del sistema. Il contesto di cui fa parte un sistema semplice non è determinante nei confronti dei risultati ottenibili dal sistema stesso.
- Le parti che compongono un sistema semplice e che concorrono a formare i propri effetti sono facilmente individuabili e possono essere analizzate disgiuntamente.
- Un sistema semplice non fa generalmente parte di un sistema gerarchico e gli elementi dei vari sistemi non interagiscono l'uno nei confronti dell'altro.
- Un sistema semplice, date le sue caratteristiche appena citate, può essere riprodotto anche in più copie identiche.

1.1.2 I sistemi complessi

I sistemi semplici sono per definizione molto facili da trattare, ma un'attenta osservazione della realtà conduce a ritenere che essa sia prevalentemente costituita da sistemi complessi più che da sistemi semplici. Nonostante i sistemi complessi si rivelino di gran lunga più articolati dei sistemi semplici, costituiscono formalmente i principali sistemi su cui poggia la realtà, sociale e non solo. Uno degli aspetti più cruciali dei sistemi complessi risiede nelle loro caratteristiche, esattamente contrarie a quelle dei sistemi semplici. La principale caratteristica che accomuna i sistemi complessi riguarda innanzitutto l'assenza di indipendenza tra i propri elementi

costitutivi; tutti concorrono unitamente alla formazione degli effetti e le relazioni esistenti tra le cause non sono né lineari né additive. In un simile contesto, non è possibile isolare ciascun singolo elemento dal proprio sistema di appartenenza. Gli elementi di un sistema complesso sono generalmente molti e diversi l'uno dall'altro; nonostante ogni singolo elemento interagisca solo con un ristretto numero di altri elementi, da tali interazioni possono emergere effetti all'interno del sistema che non sono né deducibili né prevedibili a priori benché si conosca perfettamente la struttura dei singoli elementi. I sistemi complessi possiedono alcune caratteristiche di notevole rilievo (*Parisi, 2001*):

- Quando un sistema complesso è dinamico nel tempo, le modalità di cambiamento che lo governano non sono né determinabili né prevedibili e, nel caso lo siano, lo sono comunque poco. In questi casi non è possibile determinare l'evoluzione del sistema.
- Due sistemi complessi, caratterizzati da impercettibili differenze nelle proprie condizioni iniziali, possono evolversi in maniera completamente diversa.
- All'interno dei sistemi complessi sono presenti rapporti di causa ed effetto interagenti tali per cui ogni singolo elemento è in grado di influenzarne altri i quali, a sua volta, possono influenzare il primo.
- I sistemi complessi sono generalmente costituiti da gerarchie di elementi e da altre gerarchie di sistemi. Ogni singolo sistema rappresenta un sottoinsieme e contemporaneamente un superinsieme di altri sistemi.
- All'interno dei sistemi complessi non è facilmente identificabile il ruolo che ogni singolo elemento assume all'interno del sistema nella generazione degli effetti. Questa caratteristica impedisce che un elemento sia isolato dal resto del sistema per individuare le conseguenze che è in grado di generare.
- A differenza di quanto accade per i sistemi semplici, i sistemi complessi non possono essere riprodotti in copie identiche.

1.1.3 Sistemi semplici o sistemi complessi?

I sistemi semplici sono in stretto legame con qualcosa che è comprensibile, prevedibile e controllabile, mentre i sistemi complessi sono ciò che appare all'uomo ed alla società che lo circonda, meno controllabile e meno comprensibile. I sistemi semplici possono essere elaborati dalle capacità cognitive dell'uomo ma i sistemi complessi, date le loro caratteristiche, non possono essere gestiti dalla mente umana.

L'inadeguatezza dei metodi tradizionali nello studiare i sistemi complessi appare evidente nell'utilizzo dei metodi sperimentali. L'utilità dei metodi sperimentali è indiscutibile e rappresenta un elemento fondamentale per la scienza. Questo metodo presenta però il notevole svantaggio di essere principalmente adatto allo studio di sistemi semplici. Tramite il metodo sperimentale tradizionale, lo scienziato è in grado di operare su una causa tutti i cambiamenti che desidera facendo in modo che questa si verifichi oppure no e dai risultati che emergono osserva e deduce l'effetto delle proprie manipolazioni. In un sistema complesso accadde esattamente l'opposto: gli effetti sono il risultato derivante dall'azione concomitante di diverse cause che interagiscono tra loro. Sotto queste condizioni risulta completamente inutile manipolare od isolare una singola causa per studiare gli effetti che si generano all'interno dell'intero sistema.

Sempre per ciò che riguarda i metodi sperimentali classici, occorre ancora considerare alcuni aspetti. Il metodo sperimentale tradizionale, affinché possa essere attuato, richiede che i fenomeni analizzati siano ripetibili e riproducibili. Questa caratteristica è comune a tutti i sistemi semplici ma non a quelli complessi che sono caratterizzati da una maggiore sensibilità nei confronti delle proprie condizioni iniziali e dell'ambiente in cui si sviluppano. Con queste caratteristiche risulta fondamentalmente impossibile garantire che i fenomeni siano ripetibili e riproducibili nel tempo.

L'interesse della scienza è sempre stato prevalentemente rivolto alla semplificazione dei sistemi, ma ricondurre un sistema complesso ad un sistema più semplice mediante riduzioni e semplificazioni, pur essendo un'operazione necessaria per comprendere meglio le logiche alla base dei sistemi, non rappresenta certamente una soluzione sufficiente. Data l'evidente inadeguatezza degli strumenti tradizionali, oltre a dover semplificare i modelli in modo corretto e coerente con lo scopo che si desidera perseguire, occorre anche individuare nuovi strumenti concettuali e metodologici più consoni allo studio dei sistemi complessi. Grazie all'ausilio dei calcolatori, questo problema è stato in parte superato e diversi sono ora gli strumenti di ricerca rivolti alla gestione dei questi sistemi. Tra gli strumenti più importanti vanno certamente annoverate le *Reti Neurali Artificiali (RNA)* e le simulazioni; entrambi sono strumenti che richiedono l'utilizzo di un calcolatore e saranno trattati dettagliatamente in seguito. Quando la teoria è espressa attraverso l'utilizzo di un calcolatore, il modo di osservare la realtà muta radicalmente e, gran parte dei limiti che normalmente si pongono nella formalizzazione dei modelli, vengono a decadere. Il fatto che l'uomo non sia in grado di gestire direttamente i sistemi complessi ed abbia bisogno di poterli ricondurre a sistemi più semplici in grado di poter essere formalizzati dalla mente umana, rende l'utilizzo

degli strumenti informatici una necessità inderogabile. Grazie alle potenzialità in costante crescita rese disponibili dai computer, si è in grado di gestire sistemi sempre più complessi ed articolati.

Mentre la metodologia tradizionale perviene alla soluzione dei problemi tramite l'*analisi* dei fenomeni, i metodi computerizzati propongono la risoluzione dei problemi tramite una *sintesi* dei fenomeni. Il metodo tradizionale opera un'analisi della realtà partendo dalla realtà stessa; questa viene scomposta in più parti ed i fenomeni sono generati e studiati assemblando le singole parti tramite la teoria, il ragionamento e la razionalità. Questo modo semplicistico di studiare la realtà è decisamente pratico ma anche superficiale ed inappropriato per i sistemi complessi. Non bisogna dimenticare che una delle fondamentali caratteristiche dei sistemi complessi è proprio la loro reciproca interazione dalla quale discende l'inscindibilità dei singoli elementi dall'intero sistema. Le simulazioni agiscono in modo opposto: si propongono di sintetizzare la realtà partendo dall'interazione esistente tra le varie componenti. In tal modo si è in grado di giungere alla comprensione degli avvenimenti mediante l'interazione che i vari elementi presentano in determinate condizioni ambientali. A seconda delle caratteristiche proprie di ciascun elemento, e in base all'ambiente virtuale in cui gli elementi sono inseriti, si possono generare innumerevoli condizioni che rappresentano diverse situazioni realmente possibili.

Concludendo si può affermare che, mentre la scienza tradizionale cerca di comprendere la realtà così com'è osservata, attraverso le simulazioni si cerca di capire la realtà ricostruendola tramite l'interazione dei suoi elementi costitutivi. I sistemi semplici possono essere studiati e capiti tramite una semplice analisi di una realtà semplificata, ma i sistemi complessi, per poter essere compresi e studiati, devono essere necessariamente ricostruiti.

1.2 Reti Neurali Artificiali

L'uso delle reti neurali artificiali è estremamente indicato per lo sviluppo di esperimenti scientifici che riguardano tanto l'economia quanto gli altri campi di studio. Il loro grado di applicazione è molto elevato e spesso sono utilizzate come strumenti complementari per la costruzione di modelli basati su agenti di cui si discuterà meglio in seguito.

1.2.1 Principi fondamentali

Le reti neurali artificiali (*RNA*) sono reti costituite da nodi interconnessi tra loro su diversi strati. Una funzione a rete neurale è in grado di produrre un vettore di output da un determinato vettore di input in base ad un insieme di parametri denominati pesi la cui determinazione ricalca i concetti delle regressioni statistiche in ambito multivariato. Un neurone artificiale è rappresentato in figura.

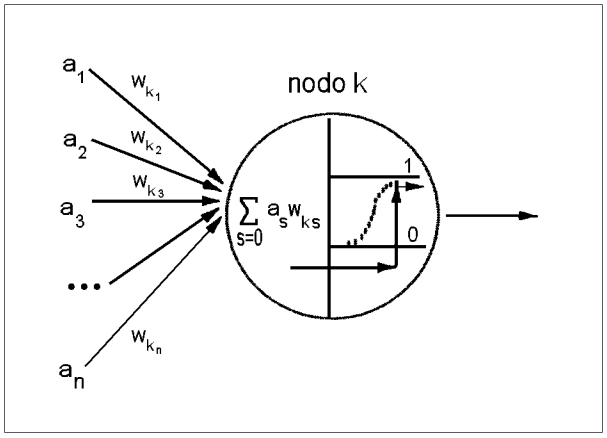


Figura 1 - Schema di un neurone artificiale

Le reti neurali artificiali maggiormente utilizzate sono quelle di tipo *feedforward*. In queste reti le informazioni non sono mai retroattive. Una rete neurale artificiale di questo tipo è generalmente costituita da tre strati di unità elementari di calcolo: uno strato costituito da nodi di *input*, uno da nodi nascosti (*hidden*) ed uno da nodi di *output*. Ogni nodo è collegato a tutti i nodi dello strato successivo mediante dei rami ai quali è attribuito uno specifico valore numerico definito peso.

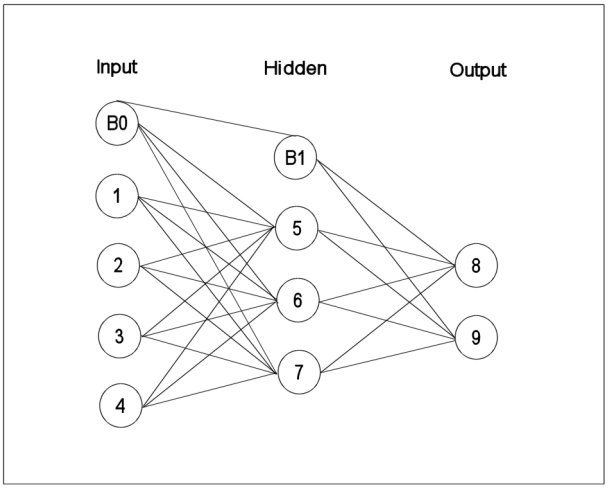


Figura 2 - Schema di una rete neurale artificiale

I segnali di input pervengono al neurone con i relativi pesi *w* specificati in un'apposita matrice qui denominata *A*. All'interno di ciascun neurone appartenente allo strato intermedio (nascosto), si effettua la sommatoria di tutti i segnali provenienti dagli input pesati in accordo con i rispettivi pesi per fornire in uscita dei valori trasformati mediante una generica funzione logistica in grado di confinarli in un intervallo prefissato. Lo stesso procedimento è attuato per lo strato successivo al fine di ottenere i valori di output veri e propri. I valori ottenuti dai nodi nascosti sono così inviati come input ai nodi di output tramite una nuova matrice di pesi qui denominata *B*. All'interno dei nodi di output si esegue nuovamente la sommatoria pesata dei segnali provenienti dai nodi nascosti e, tramite un'opportuna funzione di trasformazione, si generano gli output veri e propri. I nodi indicati con la lettera *B* sono denominati nodi di *bias* e determinano l'ordine di grandezza dei valori generati dalla rete neurale. In termini econometrici rappresentano il termine costante di una regressione matematica.

La rete deve essere in grado di riprodurre in uscita un valore predefinito per ciascun set di dati posti in input; l'insieme dei valori campione costituiti dagli input e dagli output assume il nome di *pattern*. Tramite i pattern si effettua l'apprendimento definito in termini informatici *training set*. Una rete neurale, dopo aver effettuato l'apprendimento, potrà essere verificata su un insieme di dati appositamente utilizzati per il controllo. Le matrici utilizzate in una rete neurale artificiale avente cinque nodi di input, tre nodi nascosti e due nodi di output hanno la seguente forma:

$$A = \begin{bmatrix} w_{50} & w_{60} & w_{70} \\ w_{51} & w_{61} & w_{71} \\ w_{52} & w_{62} & w_{72} \\ w_{53} & w_{63} & w_{73} \\ w_{54} & w_{64} & w_{74} \end{bmatrix}$$

$$B = \begin{bmatrix} w_{80} & w_{90} \\ w_{85} & w_{95} \\ w_{86} & w_{96} \\ w_{87} & w_{97} \end{bmatrix}$$

Il primo numero di ciascun peso indica il nodo di destinazione mentre il secondo numero rappresenta il nodo in partenza. Il peso w_{63} ad esempio sarà quello attribuito al ramo che collega il nodo di input numero *3* con il nodo nascosto numero *6*.

Definito con *y* il vettore degli *output* e definito con *x* il vettore degli *input*, la funzione completa implementata dalla rete neurale può essere così rappresentata:

$$y = f (B [1, f (A [1, x']')']')'$$

Dove $y = [y_8 \; y_9]'$ è il vettore degli *output* mentre $x = [x_1 \; x_2 \; x_3 \; x_4]'$ è il vettore degli *input*. Il valore *1* aggiunto all'inizio dei vettori rappresenta i nodi *bias*.
I nodi di *input* corrispondono ai valori che il vettore *x* assume in corrispondenza di ciascun caso (*pattern*) del *training set*, i valori dello strato nascosto corrispondono alle componenti del vettore $f (A_x)$ mentre i valori dello strato di *output* corrispondono alle componenti del vettore $f (B (f (A_x))$.

La determinazione dei pesi *w* appartenenti ad una rete neurale artificiale si basa sulla minimizzazione dell'errore misurato come differenza tra i valori *y* di *output* ed i valori utilizzati come *pattern*. Per ogni caso (*pattern*), si disporrà di un vettore *x* di valori in *input* ed un vettore *t* di risultati attesi. L'errore da minimizzare è calcolato mediante una doppia sommatoria:

$$\sum_n \sum_k (t_k - y_k)^2$$

L'indice *n* va da *1* al numero totale di *pattern* mentre l'indice *k* individua i singoli pesi degli *output*.

I pesi delle due matrici qui denominate *A* e *B* sono inizialmente fissati a caso. Durante la fase di apprendimento sono modificati tramite una correzione effettuata per ogni singolo *pattern*. Con il termine apprendimento si indica il processo tramite il quale la rete neurale, utilizzando come campioni i *pattern* forniti, calcola i pesi che minimizzano l'errore citato. L'apprendimento è svolto mediante una correzione dei pesi svolta ciclicamente diverse volte sui pattern a disposizione. Il numero di cicli che occorre effettuare per ottenere una rete neurale sufficientemente utile allo scopo varia da caso a caso e dipende strettamente dalla struttura imposta alla base della rete neurale. Gli elementi che maggiormente influenzano il tasso di apprendimento di una rete neurale

sono generalmente il numero di nodi *input*, *hidden* ed *output*, l'entità numerica dei dati forniti in *input* e la precisione nella correzione degli errori. In media il processo di apprendimento può essere ripetuto anche miglia di volte. La modifica di ciascun peso avviene sulla base di una quota, definita *learning rate*, mediante la quale ciascun peso è corretto in base alla derivata calcolata sull'errore; la quota della derivata è aggiunta al valore precedente del peso che si sta correggendo. Alla correzione è inoltre aggiunta un'ulteriore quota definita *momentum* della correzione precedente; tramite il *momentum* si intende mantenere la direzione e l'intensità del movimento di correzione. Il processo appena esposto prende il nome di *algoritmo della backpropagation* o *retropropagazione dell'errore*.

1.2.2 Utilizzo delle reti neurali artificiali

Le reti neurali artificiali, assieme alle simulazioni, rappresentano un ottimo strumento per la comprensione dei sistemi complessi e, grazie alla possibilità di utilizzarle assieme alle simulazioni, si rendono estremamente flessibili per l'implementazione di modelli economici basati su agenti artificiali adattivi. Gli *agenti artificiali adattivi* sono agenti che interagiscono all'interno di ambienti simulati sulla base di conoscenze apprese tramite reti neurali artificiali generalmente implementate con la tecnica dei *cross target*.

1.2.2.1 Agenti Artificiali Adattivi (*AAA*)

I modelli fondati su *agenti artificiali adattivi* (***AAA***) si propongono di simulare l'evoluzione di sistemi popolati da agenti che cercano di comprendere l'ambiente in cui operano ed agiscono in base ai propri livelli cognitivi. In questi modelli gli agenti sono sempre costruiti secondo il principio della razionalità limitata e si differenziano dagli agenti che dispongono di razionalità piena ed illimitata in quanto non dispongono di un'informazione completa. La costruzione di agenti con razionalità limitata è molto semplice e si attua tramite l'utilizzo di funzioni a reti neurali artificiali che non richiedono l'ottimizzazione di un particolare aspetto. Quali sono i motivi che spingono però un ricercatore ad utilizzare reti neurali artificiali per realizzare agenti con razionalità limitata? Esistono almeno cinque ragioni (*Terna, 1995*):

- Le reti neurali artificiali riproducono la struttura del cervello umano sia pure in una visione semplificata.
- Le reti neurali artificiali si allontanano significativamente dall'impostazione simbolica appartenente alla prima tipologia di intelligenza artificiale.
- Le reti neurali artificiali dimostrano una particolare rilevanza matematica e statistica soprattutto nell'approssimazione delle loro funzioni.
- Le reti neurali artificiali, grazie alla loro struttura, sono particolarmente adatte all'applicazione parallela su più macchine.
- Le reti neurali artificiali dispongono di una eccezionale specificità nella struttura delle connessioni. Questa caratteristica le rende un ottimo strumento per l'implementazione del comportamento umano.

Nel mondo reale, la capacità di apprendere rappresenta un elemento fondamentale per qualunque forma di vita non solo umana. Gli agenti, così come avviene per qualunque altro essere vivente, devono adattare il proprio comportamento in base ai cambiamenti posti in essere dall'ambiente nonché alle conseguenze derivanti dalle azioni degli altri agenti. Nella rappresentazione del comportamento umano, occorre considerare non solo il tasso di apprendimento ma anche il modo in cui ciascun essere umano apprende. Un essere umano può apprendere le conoscenze che determinano il proprio comportamento con sé e con gli altri in modo razionale o irrazionale. Quando si sviluppa un agente artificiale è necessario tener presente queste considerazioni. Un perfetto agente artificiale dovrebbe comportarsi esattamente come un essere umano in modo tale che per un osservatore esterno risulti praticamente impossibile distinguere i due soggetti. La struttura attuale degli agenti artificiali non è ancora in grado di riprodurre agenti con simili caratteristiche ma col tempo è possibile che ulteriori studi possano portare a nuovi risultati concreti.

Gli agenti introdotti nei modelli di simulazione fondano il proprio comportamento su alcuni algoritmi modificabili in base ad una serie di prove e mentre per certi aspetti possono essere considerati più semplici di quelli dei modelli neoclassici, per altri devono essere ritenuti più complessi soprattutto a causa del continuo sforzo necessario per l'apprendimento delle caratteristiche ambientali in cui si sviluppano. In conclusione, tre sono le particolarità più importanti che caratterizzano gli agenti adattivi:

- Sono in grado di apprendere ed adattarsi in base alle condizioni fornite dall'ambiente che li circonda.

- Operano ed interagiscono sulla base di una razionalità limitata.
- Sono in grado di individuare e comprendere il comportamento degli altri agenti con cui possono interagire.

1.2.2.2 Il metodo dei *Cross Target (CT)*

Si è già analizzato il funzionamento di una rete neurale artificiale e si sono già esposte le caratteristiche generali di un agente adattivo. Ora si approfondisce il funzionamento reale di questi agenti illustrando dettagliatamente il modo in cui agiscono ed apprendono (*Terna, 1995*). Un agente artificiale adattivo agisce ed apprende tramite una rete neurale basata sui *Cross Target*. Il nome *Cross Target* (*CT*) deriva dal metodo tramite il quale sono costruiti i *target* necessari all'apprendimento durante la fase di *training*. Gli agenti costruiti tramite una rete neurale a *Cross Target* sono caratterizzati dallo sviluppo di una specifica coerenza interna. La struttura di una rete neurale artificiale costruita tramite *Cross Target* è come in figura.

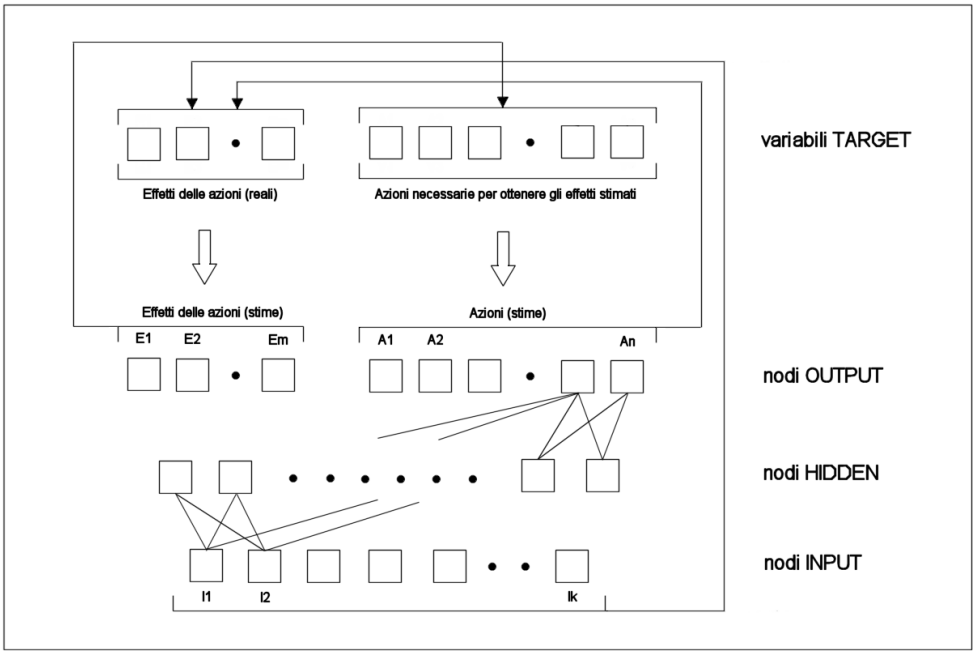


Figura 3 – Rete neurale artificiale implementata con il metodo dei Cross Target

Gli output della rete neurale si suddividono in due parti: nodi relativi alle *azioni* da compiere e nodi relativi agli *effetti* derivanti da tali azioni. Tramite i *cross target*, sia i target necessari per l'apprendimento della rete dal lato delle azioni, che i target necessari per l'apprendimento della rete dal lato degli effetti, sono costruiti in modo incrociato. I target delle azioni sono costruiti in coerenza con gli output relativi agli effetti; questo per sviluppare la capacità di decidere quali azioni intraprendere affinché

si producano gli effetti attesi. I target degli effetti, al contrario, sono costruiti in coerenza con gli output relativi alle congetture sulle azioni; questo per sviluppare la capacità dell'agente di stimare gli effetti relativi alle azioni che sta intraprendendo. Il concetto che sta alla base del funzionamento dei *cross target* è che ogni agente economico sviluppi con l'apprendimento le capacità necessarie per agire in maniera coerente; deve essere in grado di individuare quali azioni compiere per ottenere un risultato specifico e quali conseguenze derivano da specifiche azioni. Nonostante il metodo dei *cross target* utilizzi anche altri criteri per adattare il comportamento degli agenti, quello esposto rappresenta il metodo di maggior rilevanza.

Come anticipato nel paragrafo precedente, una rete neurale che utilizzi la tecnica dei *cross target* suddivide gli output in due parti: le *azioni* che un agente compie o congettura di compiere e le congetture in merito agli *effetti* di tali azioni. Gli *input* del modello sono costituiti dai dati provenienti dall'ambiente e dalle azioni svolte precedentemente dal soggetto o dagli altri agenti; entrambe le informazioni possono anche essere indipendenti dalle azioni svolte precedentemente dal soggetto simulato. I *target*, a differenza degli input, sono noti solo al momento dell'azione. L'algoritmo dei *cross target* è in grado di produrre simultaneamente azioni ed apprendimento; questo perché le azioni sono necessarie per costruire le informazioni su cui si basa l'apprendimento dell'intera rete neurale. Le azioni e l'apprendimento si svolgono in quattro fasi:

1) ***Output della RNA:*** in base ai valori di input e ai pesi, la rete neurale valuta le azioni da compiere e gli effetti derivanti da tali azioni.
2) ***Target per gli effetti:*** rappresentano i valori che la rete deve imparare a riprodurre e sono costruiti sulla base delle azioni decise dall'altro lato della rete neurale. In tal modo le valutazioni sugli effetti diventano coerenti con le valutazioni sulle azioni.
3) ***Target per le azioni:*** costituiscono le differenze esistenti tra i target e gli output della rete. Possono essere considerate come punto di partenza per la modifica delle azioni in modo da avvicinare queste alle relative congetture sugli effetti. Il processo è duplice: dagli effetti si determinano le azioni e viceversa.
4) ***Backpropagation:*** l'ultima fase consiste nell'apprendimento effettuato mediante la correzione dei pesi della rete. In tal modo è possibile ottenere delle stime sugli effetti maggiormente coerenti con le conseguenze delle azioni congetturate e congetture sulle azioni maggiormente coerenti con le stime sugli effetti.

Si supponga di avere un generico effetto E_1 conseguente a due azioni denominate A_1 e A_2. Il target per l'effetto è:

$$E_1' = f(A1,A2)$$

Il primo scopo è quello di ottenere un output E_1 simile a E_1' che costituisce la misura esatta dell'effetto delle azioni $A1$ e A_2. L'errore nei confronti di E_1 è:

$$e = E_1' - E_1$$

Per minimizzare l'errore si utilizza il sistema della retro-propagazione secondo le regole già esposte nel paragrafo precedente. Il secondo scopo è quello di fare in modo che le azioni siano coerenti con i valori prodotti dagli effetti. Per far ciò occorre correggere i pesi che determinano le grandezze A_1 e A_2 al fine di renderle coerenti con le grandezze A_1' e A_2'. Seguendo la forma già adottata per gli effetti, si perviene ad una formula analoga anche per le azioni:

$$A_1 = g_1(E_1', A_2)$$
$$A_2 = g_2(E_1', A_1)$$

Scegliendo un valore casuale T da una distribuzione uniforme, dalle due formule precedenti si ottiene:

$$A_1' = g_1(E_1' - e\ T_1, A_2)$$
$$A_2' = g_2(E_1' - e\ T_2, A_1)$$

Le funzioni g_1 e g_2 collegano le azioni agli effetti ed hanno generalmente natura lineare. Gli errori che occorre minimizzare dal lato delle azioni sono pertanto i seguenti:

$$a_1 = A_1' - A_1$$
$$a_2 = A_2' - A_2$$

Nel caso le azioni determinassero più di un effetto, si applicherebbe la correzione relativa alla grandezza con il maggiore errore in modulo. L'utilizzo di una correzione basata su un processo pseudo-casuale è necessario per garantire una buona variabilità nel processo di apprendimento che altrimenti si bloccherebbe rapidamente

sui valori centrali degli output. Tramite una adeguata variabilità si forniscono alla rete le condizioni per un continuo adattamento alle varie modificazioni ambientali.

Ripetendo le fasi di apprendimento per più cicli, è possibile notare che le matrici variano i propri pesi in maniera diversa a seconda che si tratti della matrice *A* o della matrice *B*. I pesi della matrice *B* che collega i nodi nascosti con i nodi di output mutano molto più significativamente di quanto non mutino i pesi della matrice *A* che collega invece i nodi di input con i nodi nascosti. Il motivo deve essere ricercato nel metodo di correzione degli errori. Dal momento che gli errori sono corretti partendo dagli output mediante retro-propagazione, è deducibile che i pesi della matrice *B* cambino più rapidamente di quelli appartenenti alla matrice *A*. In termini scientifici, la modifica della matrice *B* determina un apprendimento a breve termine mentre la modifica della matrice *A* determina un apprendimento a lungo termine.

Grazie ai *cross target* è possibile creare agenti in grado di agire sulla base dello sviluppo di una propria coerenza interna. Questi elementi sono più che sufficienti per determinare lo sviluppo autonomo degli agenti, ma se si desidera raffinare ulteriormente le capacità offerte dai *cross target* in modo da poterli maggiormente utilizzare in contesti di natura economica, è necessario sviluppare nuove estensioni al metodo. Generalmente le varianti introdotte sono due ed entrambe sono considerate target esterni:

a) ***Obiettivi esterni:*** gli obiettivi esterni sono necessari per influenzare le stime sugli effetti e sostituiscono in genere quelli costruiti in modo incrociato. Gli obiettivi esterni consentono di impostare un determinato scopo agli agenti i quali, nella scelta delle proprie azioni, dovranno considerare l'obiettivo che è stato loro chiesto di ottenere.
b) ***Proposte esterne:*** le proposte esterne sono necessarie per influenzare le congetture relative alle azioni e rappresentano in genere uno dei target disponibili per l'apprendimento dal lato delle azioni. Tra questi sarà poi scelto quello maggiore in modulo.

Un semplice esempio di obiettivo esterno applicato ad un modello economico può essere il raggiungimento di una certa ricchezza oppure il mantenimento di un certo prezzo. L'utilizzo degli obiettivi esterni e delle proposte esterne rappresenta un elemento di notevole rilevanza in quanto, dalla variazione di alcuni parametri o caratteristiche, possono emergere risultati in grado di mostrare sostanziali differenze

tra un modello e l'altro. Su queste esperienze e su questi risultati, lo sperimentatore sarà in grado di comprendere i fenomeni alla base del modello nonché gli elementi che maggiormente lo influenzano.

1.2.2.3 Modelli basati su agenti (*ABM*)

I *modelli basati su agenti* possono implementare tanto strutture economiche e finanziarie quanto altri mondi artificiali di cui spesso ne risulta difficile, se non addirittura impossibile, l'osservazione. In qualunque campo un modello sia sviluppato, l'obiettivo che questo si propone di raggiungere riguarda sempre la scoperta e l'interpretazione di nuove conseguenze derivanti dalle azioni e dalle interazioni dei propri agenti adattivi (*AAA*). Queste conseguenze possono essere prevedibili, imprevedibili o impreviste; in ogni caso, lo sperimentatore che intenderà studiare ed approfondire il funzionamento alla base del modello, apprenderà direttamente dal modello che ha sviluppato. E' possibile individuare due tipologie di modelli ad agenti: un modello che utilizza *un solo agente* e dei modelli che utilizzano *più agenti* o *più popolazioni*.

1. Modelli con un solo agente

Rappresentano i modelli più semplici da realizzare. Qui non esiste un'esplicita interazione tra gli agenti in quanto ne esiste solo uno ma sussiste un'interazione indiretta derivante dalle risposte provenienti dall'ambiente. Il contesto può anche essere considerato prevedibile e sotto controllo ma il processo di apprendimento dell'agente è determinato dall'introduzione di apposite *regole esterne*, dalle reazioni provenienti dall'*ambiente esterno* e dallo sviluppo della propria *coerenza interna*.

I dati su cui gli agenti fondano l'apprendimento derivano dallo sviluppo della coerenza interna tra due tipi di congetture: *congetture sulle azioni* da intraprendere e *congetture sui possibili effetti* derivanti da tali azioni. Questa forma di apprendimento è svolta mediante l'utilizzo dei *cross target* ed i comportamenti che possono emergere sono tanto complessi quanto curiosi.

2. Modelli più agenti o più popolazioni

In questi modelli gli agenti interagiscono tra loro e da tale interazione apprendono come meglio adattarsi all'ambiente e agli altri agenti. Lo sperimentatore ha interesse a studiare l'evolversi delle competenze degli agenti nonché gli effetti aggregati delle loro azioni. Anche in questi modelli gli agenti apprendono tramite *regole esterne* introdotte dallo sperimentatore, tramite l'*ambiente* in cui agiscono e mediante lo sviluppo della propria *coerenza interna*. Ciò che li differenzia dai modelli con un solo agente è la presenza dell'*interazione*. L'interazione reciproca tra i vari agenti del modello determina la non linearità nella dinamica del modello per cui, anche piccole modifiche alla struttura degli agenti, alle regole, ai loro obiettivi o all'ambiente in cui agiscono, possono influenzare ed alterare notevolmente l'intero evolversi del sistema. Nonostante non sia stato omesso, è anche possibile sviluppare dei modelli nei quali interagiscono tipi diversi di popolazioni. Le popolazioni possono differire in termini di patrimonio informativo, tipo e numero di output (intesi come azioni ed effetti), regole applicate alle previsioni, modalità di sviluppo delle regole, presenza o assenza di specifici obiettivi e suggerimenti esterni.

Per quanto riguarda la loro interazione, i modelli ad agenti possono essere ordinati in base al loro livello di complessità:

- Al ***primo livello*** si collocano i modelli in cui l'ambiente è considerato esogeno. Questi modelli utilizzano equazioni introdotte dallo sviluppatore per generare tanto i dati di input quanto le risposte provenienti dall'ambiente nei confronti delle azioni degli agenti.
- Al ***secondo livello*** si collocano i modelli la cui interazione tra agenti influenza implicitamente l'ambiente. In questi modelli, le azioni ed i comportamenti attuati dagli agenti delle diverse popolazioni sono in grado di influenzare direttamente le variabili d'ambiente.
- Al ***terzo livello*** si collocano i modelli i cui gli agenti si influenzano reciprocamente sulla base delle rispettive azioni compiute. Ciascun agente determina le proprie azioni in base alle conoscenze che ha a disposizione sul comportamento degli altri agenti appartenenti allo stesso sistema. Ciò è sufficiente per garantire all'intero sistema un funzionamento coerente.

1.2.2.4 I modelli economici e finanziari

Terminata la trattazione sulle reti neurali e sui modelli ad agenti, occorre considerare l'aspetto applicativo di tali modelli in ambito economico e finanziario. Si possono individuare almeno tre modalità nella costruzione di modelli economici e finanziari tramite reti neurali artificiali. Le prime due riguardano modelli fondati su agenti mentre la terza è utilizzata per effettuare previsioni.

1) ***Modelli fondati su agenti che usano le reti neurali artificiali per formulare previsioni:*** sulla base di regole fornite direttamente dall'esterno, gli agenti operano e modificano le proprie capacità previsionali tramite l'apprendimento.
2) ***Modelli fondati su agenti aventi regole di comportamento determinate dalle reti neurali artificiali:*** gli agenti modificano le proprie regole di comportamento e decisione per adattarsi all'ambiente in cui operano. Da questi modelli è possibile ottenere ottime spiegazioni sul funzionamento dei meccanismi economici reali. Il principale svantaggio derivante da questi modelli è la loro scarsa applicabilità al campo delle previsioni nonostante possano ugualmente essere utilizzati per tale fine.
3) ***Modelli fondati su reti neurali artificiali quali funzioni matematiche:*** il terzo tipo di modelli utilizza le reti neurali artificiali come semplici funzioni matematiche in grado di produrre determinati output in base ad altri dati posti in input.

1.3 Simulazioni

L'analisi e lo studio di alcuni fenomeni complessi, hanno col tempo orientato gli scienziati all'utilizzo delle simulazioni come strumento di ricerca da affiancare alle metodologie tradizionali come la formalizzazione matematica e la trattazione verbale. Spesso, nella formulazione dei modelli, si utilizzano metodologie matematico-statistiche. Questi modelli, per poter essere ricondotti ad equazioni più facilmente risolvibili, sono costretti ad apportare notevoli semplificazioni ai fenomeni che intendono studiare; queste semplificazioni rendono generalmente i sistemi troppo semplici rispetto alla realtà che propongono di rappresentare e questa caratteristica rappresenta un vincolo rilevante per la ricerca scientifica. La trattazione puramente verbale, al contrario, rende il sistema estremamente complesso e difficilmente interpretabile non solo per un osservatore esterno, ma anche per il ricercatore stesso.

L'individuazione delle implicazioni sui risultati ottenuti, e la verifica dei legami di tipo quantitativo esistenti tra le diverse grandezze considerate, diventa decisamente complessa. Quale definizione si potrebbe allora attribuire al concetto di simulazione?

1.3.1 Cos'è la simulazione?

"Le simulazioni sono un nuovo modo di esprimere le teorie che non usa simboli" (*Parisi, 2001*). Mediate la simulazione, un modello può essere rappresentato tramite un programma sviluppato a computer in grado di essere utilizzato tanto per la descrizione di teorie qualitative quanto per la descrizione di teorie quantitative. Grazie alle elevate potenzialità e prestazioni di un calcolatore, è possibile rappresentare un sistema complesso caratterizzato da interazioni tra elementi non necessariamente lineari. I diversi concetti e postulati della teoria non sono né descritti a parole né rappresentati da formule, sono esplicitamente dichiarati e contenuti all'interno di un programma tramite il cui codice non si fa altro che definire il comportamento dei singoli elementi appartenenti al modello; elementi che in termini informatici sono definiti agenti. Attraverso una simulazione, a differenza di quanto accade attraverso le metodologie tradizionali, i fenomeni sono ricreati in un ambiente virtuale che può essere definito a piacere dallo sviluppatore in base alle caratteristiche che si intendono fornire al modello. Tanto il metodo matematico-statistico quanto quello verbale cercano di spiegare la realtà tramite la sua osservazione e la sua descrizione; le simulazioni ricreano una realtà alternativa in grado di essere osservata e confrontata con la realtà effettiva. L'utilizzo delle simulazioni permette un'evoluzione logica del concetto di ricerca. Il compito del ricercatore, nel creare una nuova teoria, si riconduce alla formalizzazione della teoria stessa attraverso un codice informatico al quale farà seguito l'osservazione dei risultati che il programma è in grado di generare e fornire.

Lo scienziato può elaborare le proprie teorie sulla base di ciò che già sa ed in seguito, anziché verificarle osservando la realtà, può ricreare un ambiente simulato all'interno di un computer per verificare i dati rilevati. La simulazione rappresenta uno strumento in grado di imprimere un carattere più oggettivo agli esperimenti mentali dello scienziato: lo studioso formula le teorie e le relative previsioni, il calcolatore ne verifica la correttezza mediante i risultati ottenuti dalla simulazione. L'utilizzo delle simulazioni non deve comunque accompagnarsi con l'abbandono delle altre metodologie tradizionali anzi, l'utilizzo congiunto delle diverse metodologie, informatiche e non, è fortemente consigliato per poter confrontare i risultati ottenuti dai vari metodi: se i

risultati provenienti dai diversi metodi conducono tutti allo stessa conclusione, si può affermare con una certa precisione che questi godano di una elevata attendibilità.

Un concetto che merita particolare attenzione riguarda la distinzione tra realtà e realtà artificiale. Lo scienziato, nelle formulazione e verifica delle proprie teorie, può adottare un controllo diretto sulla realtà oppure un controllo artificiale tramite l'analisi di ciò che accade all'interno di una simulazione. Quali elementi distinguono allora la realtà naturale da una realtà artificiale? Per il genere animale visto nel suo insieme, esiste un solo tipo di realtà che è rappresentata dalla realtà naturale. La realtà naturale è rappresentata da tutto ciò che avvolge l'uomo ed ha caratteristiche che non dipendono né dalla presenza dell'uomo né dalle sue azioni. L'uomo, come essere animale superiore, è stato in grado di creare una nuova realtà che è scaturita dal suo ingegno ed ha permesso di creare ambienti e condizioni che nella realtà naturale non sarebbero assolutamente realizzabili: la realtà artificiale. La realtà artificiale esiste in quanto creata dall'uomo all'interno di una macchina; ha le proprie caratteristiche, il proprio ambiente e le proprie interazioni con gli elementi che la costituiscono. Le simulazioni, come elementi facenti parte di questa realtà, possono essere considerate realtà; una realtà tramite la quale si cerca di comprendere tutti quei fenomeni che non possono e non potrebbero essere studiati tramite l'osservazione diretta della realtà naturale.

1.3.2 Vantaggi derivanti dalle simulazioni

E' difficile individuare e classificare tutti i vantaggi derivanti dalle simulazioni. E' stato più volte sottolineato che le simulazioni permettono di superare alcuni limiti imposti dalle tradizionali metodologie scientifiche ma questo non rappresenta di certo il loro unico vantaggio. I vantaggi derivanti dalle simulazioni sono molteplici ricalcano le orme di quanto già affermato in precedenza. I principali vantaggi derivanti dalle simulazioni sono i seguenti:

1) Le simulazioni consentono la creazione di ambienti che sarebbero impossibili da riprodurre nella realtà naturale. Tramite le simulazioni si possono riprodurre tutti gli ambienti che si desiderano.
2) Le simulazioni permettono la gestione dei sistemi complessi che non potrebbero essere studiati con le metodologie tradizionali sia per limiti cognitivi che per limiti mnemonici e di calcolo che la mente umana presenta.

3) Le simulazioni sono più vantaggiose di una sperimentazione tradizionale in quanto sono attuabili in tempi generalmente più contenuti e con costi notevolmente inferiori.

1. Creazione di ambienti specifici

Una caratteristica particolarmente importante delle simulazioni è la possibilità di creare ambienti che in altri contesti sarebbero difficili se non totalmente impossibili da realizzare. Le metodologie tradizionali conducono degli esperimenti di laboratorio tramite i quali si cerca di osservare la realtà per poterne dedurre il suo funzionamento nonché le leggi che la governano. Nonostante i notevoli sforzi dimostrati dalla scienza nel realizzare esperimenti prima inconcepibili, molti esperimenti non possono ancora essere completamente realizzati in laboratorio; questo a causa dell'impossibilità di creare un ambiente perfettamente consono al loro svolgimento. Grazie all'utilizzo delle simulazioni, è resa possibile la creazione dei nuovi mondi virtuali in cui le regole sono alterate in base alle esigenze. L'utilizzo di questi mondi può condurre ad una maggiore comprensione del mondo reale e consente una possibile comprensione di ciò che accadrebbe nel caso esistessero implicazioni ed ideologie alterate. Sempre grazie alle simulazioni, è possibile inoltre eseguire esperimenti che altrimenti non potrebbero essere attuati in un laboratorio reale poiché moralmente inaccettabili.

2. Gestione dei sistemi complessi

Le simulazioni rappresentano degli ottimi strumenti per la gestione dei sistemi complessi. I sistemi complessi, date le loro caratteristiche, rappresentano un ostacolo di notevole rilevanza per i tradizionali metodi di studio legati alla sperimentazione. Mediante l'utilizzo delle simulazioni è possibile gestire questi sistemi senza troppe difficoltà. La complessità dei modelli è gestita direttamente all'interno del calcolatore che, definendo tutti i comportamenti relativi agli agenti del modello, evita la creazione di regole che potrebbero ostruire ed alterare significativamente le basi dello stesso. Su questa caratteristica si è già notevolmente discusso durante la trattazione della complessità e delle simulazioni in generale e non sarà illustrata ulteriormente.

3. Maggiore rapidità nell'attuazione e minori costi di gestione

Le simulazioni presentano due vantaggi di notevole importanza pratica. Innanzitutto sono relativamente semplici da realizzare. Grazie all'utilizzo di diversi linguaggi di

programmazione, è possibile creare i propri ambienti virtuali senza dover dedicare troppo tempo alla stesura del codice necessario. La riproduzione di un ambiente artificiale è decisamente più veloce della riproduzione di un ambiente reale. La riproduzione di un ambiente reale richiede l'allestimento di molti elementi e non tutti potrebbero essere reperiti con facilità ed efficienza; un ambiente artificiale non richiede l'utilizzo di elementi fisici in quanto si rivelano sufficienti poche righe di codice. Le simulazioni presentano anche lo straordinario vantaggio di risultare decisamente più economiche rispetto alle tradizionali metodi sperimentali di laboratorio. Allestire un vero laboratorio sperimentale richiede tempo e soprattutto elevati costi di gestione che possono essere di diversa natura: dalle spese determinate dall'acquisto dei beni fisici come gli strumenti di laboratorio, alle spese legate all'aspetto umano come gli stipendi dei ricercatori nonché le spese di manutenzione dei locali allestiti. Le simulazioni, benché non eliminino completamente questi costi, ne consentono una notevole riduzione soprattutto per quanto riguarda le attrezzature fisiche e gli strumenti di laboratorio ricondotti generalmente ai soli sistemi informatici.

1.3.3 Critiche alle simulazioni

Anche le simulazioni sono suscettibili di critiche. Innanzitutto occorre sottolineare che esistono due tipologie di critiche nei confronti delle simulazioni: alcune sono considerate contestabili, altre fondate. Per evitare il sorgere di troppa confusione, saranno trattate separatamente.

1.3.3.1 Critiche contestabili

Le critiche contestabili sono considerate tali in quanto concedono la possibilità di fornire un'adeguata risposta in difesa. Le critiche provenienti dall'intero mondo informatico e scientifico imputabili a questa categoria sono principalmente quattro (*Parisi, 2001*):

1) Le simulazioni rappresentano una realtà troppo semplificata impossibile da utilizzare come materia di studio.
2) Le simulazioni non dicono nulla di nuovo rispetto a quanto non si sappia già: ricreano la realtà e basta.
3) La simulazione della realtà è praticamente inutile dal momento che la realtà non è perfettamente nota allo sperimentatore.

4) Le simulazioni non permettono di comprendere cosa accade realmente all'interno dell'ambiente simulato.

1. Le simulazioni sono una semplificazione della realtà

Le simulazioni sono spesso oggetto di critiche in quanto considerate una semplificazione eccessiva della realtà. Tale critica è evidentemente infondata e sottolinea la mancanza di piena consapevolezza sul significato delle simulazioni. Le simulazioni sono la traduzione di un modello scientifico ma non solo; all'interno di un calcolatore, tramite la stesura di un apposito codice di programmazione, è realizzata una vera e propria realtà alternativa in grado di riprodurre fenomeni complessi che circondano e regolano anche la realtà naturale. Affermare che le simulazioni semplificano troppo la realtà non è corretto soprattutto se si considera che la maggior parte delle metodologie tradizionali attua le proprie ricerche e perviene alle relative conclusioni proprio sulla base di modelli semplificati. E' sufficiente pensare allo studio dell'economia. L'economia attualmente studiata ed insegnata nel mondo intero è costituita in gran parte da modelli matematici rappresentativi di una realtà notevolmente semplificata che non sempre riflette l'esatto funzionamento di quanto è possibile osservare ogni giorno nella realtà dei fatti. Perché occorre allora criticare le simulazioni sotto questo aspetto quando la maggioranza degli attuali modelli fa di questo vincolo la base delle proprie ricerche? La critica che certamente può essere accolta riguarda il modo in cui queste semplificazioni sono attuate e, più precisamente, il modo in cui sono orientate. Quando si parla di orientamento ci si riferisce principalmente all'omissione di elementi considerati fondamentali per una corretta comprensione del fenomeno che si intende studiare. Di questo argomento si discuterà meglio in seguito.

2. Le simulazioni non dicono nulla di nuovo

Un'altra critica che spesso è rivolta alle simulazioni riguarda il fatto che queste non forniscono nulla di nuovo rispetto a quanto non si sappia già della realtà oggetto di studio. Ma se non introducono nulla di nuovo, perché utilizzarle? In realtà questa critica è facilmente contestabile. Spesso accade che una simulazione non restituisca il risultato che ci si aspettava bensì produca dei fenomeni leggermente o totalmente diversi. In molti casi le differenze possono assumere entità talmente notevoli al punto di dover riconsiderare seriamente l'intero modello chiedendosi se tali discrepanze siano realmente possibili oppure se si tratti solo di un errore concettuale commesso in fase di

progettazione. Dopo un accurato riesame del modello, nel caso non emergano particolari errori concettuali, non si può far altro che accettare i risultati ottenuti come qualcosa di cui non se ne era a conoscenza; qualcosa di realisticamente possibile, ma matematicamente così improbabile tale da non essersi mai verificato prima nella realtà osservabile. I risultati di una simulazione non consistono in semplici repliche della realtà; i fenomeni simulati, anche quando producono gli stessi risultati ottenibili dall'osservazione della realtà, si fondano sempre su teorie interpretative della realtà. Una simulazione è fondamentale anche se riproduce fenomeni empirici già rilevati; il fatto che i risultati derivanti da una simulazione coincidano con quelli riscontrabili nella realtà, rappresenta un elemento di conferma sulla correttezza della teoria alla base della simulazione.

3. La realtà non si conosce a sufficienza

Una delle maggiori critiche che spesso emerge dai dibattiti riguarda il legame esistente tra le simulazioni e le conoscenze delle realtà sottostanti. Secondo alcuni critici, l'uomo sa poco della realtà in cui vive, e questa caratteristica priverebbe di significato l'utilizzo delle simulazioni come strumento di ricerca. Non ha senso studiare e simulare qualcosa di cui non si ha perfetta conoscenza. Benché questa critica possa essere in parte accettata, occorre comunque sottolineare che le simulazioni servono proprio per comprendere qualcosa di cui non si ha perfetta conoscenza. Le simulazioni possono essere considerate delle ottime guide per la scoperta di nuovi fenomeni oltre quelli già noti; qualora una simulazione fornisca dati discordanti da quelli emergenti dalla realtà, lo scienziato potrebbe modificare la teoria che sta alla base del sistema oppure suggerire lo studio di nuovi fenomeni che prima non erano stati adeguatamente presi in considerazione.

4. Le simulazioni non spiegano ciò che accade realmente

Un'ultima critica sollevata nei confronti delle simulazioni riguarda la loro incapacità di spiegare cosa accade realmente all'interno dell'ambiente virtuale durante la simulazione. Le simulazioni non sono altro che l'espressione di una realtà tradotta in un modello implementato con linguaggi informatici. Per questa loro natura strettamente informatica, le simulazioni risultano pienamente comprensibili solo da parte di chi effettivamente possiede sufficienti conoscenze informatiche sulla programmazione. Tralasciando questo aspetto tecnico, occorre comunque riconoscere che, da un punto di vista logico, le simulazioni possono essere facilmente comprese anche da parte di

chi non dispone delle adeguate conoscenze matematiche necessarie per la comprensione di qualunque altro modello implementato con le metodologie tradizionali. La critica afferma che in una simulazione si conoscono perfettamente le variabili che sono introdotte nel modello ma non si capisce perché da queste emergano determinati risultati. Anche questa critica è facilmente contestabile. Si è precedentemente affermato che le simulazioni rappresentano uno dei migliori strumenti in grado di gestire i sistemi complessi e, grazie a questa loro caratteristica, si è anche affermato che nello sviluppo di una simulazione non è necessario comprendere tutte le regole alla base dei modelli; ciò che è necessario definire sono solo i comportamenti dei singoli elementi appartenenti al sistema. Il calcolatore, grazie alle elevate capacità di calcolo, gestisce il sistema al posto dello sviluppatore fornendo dei risultati che, in base alle condizioni fissate a priori, rappresentano la principale fonte di studio per il ricercatore. Compito del ricercatore sarà quello di confrontare i dati ottenuti dalla simulazione con quelli ottenibili dal realtà secondo le modalità espresse al punto precedente. Tramite semplici modifiche apportate alle caratteristiche dell'ambiente virtuale ed alle caratteristiche dei singoli elementi, lo scienziato è in grado di individuare quali effetti possono derivare dai cambiamenti effettuati.

1.3.3.2 Critiche fondate

Nonostante la maggior parte delle critiche sollevate nei confronti delle simulazioni possa essere facilmente contestata da opportune giustificazioni e spiegazioni, esistono alcune critiche che devono formalmente essere accettate come fondate. Queste critiche non rappresentano una minaccia all'utilizzo delle simulazioni ma certamente sottolineano dei limiti che devono essere presi in considerazione durante lo sviluppo dei modelli. Le critiche appartenenti a questa categoria sono formalmente due:

1) Esiste la tendenza da parte dello scienziato a legarsi troppo all'osservazione dei mondi simulati trascurando la verifica esterna della sua teoria rispetto alla realtà.
2) Le simulazioni possono semplificare in maniera eccessiva e poco opportuna la realtà che studiano introducendo o trascurando elementi non necessari o, viceversa, di fondamentale importanza.

1. Poche verifiche esterne

Uno dei maggiori problemi reali derivanti dalle simulazioni riguarda la miopia scientifica dimostrata dai ricercatori e dagli scienziati. Questi tendono spesso a dare maggior importanza alle verifiche interne piuttosto che alle verifiche esterne. Le verifiche interne consistono normalmente nell'ammissibilità dimostrata dal modello sviluppato mediante l'osservazione dei suoi risultati. La verifica interna è indubbiamente un elemento fondamentale della ricerca scientifica soprattutto quando si studiano sistemi e teorie complesse, pur tuttavia la scienza non può fermarsi alla verifica interna; per accertare l'efficienza dimostrata da un modello di simulazione è necessario procedere con un'adeguata verifica esterna. La verifica esterna consiste nel controllo dei risultati ottenuti durante la fase di simulazione, con la realtà che questi intendono rappresentare. Quello che si desidera ottenere mediante la verifica esterna è la presenza di coerenza tra il sistema simulato e la realtà sottostante. Spesso accade che lo scienziato concentri la propria attenzione solo sui risultati che ottiene attraverso la simulazione, senza accertarsi quale sia la reale corrispondenza tra i risultati ottenuti e la realtà osservabile. Per rendere le simulazioni un valido strumento di ricerca, è necessario che il ricercatore confronti sempre la realtà artificiale con la realtà osservabile.

2. Le simulazioni possono semplificare in modo errato il modello

Le simulazioni sono anch'esse una semplificazione della realtà e, come accade in tutti i modelli, anche le simulazioni sono soggette ad un errore sistematico causato da tali semplificazioni. Il problema riguarda soprattutto il modo in cui le semplificazioni sono effettuate. Quando si sviluppa un nuovo modello occorre semplificare ciò che è corretto semplificare ossia tutti quegli elementi che non sono di particolare rilevanza per lo studio del sistema in questione. L'attenzione prestata a questo tipo di problema è generalmente scarsa e la distinzione tra elementi rilevanti ed elementi irrilevanti è spesso trascurata. Un modello che riassume male la struttura di un sistema complesso omettendo degli elementi importanti od introducendone altri considerati superflui, non è in grado di produrre risultati che possano essere considerati accettabili e coerenti con la realtà che si intende studiare conducendo all'inutilizzabilità il modello. La scelta degli elementi da eliminare o inserire nei modelli è un'operazione estremamente delicata da cui dipende la validità o meno dei modelli stessi. Prima di determinare quali variabili siano da introdurre e quali da trascurare all'interno di un modello, è necessario un attento esame preliminare di tutta la realtà che si desidera

studiare. Grazie ad una corretta analisi preliminare, è possibile limitare notevolmente le imprecisioni e le distorsioni del modello.

Capitolo 2
Eventi

2.1 Teoria degli Eventi

Per analizzare l'effetto che gli eventi e le notizie hanno nei confronti di un mercato borsistico è indispensabile fornire innanzitutto gli elementi concettuali necessari per una buona comprensione dell'argomento. La prima parte di questo capitolo analizzerà gli eventi da un punto di vista teorico fornendo gli adeguati strumenti matematici di base necessari per una corretta comprensione dell'argomento. La seconda parte del capitolo sarà invece rivolta alla comprensione degli eventi in un contesto strettamente finanziario. In questa parte si analizzeranno i principali elementi che caratterizzano un mercato borsistico ed il modo in cui gli eventi e le notizie agiscono su tali elementi. L'ultima parte del capitolo vertirà invece sull'analisi dei mercati e dei principali strumenti economici e finanziari utilizzati per la gestione del rischio.

2.1.1 Tipologie di eventi

Un evento può essere rappresentato da tante cose: una buona notizia al lavoro, un imprevisto familiare o più genericamente qualsiasi fatto che in qualche modo possa influenzare una o più categorie di soggetti. Un evento non è altro che un avvenimento in grado di assumere solo due valori logici: vero o falso; si è verificato l'evento o non si è verificato. Gli eventi e le notizie coinvolgono la vita quotidiana in ogni modo e forma e, mentre alcuni entrano direttamente a far parte della nostra vita, altri si limitano solamente ad influenzarla o a lasciarla totalmente inalterata. Purtroppo è sempre molto difficile riuscire a formulare un'adeguata classificazione di tutti gli eventi possibili e questo a causa dell'elevata, per non dire infinita, quantità di eventi stessi. Il modo più semplice e completo di esaminare gli eventi può essere quello di classificarli in base alla loro natura e, a tal fine, si può iniziare con l'esposizione di tre concetti molto importanti che rappresentano i fondamenti dell'aleatorietà e sono oggetto di notevole confusione: la *certezza*, la *prevedibilità* e la *determinabilità*. Questi termini rappresentano tre aspetti ben distinti dell'aleatorietà.

2.1.1.1 Classificazione di base

- Un evento è definito ***certo*** quando la sua probabilità di verificarsi è pari ad uno. Tutti gli eventi che hanno probabilità di verificarsi compresa tra zero ed uno esclusi sono considerati eventi incerti. Un evento la cui probabilità di verificarsi è uguale a zero è definito evento *impossibile*. Un'assicurazione sulla vita è un classico esempio di evento certo: la morte di un soggetto. La scommessa fatta dalla società assicuratrice non sarà sulla morte o meno del soggetto, in quanto è ovvio che questa prima o poi avverrà, quello su cui l'impresa di assicurazioni scommette è il momento in cui accadrà l'evento.

- La ***determinabilità*** riguarda la possibilità che un evento possa o meno essere valutato a priori sulla sua entità od istante nel quale si verificherà o potrà verificarsi.

- La ***prevedibilità*** rappresenta il fatto che un evento sia prevedibile a priori oppure no. Un evento è considerato *prevedibile* quando, nonostante non si sia ancora verificato, lasci pensare che esso potrà verificarsi in un futuro più o meno prossimo. E' considerato imprevedibile ogni evento che, pur non essendosi ancora verificato, lasci intendere che non si verificherà neanche in un prossimo futuro. Mentre le prime due caratteristiche sono generalmente facili da determinare, quest'ultimo aspetto non sempre risulta di facile determinazione. La prevedibilità di un evento è sempre un estremamente soggettivo. Sebbene alcuni eventi possano essere ritenuti più che prevedibili da alcuni, da altri possono essere considerati assolutamente imprevedibili; tirando le somme ben pochi sono quelli che risultano indiscutibilmente prevedibili.

Le tre caratteristiche appena citate costituiscono la base principale su cui si fonda la principale classificazione degli eventi; la loro combinazione fornirà lo schema generale valido per la valutazione di tutti gli eventi. Non si può mai affermare che un evento sia incerto senza prima poter dire se esso sia anche determinabile e prevedibile. Alcuni eventi hanno delle implicazioni esplicite ed è naturale pensare che un evento certo sarà anche considerato prevedibile, ma queste implicazioni non tolgono nulla alla validità della classificazione. Un evento certo potrebbe essere indeterminabile come nell'esempio delle assicurazioni, mentre un evento incerto potrebbe essere perfettamente determinabile o prevedibile. E' sufficiente pensare ad una giocata al lotto: non si può sapere a priori se si vincerà oppure no, ma si sa con determinazione

che, nel caso si vincesse, la vincita sarebbe di un determinato importo matematicamente prestabilito in base alla giocata effettuata. Stando sulla falsa riga di quanto appena esposto, si possono individuare anche degli eventi che, pur essendo determinabili, sono anche imprevedibili.

2.1.1.2 Dipendenze tra eventi

Se si desidera approfondire ulteriormente questo argomento cercando di avvicinarsi il più possibile ad una realtà sempre più complessa, occorre introdurre un altro concetto senza il quale tutto apparirebbe incompleto ed astratto: la correlazione tra eventi. Finora si è parlato degli eventi come singoli fatti prevedibili o imprevedibili che, verificandosi, portano a delle precise implicazioni che possono o meno essere determinabili a priori. Nonostante la correttezza di quest'affermazione, è difficile immaginare una realtà fatta solo da singoli eventi; la maggior parte di questi, quando si verifica, genera delle implicazioni che spesso non sono conseguenze finali di un processo aleatorio bensì semplici eventi intermedi legati in qualche modo al verificarsi del primo. Ma non è tutto. In alcuni casi il verificarsi di un evento può anche escludere totalmente il verificarsi di un altro. Nel primo caso si parla di *dipendenza* tra eventi, nel secondo di *compatibilità*. Si analizzino le due tipologie in maniera disgiunta.

1. Eventi indipendenti

Due eventi si definiscono *indipendenti* se il verificarsi di uno non condiziona il verificarsi dell'altro. In termini analitici:

$$P(A \cap B) = P(A) \cdot P(B)$$

Per contro, due eventi saranno dipendenti se:

$$P(A \cap B) \neq P(A) \cdot P(B)$$

Quanto appena scritto merita alcune riflessioni. Perché due eventi se sono indipendenti devono avere una probabilità congiunta uguale al prodotto delle rispettive probabilità? Per rispondere a questa domanda è necessario fare un breve richiamo al calcolo delle probabilità. Nel calcolo delle probabilità, si definisce probabilità composta il prodotto delle probabilità dei singoli eventi. Essa rappresenta la probabilità che tutti gli eventi

presi in considerazione si verifichino contemporaneamente. Se il loro prodotto non corrisponde alla probabilità che si verifichi l'evento congiunto allora significa che alcuni di essi sono legati da rapporti di interdipendenza tali da influenzare la probabilità globale. La differenza tra i due valori rappresenta la componente della probabilità comune ad entrambi gli eventi. In questi termini è possibile estendere l'argomento prendendo in considerazione il caso in cui gli eventi indipendenti siano più di due. In questo caso la situazione sarà del tutto analoga alla precedente con l'unica differenza che la formulazione logica dovrà riflettere le relazioni fra tutti gli eventi. In formule:

$$P(A_1 \cap A_2 \cap ... \cap A_k) = P(A_1) \cdot P(A_2) \cdot ... \cdot P(A_k)$$

L'espressione è del tutto analoga a quella illustrata nel caso precedente. La formula esposta definisce degli eventi *mutuamente indipendenti*. Dalla formula esposta discende un'ulteriore caratteristica che merita particolare attenzione: la *probabilità condizionata*.

La probabilità condizionata è la probabilità associata al verificarsi di un evento il quale dipende a sua volta dal verificarsi di un altro particolare evento ad esso legato. In formule si ha:

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

La probabilità che un evento *B* condizionato all'evento *A* si verifichi è data dal rapporto tra la probabilità che i due eventi si verifichino congiuntamente e la probabilità che si verifichi il solo evento *A*. Sebbene queste argomentazioni matematiche possano sembrare ovvie ed anche un po' inutili ai fini pratici, rappresentano un ottimo strumento analitico per una corretta valutazione dei avvenimenti reali e, nonostante appaiano di primo impatto un po' complicate, sono senza dubbio necessarie per una corretta trattazione dell'argomento.

2. Eventi compatibili

Due eventi risultano compatibili quando possono realizzarsi contemporaneamente senza che il verificarsi di uno escluda il verificarsi dell'altro. Per contro, due eventi si definiscono incompatibili quando non possono realizzarsi contemporaneamente ed il verificarsi di uno esclude implicitamente il verificarsi dell'altro.

Questa classificazione è abbastanza facile da comprendere e centinaia sono gli esempi quotidiani che seguono questa classificazione. E' sufficiente pensare al lancio di un dado: ogni evento esclude categoricamente gli altri. Avendo un dado sei facce e potendo uscire solo una faccia alla volta è chiaro che se esce un numero non escono i rispettivi cinque complementari. L'esempio dei dadi è solo uno dei tanti esempi che si possono fare per illustrare la compatibilità degli eventi. Si possono individuare innumerevoli altri casi in cui un evento è in grado di escluderne altri. La rappresentazione in formule è la seguente:

$$A \cup B = \Omega$$

Il verificarsi dell'evento *A* e dell'evento *B* ha come probabilità l'insieme di tutti gli eventi possibili indicato con Ω. Se l'unione di due eventi fornisce l'evento certo è deducibile che i due eventi sono complementari ed il verificarsi di uno esclude il verificarsi dell'altro. Per maggior completezza si può scrivere la stessa espressione in una forma equivalente:

$$A \cap B = \emptyset$$

Tale formulazione è del tutto equivalente alla precedente, differisce solo per il modo in cui è stata scritta. Il verificarsi congiuntamente dell'evento *A* e dell'evento *B* ha come probabilità l'insieme vuoto che rappresenta l'evento impossibile. Questo è proprio dovuto al fatto che i due eventi sono incompatibili.

2.1.2 Elementi di matematica attuariale applicata agli eventi

La prima parte della trattazione ha preso in considerazione le classificazioni che gli eventi possono assumere in ambito generale esaminando tutti quegli aspetti che sono essenziali per una completa introduzione agli eventi. Ciò che si desidera fare ora riguarda l'analisi matematica dei principali concetti aleatori. Cosa sono i numeri aleatori, quali sono i loro parametri più significativi ed quali sono le principali distribuzioni di probabilità usate in ambito statistico. Per rendere la trattazione chiara ed ordinata, questi tre argomenti sono trattati separatamente.

2.1.2.1 Numeri aleatori

Indicato con Ω l'insieme di eventi elementari possibili, si definirà *numero aleatorio* ogni funzione $y = X(\omega)$ definita su Ω che soddisfa la seguente proprietà:

$$\{\omega \mid X(\omega) = x\}$$

Un numero aleatorio è una funzione che attribuisce a ciascun evento la relativa probabilità di verificarsi. A titolo di esempio si può citare il caso relativo al lancio di due dadi: la funzione che in queste circostanze è maggiormente presa in considerazione è la somma dei valori mostrati dalle facce dei due dadi. Nulla vieterebbe di applicare all'insieme universo altri tipi di funzioni come ad esempio la differenza dei due valori oppure il loro prodotto. Il motivo per cui si è preferito optare per la somma si riconduce all'esigenza di mantenere una certa uniformità con le regole socialmente note. I numeri aleatori si dividono in due categorie: numeri aleatori *discreti* e numeri aleatori *continui*.

1. Numeri aleatori discreti

Un numero aleatorio *X* è definito *discreto* quando l'insieme dei valori che può assumere è finito oppure è rappresentato da un'infinità numerabile di valori. In forma analitica:

$$R_x = \{x_1, x_2, ..., x_N\}$$

Per descrivere le proprietà di un numero aleatorio discreto, occorre introdurre un ulteriore elemento che in matematica attuariale prende il nome di *funzione di probabilità*.

Si definisce ***funzione di probabilità*** di un numero aleatorio discreto quella funzione definita per ogni *x* tale per cui:

$$p(x_i) = \Pr\{X = x_i\} \qquad \text{con} \qquad p(x_i) \geq 0$$

La funzione di probabilità associa ad ogni evento $X=x_i$ la relativa probabilità di verificarsi. La somma di tutte le probabilità associate ai singoli eventi è pari ad uno. Questo caratteristica è necessaria affinché sussista coerenza con quanto esposto riguardo la distinzione tra eventi certi ed incerti. Si è detto che l'insieme di tutti gli eventi

elementari di uno spazio Ω deve corrispondere all'evento certo e tale evento ha per definizione probabilità pari ad uno. In formule:

$$\sum_{i=1}^{N} p(x_i) = 1$$

Accanto alla funzione di probabilità di un numero aleatorio, si trova un'altra importante funzione che integra le caratteristiche della distribuzione: la *funzione di ripartizione*. Dato un numero aleatorio discreto, è definita ***funzione di ripartizione*** quella funzione *y=F(x)* definita per ogni *x* reale tale per cui:

$$F(x_i) = \Pr\{X \le x_i\} \le 1$$

L'ordinata della funzione di ripartizione calcolata in ciascun punto *x* indica la probabilità con cui il numero aleatorio *X* assume valori non maggiori di *x*. In forma analitica:

$$F(x_i) = \sum_{n=0}^{i} \Pr\{X = x_n\} \le 1$$

Le funzioni di probabilità e di ripartizione di un numero aleatorio discreto sono rappresentate graficamente nel seguente modo:

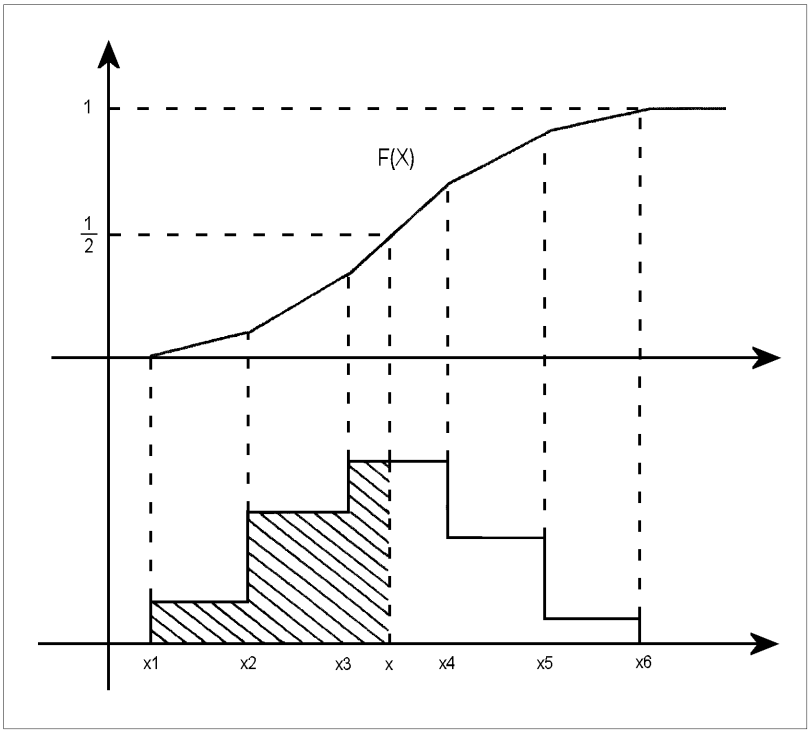


Figura 4 - Funzione di probabilità e di ripartizione di un numero aleatorio discreto

Sull'asse delle ascisse sono riportati i singoli eventi mentre sull'asse delle ordinate sono espresse le rispettive probabilità. Per calcolare la funzione di ripartizione in qualunque punto *x* appartenente all'insieme degli eventi possibili, è sufficiente sommare le probabilità di tutti i punti x_i relativi agli eventi minori o uguali ad *x*. La funzione di ripartizione assume un'importante particolare in quanto permette di ottenere anche altre importanti informazioni sulla distribuzione di probabilità. La differenza di una funzione di ripartizione calcolata in due punti diversi permette di ottenere informazioni sulla probabilità che si verifichino tutti gli eventi compresi tra i due presi in considerazione. In forma analitica:

$$\Pr\{a < X \leq b\} = F(b) - F(a)$$

2. Numeri aleatori continui

La differenza concettuale tra i numeri aleatori discreti ed i numeri aleatori continui è concettualmente nulla. Mentre nel primo caso si ha un insieme finito di elementi, nel secondo caso ci si trova di fronte ad un insieme infinito di eventi. Un numero aleatorio *X* è continuo quando può assumere tutti i valori in campo reale ed esiste una funzione *y=f(x)* tale per cui:

$$f(x) \geq 0$$
$$\int_{-\infty}^{+\infty} f(x) \cdot dx = 1$$
$$\Pr\{a < X \leq b\} = \int_a^b f(x) \cdot dx$$

La *prima* espressione rappresenta il segno della distribuzione della probabilità; il fatto che questa debba essere maggiore o uguale a zero deriva implicitamente dalla natura degli eventi che non possono per definizione avere probabilità negativa. La *seconda* espressione identifica ciò che nel discreto era rappresentato da tutto l'insieme degli eventi elementari Ω; anche in questo caso l'insieme di tutte le probabilità associate agli eventi deve coincidere con l'evento certo avente per definizione probabilità pari ad uno. Nel continuo, tale insieme non può che essere rappresentato dall'integrale tra meno infinito e più infinito della funzione di densità. La *terza* espressione rappresenta infine la probabilità che si verifichi un evento compreso all'interno di un dato intervallo. Trattandosi del continuo, tale probabilità sarà data dall'integrale definita tra *a* e *b* della funzione di densità; il caso è analogo al discreto e l'unica differenza si riscontra nell'utilizzo dell'operatore integrale al posto dell'operatore sommatoria.

La funzione *y=f(x)* appena esaminata prende il nome ***funzione di densità di probabilità*** del numero aleatorio continuo *X* ed è del tutto analoga alla relativa funzione di probabilità esaminata nel caso discreto. Il fatto che assuma la denominazione "densità" riguarda direttamente il campo di applicazione che in tal caso è continuo. Una particolare osservazione merita di essere esposta. Dalla definizione stessa di numero aleatorio continuo, per ogni punto x_0 della funzione di densità si ha che:

$$\Pr\{X = x_0\} = 0$$

La probabilità associata a ciascun punto *x* della funzione di densità è pari a zero. Questa nozione, sebbene possa risultare assurda, rappresenta un aspetto fondamentale del calcolo combinatorio nel continuo e risulta estremamente ovvia se si richiamano alcuni concetti di matematica differenziale. Nel continuo, la probabilità di un singolo evento può anche essere espressa mediante un'integrale della funzione di densità tra x_i ed x_i stesso. In formule:

$$\Pr(x_i) = \int_{x_i}^{x_i} f(x) \cdot dx$$

Dalla formulazione si intuisce chiaramente il motivo per cui la probabilità di ogni singolo evento elementare è nulla: l'integrale di una funzione continua calcolato all'interno di un intervallo nullo da come risultato zero per cui:

$$\Pr(x_i) = \int_{x_i}^{x_i} f(x) \cdot dx = 0$$

Come calcolare dunque la probabilità che un determinato evento accada se tale evento ha probabilità nulla? E' sufficiente calcolare la probabilità all'interno di un intervallo infinitamente piccolo nell'intorno del punto considerato; solo in tal caso il valore della probabilità avrà un senso e potrà essere utilizzato per altre implicazioni matematiche.

Trattando i numeri aleatori discreti, si è parlato di una particolare funzione che prende il nome di *funzione di ripartizione*. Anche i numeri aleatori continui sono dotati di questa funzione.

Si definisce ***funzione di ripartizione*** di un numero aleatorio continuo *X* quella funzione *y=F(x)* tale per cui:

$$F(x_i) = \Pr\{X \le x_i\} = \int_{-\infty}^{x_i} f(x) \cdot dx$$

La funzione di ripartizione calcolata in un punto *x* fornisce la probabilità che il numero aleatorio assuma un valore compreso tra meno infinito e x_i. Il caso è del tutto analogo a quello esaminato nel discreto con l'unica eccezione che, mentre nel caso precedente si utilizzava l'operatore *sommatoria*, ora occorre utilizzare l'operatore *integrale*. Di seguito è riportata una funzione di densità con relativa funzione di ripartizione entrambe espresse nel continuo.

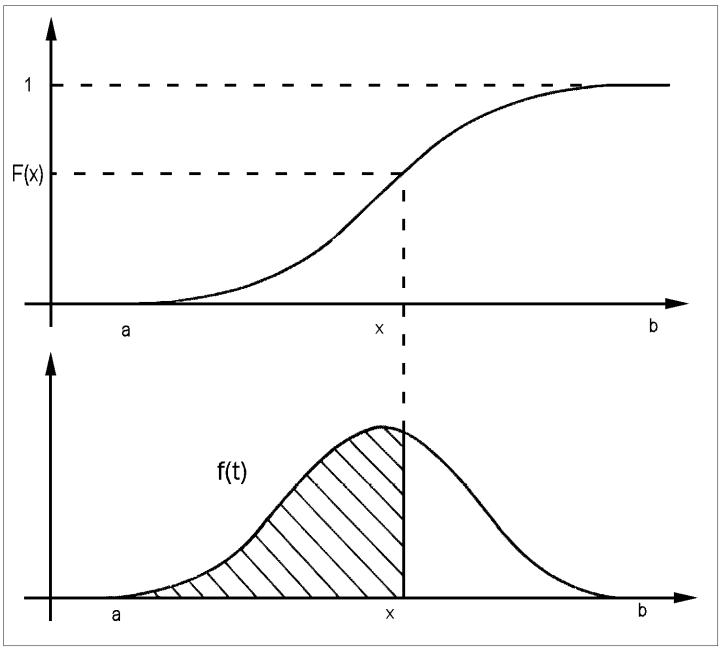


Figura 5 - Funzione di densità e di ripartizione di un numero aleatorio continuo

Anche nel caso di numeri aleatori continui è possibile utilizzare la funzione di ripartizione per calcolare la probabilità che si verifichino tutti gli eventi compresi all'interno di un determinato intervallo.

$$\Pr\{a < X \le b\} = F(b) - F(a)$$
$$\int_a^b f(x) \cdot dx = \int_{-\infty}^{b} f(x) \cdot dx - \int_{-\infty}^{a} f(x) \cdot dx$$

Questa operazione è usata molto di rado in quanto lo stesso risultato è calcolabile direttamente mediante l'integrale definita tra i due valori estremi.

2.1.2.2 Parametri significativi

Finora si è analizzato qual è il significato di numero aleatorio e si sono analizzati i due principali tipi di numeri aleatori con le relative caratteristiche e differenze matematiche e concettuali. Senza la conoscenza di alcuni parametri essenziali che caratterizzano una distribuzione di probabilità, difficilmente si riuscirà a trarre da questa informazioni utili. Per poter utilizzare al meglio una funzione di distribuzione è innanzitutto necessario ricavare da essa alcuni parametri che forniscano maggiori informazioni sulle sue caratteristiche. Questi parametri rappresentano la carta d'identità di una distribuzione: nonostante non forniscano un'informazione completa, sono in grado di individuare con una discreta precisione l'appartenenza della distribuzione ad una particolare categoria. Questi parametri sono la ***Speranza matematica*** detta anche *Valore atteso* o più comunemente *Media*, la ***Mediana***, la ***Varianza*** anche nella sua versione lineare chiamata *Deviazione standard* ed il ***Coefficiente di correlazione***, che presenta molte affinità sia con la *Varianza* che con la *Covarianza*, altro fondamentale parametro che sarà analizzato in seguito.

1. Speranza Matematica

Dato un numero aleatorio $X(\omega)$ definito su di un insieme Ω, prende il nome di speranza matematica la quantità:

$$E(X) = \sum_{i=1}^{n} X(\omega_i) \cdot p\{\omega_i\}$$

La speranza matematica rappresenta la media ponderata dei valori associati ai singoli eventi utilizzando come pesi le probabilità attribuite ai singoli eventi stessi. La speranza matematica esprime il valore che ci si dovrebbe attendere dal numero aleatorio, da qui l'attribuzione del nome *valore atteso*. Una cosa del tutto analoga emerge se si analizza il caso nel continuo. La logica sottostante è esattamente identica e, come avvenuto in precedenza, è sufficiente sostituire l'operatore sommatoria con il più appropriato operatore integrale.

Dato un numero aleatorio X continuo, associato ad una funzione di densità di probabilità $f(x)$, sarà definita speranza matematica di X la quantità:

$$E(X) = \int_{-\infty}^{+\infty} x \cdot f(x) \cdot dx$$

Ciò che prima era indicato con il termine di ω_i, ora è rappresentato dal termine *x* mentre le singole probabilità $p(\omega_i)$, sono ora rappresentate da *f(x)* integrato per *dx*. Ciò che è stato finora esposto merita qualche approfondimento soprattutto per quanto riguarda le principali proprietà della speranza matematica. Le principali sono quattro:

a) La speranza matematica di un numero aleatorio *X*=*c* costante coincide con la costante stessa:

$$E(c) = c$$

b) La speranza matematica di un numero aleatorio premoltiplicato per una costante è pari al numero aleatorio moltiplicato per la costante stessa.

$$E(c \cdot g(x)) = c \cdot E(g(x))$$

c) La speranza matematica della combinazione lineare di due numeri aleatori coincide con la somma delle speranze matematiche dei due numeri aleatori premoltiplicate per le relative costanti.

$$E(aX + bY) = aE(X) + bE(Y)$$

d) Se due numeri aleatori *X* ed *Y* sono indipendenti, la speranza matematica del loro prodotto è data dal prodotto delle singole speranze matematiche.

$$E(X \cdot Y) = E(X) \cdot E(Y)$$

2. Mediana

Si definisce *mediana* di un numero aleatorio *X* con funzione di ripartizione *F(x)* il valore x_0 tale per cui siano soddisfatte le due disuguaglianze:

$$\Pr\{X \le x_0\} \ge \frac{1}{2} \qquad \textit{e} \qquad \Pr\{X \ge x_0\} \ge \frac{1}{2}$$

Nel caso esistano più valori per cui valgano i seguenti vincoli, tutti i punti che rispettano tali vincoli prenderanno il nome di *valori mediani*. Il *valore mediano* o *mediana* è quel valore che ripartisce in due esatte metà la funzione di ripartizione; rappresenta quel particolare evento tale per cui la probabilità che accada un evento inferiore o un evento superiore è esattamente la stessa pari quindi a 0.5.

3. Varianza

Dato un numero aleatorio *X*, si definisce varianza quella quantità:

$$\sigma^2(X) = V(X) = E(X - E(X))^2 = E(X^2) - E^2(X) = \mu_2 - \mu_1^2$$

Quella che prima è stata definita deviazione standard non è altro che la radice quadrata della varianza indicata con *σ(x)*. La varianza rappresenta la media degli scarti al quadrato. In termini più complessi, la varianza può anche essere descritta come il momento secondo meno il quadrato del momento primo del numero aleatorio. Nel caso discreto si ha:

$$Var(X) = \sum_{i=1}^{n} p_i (x_i - \bar{x})^2$$

La varianza, insieme alla media aritmetica, rappresenta uno dei più importanti parametri di analisi statistica. Da un punto di vista concettuale, la varianza rappresenta la dispersione di una serie di valori attorno alla loro media definita speranza matematica o valore atteso. La varianza può assumere solamente valori positivi o nulli, mai valori negativi. Una varianza molto piccola rappresenta una piccola dispersione intorno alla media; una varianza molto grande indica una maggiore dispersione dei dati rispetto alla stessa.

Si può ora procedere con l'analisi delle principali proprietà della varianza, proprietà che possono principalmente essere sintetizzate in quattro punti:

a) La varianza di un numero aleatorio addizionato ad una costante è pari alla varianza del numero aleatorio stesso.

$$V(X + c) = V(X)$$

b) La varianza di un numero aleatorio moltiplicato per una costante è pari al quadrato della costante moltiplicato per la varianza del numero aleatorio stesso.

$$V(c \cdot X) = c^2 \cdot V(X)$$

c) La varianza di un numero aleatorio che è trasformazione lineare di un altro numero aleatorio da esso derivante è pari al quadrato del coefficiente premoltiplicato al primo numero aleatorio per la varianza dello stesso numero aleatorio.

$$V(Y) = V(a \cdot X + b) = a^2 \cdot V(X)$$

d) La varianza della somma di due numeri aleatori stocasticamente indipendenti è pari alla somma delle rispettive varianze.

$$V(X + Y) = V(X) + V(Y)$$

La varianza, a differenza della speranza matematica, non è quasi mai additiva; l'unico caso in cui può essere considerata come tale è proprio in presenza di due numeri aleatori indipendenti.

4. Covarianza

La *covarianza* è la varianza misurata congiuntamente su due numeri aleatori distinti. Analiticamente:

$$Cov(X, Y) = E(X \cdot Y) - E(X) \cdot E(Y)$$

La varianza assolve alla sua funzione indicando qual è la dispersione di una serie di dati rispetto alla loro media; la *covarianza* indica il legame che intercorre tra due numeri aleatori o due serie di dati. In ambito discreto, la covarianza è espressa come:

$$Cov(X, Y) = \sum_{i=1}^{n} p_{XY_i} \cdot (x_i - \overline{x}) \cdot (y_i - \overline{y})$$

A differenza della varianza, la *covarianza* può assumere valori negativi. L'assenza di funzioni quadratiche al suo interno, estende il campo di esistenza della *covarianza* a

tutto il campo reale, da meno infinito a più infinito. Da quanto appena detto, sorgono importanti osservazioni che meritano di essere esaminate con maggiore attenzione. Perché è possibile ottenere tanto valori positivi quanto valori negativi? Si è detto che la covarianza rappresenta il legame che intercorre tra due serie distinte di valori; se questo è vero, dovrebbe risultare abbastanza chiaro che due serie possono presentare affinità simili, del tutto opposte o addirittura assenti. Si prenda in considerazione l'andamento di due rette, una crescente e l'altra decrescente con la stessa inclinazione; entrambe hanno senz'altro una grossa caratteristica in comune, il fatto che entrambe siano rette ed abbiano entrambe la stessa inclinazione. Ciò che le differenzia è il segno dell'inclinazione; questa caratteristica fa si che la loro covarianza risulti decisamente negativa, giustificata proprio al diverso andamento che le due mostrano l'una nei confronti dell'altra. La covarianza rappresenta la somiglianza di due serie: tanto più elevata sarà la covarianza, tanto maggiore sarà il loro grado di somiglianza. Il segno della covarianza indicherà invece il tipo di legame: positivo o negativo.

5. Coefficiente di correlazione

Strettamente legato alla covarianza tra due serie di numeri o vettori aleatori, è il *coefficiente di correlazione*. Il *coefficiente di correlazione* è molto simile alla covarianza ma mentre la covarianza esprime il legame tra due serie in valore assoluto, il *coefficiente di correlazione* esprime lo stesso legame in termini relativi. Tramite la covarianza non si è in grado di determinare l'intensità con cui due serie sono legate tra loro; se si è ottenuta una covarianza positiva molto elevata, non disponendo di riferimenti relativi, non si potrà mai affermare con certezza che la relazione tra le serie è più o meno forte: ciò che in apparenza può sembrare una grande correlazione, può in realtà essere un legame del tutto insignificante. Per valutare la rilevanza della covarianza, occorrerà prima confrontarla all'ordine di grandezza del campione o, più precisamente, della serie: se tale serie appartiene ad un ordine di grandezza cento volte più elevato di quello emerso dalla covarianza, tale covarianza, sebbene positiva ed elevata, rappresenterà comunque qualcosa di poco significativo in tale contesto. Il *coefficiente di correlazione*, a differenza della covarianza, evidenzia il legame tra due serie di numeri in termini esclusivamente relativi fornendo con maggior chiarezza il livello di correlazione intercorrente tra i due. In termini matematici, il *coefficiente di correlazione* è indicato come:

$$\rho = \frac{Cov(X,Y)}{\sqrt{Var(X)\cdot Var(Y)}}$$

Sia il numeratore che il denominatore sono della stessa grandezza; il denominatore può solo essere positivo ed il segno di questa espressione dipende esclusivamente dal covarianza tra le due serie che è posta a numeratore. Se le due serie sono perfettamente identiche, il numeratore coinciderà con il denominatore ed il coefficiente di correlazione sarà pari a più uno. Se le due serie hanno invece valori perfettamente identici ma simmetrici, il numeratore avrà lo stesso valore in termini assoluti del denominatore ma con segno opposto; il coefficiente di correlazione varrà pertanto meno uno. Il campo di esistenza di questo parametro è compreso tra meno uno e più uno. In formule:

$$-1 \leq \rho \leq +1$$

Un caso particolare che merita una certa attenzione riguarda quello in cui il coefficiente di correlazione sia pari a zero. Tale situazione si verifica solo quando il legame tra due serie è del tutto assente. In tale situazione si parla di serie stocasticamente indipendenti. Il motivo per cui risulta molto più utile utilizzare il *coefficiente di correlazione* anziché la relativa covarianza, è legato all'evidenza matematica ad esso associata: la sola covarianza non è in grado di stabilire con precisione quale sia il grado di relazione tra più serie; il coefficiente di correlazione è perfettamente in grado di fornire con una certa precisione il livello di reciproca correlazione.

2.1.2.3 Principali distribuzioni di probabilità notevoli

Finora si è discusso della definizione di numeri aleatori e sono stati presi in considerazione i loro principali parametri di rilievo. Restano da esaminare quelle che prendono il nome di distribuzioni di probabilità notevoli. Esistono vari tipi di distribuzioni di probabilità notevoli e se ne possono enumerare almeno una decina. In questa sede sarebbe totalmente inappropriato analizzarle tutte e l'esposizione si limiterà ad analizzare solo quelle ritenute più importanti tanto sotto l'aspetto matematico quanto sotto l'aspetto pratico. Le distribuzioni notevoli che in questa sede sono prese in considerazione sono tre: la ***Distribuzione binomiale***, la ***Distribuzione di Poisson*** e la ***Distribuzione normale*** definita anche *Gaussiana*. Le prime due sono distribuzioni discrete mentre la terza è una distribuzione continua.

1. Distribuzione di probabilità binomiale

Dato un numero aleatorio *X*, si dice che questo si distribuisce con legge binomiale di parametri *N* maggiore o uguale ad uno intero e con probabilità *p* compresa tra zero ed uno se la sua funzione di probabilità è pari a:

$$\Pr\{X = x\} = p_X(x) = \binom{N}{x} p^x \cdot (1-p)^{N-x} \qquad \text{per } x=0,1,2,\ldots,N$$

La probabilità che il numero aleatorio *X* assuma il valore *x* dipende dalla probabilità associata al caso favorevole e da quella associata al caso non favorevole moltiplicato per il numero di combinazioni di *x* sullo spazio *N*. L'espressione *(1-p)* è spesso sostituita con la lettera *q*. Il parametro *(N x)* rappresenta le combinazioni dell'evento *x* sullo spazio *N*.

Così come è stato fatto per i principali parametri di valutazione statistica, si procede ora con l'esposizione delle principali caratteristiche matematiche della distribuzione binomiale. Sono due e si riferiscono rispettivamente alla media e alla varianza:

a) Il valore atteso di un numero aleatorio *X* distribuito con probabilità binomiale è pari al prodotto tra il numero di eventi e la probabilità a loro associata.

$$E(X) = E(\sum_{i=1}^{N} X_i) = \sum_{i=1}^{N} E(X_i) = N \cdot p$$

b) La varianza di un numero aleatorio *X* distribuito con probabilità binomiale è pari al prodotto tra il numero di eventi, la probabilità di evento favorevole e la probabilità di evento non favorevole a loro associata.

$$Var(X) = Var(\sum_{i=1}^{N} X_i) = \sum_{i=1}^{N} Var(X_i) = N \cdot p(1-p)$$

2. Distribuzione di Poisson

Dato un numero aleatorio discreto *X*, si dice che questo segue la legge di Poisson con media μ>0 se:

$$\Pr\{X = x\} = \frac{\mu^x \cdot e^{-\mu}}{x!} \qquad \textit{con } x = 0,1,2,\ldots$$

Anche la *distribuzione di Poisson* è una distribuzione discreta che può essere applicata solamente ai numeri interi. Se si considera che $N*p=\mu$, la distribuzione binomiale può essere ricondotta alla distribuzione di *Poisson*. In formule:

$$\binom{N}{x} \cdot p^x \cdot (1-p)^{N-x} \approx \frac{(N \cdot p)^x}{x!} \cdot e^{-N \cdot p} = \frac{\mu^x}{x!} \cdot e^{-\mu}$$

Questa proprietà rappresenta un ottimo strumento per correggere eventuali carenze modellistiche: là dove non può essere applicata la legge di distribuzione binomiale, la legge di Poisson rappresenta un ottimo strumento sostitutivo.

Si procede ora con l'analisi delle principali caratteristiche statistiche:

a) Il valore atteso di un numero aleatorio *X* distribuito secondo la legge di Poisson è pari alla sua media aritmetica.

$$E(X) = \sum_{x=0}^{\infty} x \cdot \frac{\mu^x \cdot e^{-\mu}}{x!} = \mu$$

b) La varianza di un numero aleatorio *X* distribuito secondo legge di Poisson è pari al suo valore atteso o media aritmetica.

$$V(X) = \sum_{x=0}^{\infty} x^2 \cdot \frac{\mu^x \cdot e^{-\mu}}{x!} - \mu^2 = \mu$$

Quanto appena esposto rende notevolmente più facile intuire l'enorme importanza che questa distribuzione riveste in ambito statistico. Una distribuzione con valore atteso e varianza uguali entrambi pari a μ semplificano notevolmente il lavoro di analisi statistica sottostante e permettono di ottenere semplici conclusioni senza dover ricorrere a complessi calcoli matematici che il più delle volte possono anche rilevarsi errati.

3. Distribuzione normale

La *distribuzione normale*, definita anche *Gaussiana* dal nome del suo studioso, è quella che maggiormente trova applicazione nell'analisi statistica. Le ragioni principali sono localizzate nelle sue caratteristiche:

- E' in grado di approssimare numerose altre distribuzioni quali la binomiale e la stessa distribuzione di Poisson.
- Possiede proprietà matematiche che la rendono particolarmente flessibile in diversi campi di applicazione.
- La distribuzione normale è una distribuzione continua ed il suo campo di esistenza si estende su tutto il campo reale, da meno infinito a più infinito.

Dato un numero aleatorio *X* si dice che è distribuito con legge di probabilità normale con media μ e deviazione standard σ se possiede funzione di densità pari a:

$$f(x,\mu,\sigma) = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{(-(\frac{x-\mu}{\sigma})^2)} \qquad -\infty < x < +\infty$$

La principale caratteristica di una distribuzione normale è quella di essere simmetrica rispetto alla sua media e la classica forma a campana che ne scaturisce dipende fortemente dalla sua varianza. La funzione di densità di una distribuzione normale è rappresentata dal seguente grafico.

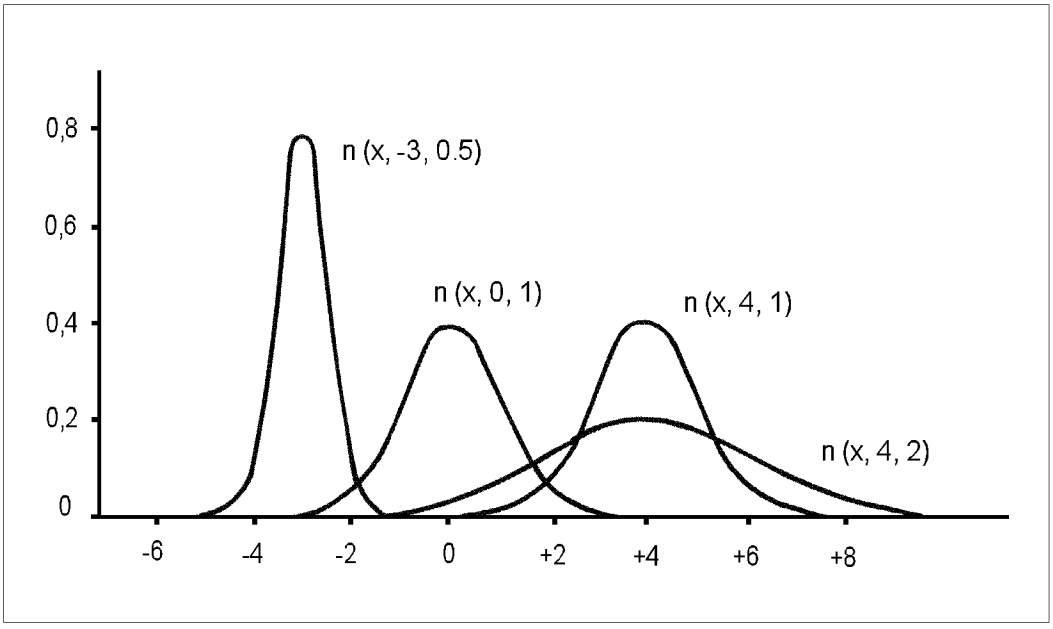


Figura 6 - Funzioni di densità distribuite con legge normale

La distribuzione avente maggiore varianza è caratterizzata da una forma molto schiacciata (distribuzione di destra) mentre la distribuzione con minor varianza è caratterizzata da una forma decisamente più allungata e mostra una maggiore concentrazione di valori attorno alla propria media.

.

La notevole importanza della *distribuzione normale* risiede nell'elevata flessibilità che questa dimostra nel saper rappresentare anche altre distribuzioni tanto utili quanto utilizzate. In questa sede non è possibile addentrarsi troppo in questo argomento strettamente statistico che non rappresenta il tema principale di questa trattazione; la ragione per cui si sono prese in considerazione le tre principali distribuzioni notevoli va ricercato nell'esigenza di fornire un quadro teorico completo adeguato al contesto su cui si fonda questa trattazione. Ora che le necessarie nozioni di matematica attuariale sono state illustrate, si può procedere ad una specifica analisi degli eventi in ambito strettamente borsistico.

2.2 Eventi in ambito borsistico

Finora si è discusso degli eventi in un contesto puramente matematico e gli aspetti che sono stati presi in considerazione riguardavano i concetti di evento da un punto di vista strettamente statistico. Per poter affrontare una trattazione tecnica ed informatica sugli eventi in ambito borsistico è necessario passare in rassegna i loro principali aspetti teorici. Innanzitutto occorre fornire una possibile definizione di evento.

Può essere definito *evento borsistico* qualunque fatto, notizia od azione che, tramite l'informazione pubblica o privata, viene a conoscenza di più soggetti che operano sui mercati borsistici incentivandoli a modificare le proprie posizioni finanziarie detenute.

Tutti i giorni la società viene a conoscenza di nuove informazioni e qualunque mezzo di è valido per ottenere informazioni e notizie sui fatti che caratterizzano i mercati finanziari: è sufficiente pensare ai quotidiani, ai telegiornali, alla radio. Di tutte le notizie la cui collettività viene a contatto, alcune sono considerate positive, altre negative. Gli elementi che determinano l'imputazione di ciascun evento all'una o all'altra categoria sono molteplici e saranno presi in considerazione più avanti. Mentre le notizie considerate positive spingono normalmente i mercati verso l'alto, le notizie intese come negative, al contrario, tendono a comprimere i mercati verso il basso accentuando talvolta le perdite degli investitori.

Una caratteristica determinante che deve essere presa in considerazione quando si valuta un evento è la sua sfera di azione. Alcuni eventi coinvolgono l'intero sistema finanziario mondiale, altri si limitano ad interessare solo una parte di esso mentre altri ancora possono coinvolgere un singolo settore di mercato se non addirittura un singolo titolo quotato.

Ciò che si vuole perseguire con questa trattazione risponde pertanto alle seguenti domande: "Quali sono gli effetti che gli eventi e le notizie generano sul sistema borsistico globale? Gli effetti che ne derivano si ripercuotono su tutto il mercato oppure solo su una parte di esso? L'entità degli effetti dipende solo dalle loro caratteristiche endogene o anche da altri elementi esogeni come la razionalità dei soggetti operanti e le informazioni disponibili sul mercato? Esiste un modo per gestire i rischi derivanti dagli eventuali effetti negativi generati dagli eventi?"".

In questa parte della trattazione si cercherà di fornire una risposta a tutte queste domande. Innanzitutto si passeranno in rassegna i principali elementi che influenzano in modo più o meno rilevante l'effetto degli eventi sui mercati. Tra questi saranno presi in considerazione le categorie di soggetti che operano su mercati finanziari, la razionalità degli stessi soggetti, nonché le problematiche emergenti dalle asimmetrie informative. In seguito si passerà ad una analisi strettamente matematica mediante la quale si cercherà di fornire una chiara spiegazione su come il rischio di un titolo possa essere scomposto nelle sue due componenti denominate *rischio sistematico* e *rischio specifico*. Per ultimo si prenderanno in considerazione i principali strumenti economico finanziari utilizzati per ridurre la rischiosità di un investimento. Tra questi saranno citati la diversificazione di portafoglio mediante costruzione di una frontiera efficiente e l'implementazione di modelli rappresentativi del mercato come il *Capital Asset Pricing Model* e lo *Stock Index Model*.

2.2.1 Elementi determinanti di rilevanza borsistica

Molti sono gli elementi che in modo più o meno accentuato influenzano l'effetto degli eventi. I principali elementi dotati di questa caratteristica sono localizzati tra gli elementi costitutivi dei mercati stessi.

- Il ***primo*** elemento determinante è certamente costituito dalla *categoria dei soggetti* operanti. La differenza che intercorre tra un modesto operatore

individuale ed un considerevole investitore istituzionale è di notevole rilevanza non solo per quanto riguarda l'aspetto sostanziale e giuridico delle due figure, ma anche per la notevole entità ed effetto delle loro azioni. Un singolo operatore individuale è in grado di acquistare o vendere un modesto quantitativo di titoli mentre un investitore istituzionale, disponendo di maggiori fondi, è maggiormente incentivato a trasferire in acquisto o vendita maggiori quantità di capitali.

- Il ***secondo*** elemento essenziale da considerare nella valutazione degli eventi è il problema della *razionalità dei soggetti*, problema che per certi aspetti si riconduce ad un altro aspetto rilevante noto col nome di *comportamenti imitativi*.

- Il ***terzo*** elemento da non trascurare è il problema delle *asimmetrie informative*. Le asimmetrie informative sorgono quando nel mercato le informazioni non si distribuiscono in maniera uniforme presso tutti gli operatori cosicché alcuni disporranno di maggiori informazioni rispetto ad altri. Il problema delle asimmetrie informative, assieme a quello della razionalità e dei soggetti operanti, rappresenta uno dei principali motivi di instabilità globale del sistema.

Esiste tra questi un elemento che dispone di maggiore influenza rispetto agli altri oppure incidono tutti in egual misura sui mercati? La maggiore o minore quantità di grandi operatori sui mercati altera notevolmente il loro assetto oppure è del tutto ininfluente? Quali ruoli assumono la razionalità ed i comportamenti imitativi sull'andamento dei mercati? La trattazione di questo paragrafo vertirà sull'analisi di ciascun elemento per individuare la possibile influenza che ognuno di essi ha sui mercati.

2.2.1.1 Categorie dei soggetti operanti

La distribuzione delle varie categorie di soggetti operanti sui mercati rappresenta un elemento determinante per comprendere come questi si muovano e possano modificare, con le proprie condizioni, l'assetto globale dei mercati.

E' noto che i prezzi all'interno di un mercato si formano in base all'incontro tra domanda ed offerta. I prezzi che si formano sui mercati finanziari, oltre a seguire il principio base appena enunciato, riflettono i contratti di acquisto e vendita

effettivamente conclusi durante la giornata, pesati sulle effettive quantità scambiate. Dal funzionamento dei mercati discende la notevole importanza che i soggetti operanti hanno nelle loro capacità di trasferire maggiori o minori quantità di valore sui canali finanziari. L'incidenza introdotta da un operatore individuale sarà certamente inferiore a quella introdotta da un operatore maggiore quale ad esempio un società finanziaria od un investitore istituzionale. Dall'incidenza di un operatore dipende il suo grado di influenza ed un operatore individuale avrà certamente sul mercato un'influenza minore rispetto ad un operatore maggiore. Ma quali sono allora i principali soggetti che operano sui mercati? Si possono individuare principalmente tre categorie di soggetti: gli *operatori individuali*, le *società finanziarie* e gli *investitori istituzionali*.

1. Operatori individuali

Gli *operatori individuali* rappresentano una categoria che solo da pochi anni a questa parte ha assunto un ruolo fondamentale all'interno del sistema finanziario. Anni addietro questa categoria occupava una posizione decisamente più marginale ed il motivo principale era localizzato nelle scarse possibilità che gli operatori individuali avevano di accedere ai mercati finanziari. L'unico mezzo concesso loro era la negoziazione indiretta tramite altri operatori finanziari dedicati quali banche o società di gestione finanziaria. Questa notevole limitazione ha sempre avvantaggiato le altre categorie di operatori definite maggiori conducendo ad una notevole limitazione delle reazioni sui mercati. Grazie al notevole sviluppo tecnologico che ha coinvolto l'intero sistema mondiale, gli strumenti messi a disposizione a questa categoria di soggetti sono aumentati notevolmente e quello di maggior rilievo riguarda certamente lo sviluppo di Internet.

L'avvento di Internet ha mutato radicalmente le condizioni di vita sociale e ciò che prima sembrava impossibile o dai costi proibitivi, è oggigiorno di uso abituale con modesti costi di gestione. Con l'avvento di Internet, la maggioranza delle società finanziarie ha sviluppato alcuni strumenti che permettono ai singoli clienti che lo desiderano un accesso diretto ai mercati finanziari. Tali strumenti consentono ai privati cittadini di operare sui mercati finanziari direttamente da casa tramite il proprio Personal Computer. Chi lo desidera, può acquistare o vendere prodotti finanziari in maniera rapida e, grazie ai nuovi strumenti di codifica in rete, sempre più sicura. Chiunque da casa può gestire direttamente il proprio portafoglio titoli con tutta comodità senza doversi recare in banca o dal proprio gestore finanziario. Queste innovazioni tecnologiche non hanno però solo cambiato il stile di vita quotidiano;

indirettamente hanno influenzato anche l'andamento dei mercati stessi. La presenza sempre più accentuata di operatori individuali sui mercati, ha aumentato notevolmente la variabilità degli stessi generando sempre più di frequente andamenti altalenanti e bolle speculative.

2. Intermediari finanziari

Gli intermediari finanziari rappresentano la categoria di soggetti maggiormente presente sui mercati finanziari. Prima che l'avvento di Internet favorisse lo sviluppo e la partecipazione degli operatori individuali ai mercati finanziari, gli intermediari finanziari rappresentavano con le proprie operazioni quasi l'intero mercato. Dopo l'avvento di Internet la loro presenza sui mercati si è leggermente ridotta lasciando notevole spazio all'iniziativa privata.

Nonostante la maggior presenza diretta dei privati cittadini sui mercati finanziari, gli intermediari finanziari continuano ad avere un ruolo centrale nell'interno sistema finanziario globale. Grazie all'operato di queste società, i mercati sono continuamente riforniti della liquidità di cui hanno bisogno evitando così periodi di ristagno e problemi di scarsa efficienza allocativa delle risorse. Non tutte gli intermediari contribuiscono ovviamente in egual modo alla liquidità dei mercati e, mentre alcuni fanno della loro presenza sul mercato la funzione principale della loro attività, altri utilizzano i mercati finanziari solo per scopi secondari che normalmente hanno finalità speculative o gestionali. La prima categoria citata presenta una notevole rilevanza e gli intermediari appartenenti a questa categoria sono spesso definiti *Market makers*.

I principali intermediari finanziari presenti sui mercati sono: le *banche*, le *società di gestione del risparmio* (*SGR*), le *società di intermediazione mobiliare* (*SIM*) e le *assicurazioni*.

2.a Banche

Le *banche* rappresentano certamente le società finanziarie di maggiore rilievo sia per i privati cittadini che per le imprese. Molte sono le funzioni delle banche ed altrettante sono quelle già note di cui risulta piuttosto inutile fornire ulteriori dettagli. Oltre alla funzione principale di acquisire depositi dalla clientela, ogni banca ha l'obbligo di specializzarsi in una particolare attività finanziaria. Su questa classificazione è

possibile distinguere tre categorie di banche: *Commercial banks*, *Investment banks* e *Merchant banks*.

- Le ***Commercial banks*** sono quelle banche la cui attività principale è orientata al soddisfacimento di una clientela marginale che può essere definita al dettaglio. La principale attività di queste banche è prevalentemente circoscritta a quello che viene comunemente denominato *credito al consumo*. Con questo termine si indicano tutte le attività volte a prestare denaro ai piccoli consumatori che si trovano temporaneamente in condizioni di necessità. A queste tipologie di banche non è normalmente imposta una particolare presenza sui mercati finanziari; questi sono generalmente utilizzati per ottimizzare eventuali eccedenze di gestione o per compensare brevi squilibri finanziari temporanei. La presenza delle *commercial banks* si fa sentire soprattutto sui mercati monetari ed obbligazionari ma molto meno su quelli azionari.

- Le ***Investment banks***, a differenza delle *commercial banks*, appartengono ad una categoria di intermediari finanziari che normalmente svolge un'ampia gamma di attività ad elevato contenuto di servizio personalizzato per conto di una clientela di rango più elevato come altre società o grandi investitori. Come suggerisce la terminologia stessa, l'attività principale di queste banche è rivolta soprattutto al collocamento delle risorse sui mercati finanziari. Tra le principali attività delle *investment banks* vanno certamente annoverate quella *creditizia*, tramite la quale le banche partecipano al finanziamento di una società privata sia come organizzatori che come finanziatori diretti, quella di *corporate finance* che si occupa di tutte le operazioni relative a particolari operazioni finanziarie tra imprese quali le fusioni o le acquisizioni di partecipazioni, quella di *capital markets* orientata allo scambio di valori mobiliari su mercati secondari, sul mercato dei cambi e su quello dei derivati ed infine l'attività di *asset management* tramite la quale le banche si occupano della gestioni di cassa nonché dei patrimoni delle singole società non finanziarie che lo richiedono. L'attività che fa di questa categoria di banca il principale operatore bancario presente sui mercati finanziari è certamente quella di *capital markets*. Grazie a questa attività, le *investment banks* rappresentano probabilmente la categoria bancaria con maggiore presenza sul mercato.

- Le ***Merchant banks*** rappresentano una categoria particolare di banche. La loro funzione è orientata all'attività specialistica di acquisizione temporanea di

partecipazioni azionarie nel capitale di rischio di imprese normalmente non finanziarie. Le *merchant banks*, assieme alle *investment banks*, rappresentano le due categorie di banche con maggiore presenza ed influenza sui mercati finanziari.

2.b Società di gestione del risparmio

Le *società di gestione del risparmio* si differenziano notevolmente dalle banche per il numero di servizi offerti. Mentre le banche offrono una notevole varietà di servizi che si adeguano pressoché a tutte le esigenze della clientela, le società di gestione del risparmio focalizzano la loro attenzione sull'attività di investimento dei fondi che queste ricevono dai propri clienti. Lo scopo principale di queste società è individuare un portafoglio ottimale di titoli nel quale si possono investire i capitali che i privati cittadini o altre società finanziarie hanno dato loro affinché possano esser impiegati nel modo più redditizio possibile. Lo strumento principale utilizzato dalle società di gestione del risparmio (*SGR*) per ottemperare a tali scopi è quello di creare nuove forme di investimento definite finanziariamente *fondi comuni di investimento*. Nel caso la clientela lo desiderasse, è anche possibile attuare dei piani di investimento personalizzati, specifici per il singolo investitore; il tutto a scapito di maggiori costi di gestione posti a carico del cliente.

Questa attività richiede diversi studi e ricerche per ottenere buoni risultati che garantiscano elevate prestazioni e bassi rischi. I principali strumenti utilizzati normalmente dalle *SGR* saranno analizzati in seguito quando si parlerà della teoria di portafoglio. La necessità di correggere e aggiornare continuamente i portafogli titoli propri e dei propri clienti, queste società garantiscono un presenza costante e consistente sui mercati finanziari. Le quantità dei titoli trattate dalle società di gestione del risparmio rappresenta normalmente una percentuale piuttosto alta di tutto il quantitativo scambiato ed il grado di liquidità che queste garantiscono al mercato non passa inosservato.

2.c Società di intermediazione mobiliare

Le *società di intermediazione mobiliare* rappresentano una categoria particolare di intermediari finanziari. Le *SIM*, a differenza degli altri intermediari finanziari come le banche, le società di gestione del risparmio o le assicurazioni, sono caratterizzate dalla mancanza di un'attività finanziariamente più rilevante rispetto alle altre in gestione. A

queste società sono attribuite parte delle funzioni delle banche e parte delle attività delle società di gestione del risparmio. Tra le maggiori attività delle *SIM* si trova quella di negoziazione di strumenti finanziari per conto proprio (*dealing*) o per conto di terzi (*brokerage*), quella che si occupa del collocamento di strumenti finanziari con garanzia (*underwriting*) o senza garanzia (*selling*) di sottoscrizione a favore dell'emittente, la gestione individuale di portafogli nonché la trasmissione e la ricezione di ordini di negoziazione di strumenti finanziari. Grazie alle notevoli affinità con le *SGR* e le banche, le SIM sono probabilmente gli intermediari finanziari caratterizzati dalla maggiore e costante presenza sui mercati finanziari, presenza che molto spesso assume un carattere determinante e decisivo nella stabilità degli stessi mercati. Grazie alla loro funzione di *dealer* o *market makers*, le società di gestione del risparmio e le società di intermediazione mobiliare garantiscono costantemente un elevato grado di liquidità ai mercati.

2.d Assicurazioni

Le *assicurazioni* non rappresentano la categoria di intermediari finanziari caratterizzata dalla maggiore presenza sui mercati, ma col tempo hanno indubbiamente assunto una maggiore rilevanza dovuta alle nuove attività finanziarie sviluppate. Fino a poco tempo fa, la principale attività intrapresa dalle società di assicurazioni era quella di offrire servizi assicurativi mirati alla semplice copertura dei rischi: assicurazioni sui veicoli, sugli immobili, sui beni di lusso e sulle persone. Con l'evolversi dei mercati e delle condizioni economico-finanziarie mondiali, anche le assicurazioni hanno dovuto modificare le proprie attività per meglio adattarle alle nuove esigenze di mercato. Nonostante non si possa osservare una vera riforma del sistema assicurativo, occorre riconoscere che alcune di quelle attività che prima costituivano solo una parte marginale di tutte le attività delle assicurazioni, rappresentano ora una parte consistente delle loro funzioni. Accanto alla classica assicurazione sulla vita, si sono affiancati decine di altri prodotti che poco alla volta hanno perso il loro carattere tipicamente assicurativo per assomigliare sempre più ai classici prodotti finanziari trattati da banche, *SGR* e *SIM*. Questi prodotti assumono spesso le caratteristiche di piani di accumulo e di risparmio.

Un elemento che ha determinato in maniera consistente lo sviluppo di questi prodotti previdenziali privati, è la riforma del sistema previdenziale attuata dal governo Amato nel 1992 e rivista dal governo Dini nel 1996. Con l'avvento di questa riforma, il sistema previdenziale italiano è stato modificato sostituendo il vecchio sistema retributivo con il

nuovo sistema su base contributiva. La modifica del sistema ha condotto alla soluzione di diversi problemi tra i quali quello riguardante le spese a carico dello Stato per il pagamento delle pensioni pubbliche. Il nuovo sistema non sarà in grado di stabilizzare i flussi finanziari fino al 2040, ma grazie alle nuove pensioni pagate con il principio dell'equità attuariale, lo Stato potrà risparmiare notevoli quantità di denaro avendo la possibilità di pareggiare i flussi previdenziali. Se da un lato lo Stato potrà sanare le proprie finanze, dall'altro lato i privati cittadini lavoratori riceveranno una pensione inferiore a quella attualmente pagata ai cittadini pensionati. Mentre con il sistema retributivo la pensione era determinata in base alla retribuzione media degli ultimi anni lavorativi, con il metodo contributivo la pensione ricevuta sarà proporzionale ai contributi versati durante tutta la vita lavorativa.

Questa riforma è stata certamente decisiva per lo sviluppo dei prodotti assicurativi previdenziali su base privata. Tutte le assicurazioni offrono oggi svariati prodotti per gestire con serenità la propria vecchiaia: chi preferisce un capitale a scadenza e chi preferisce una rendita vitalizia mensile o annuale. I singoli contratti sono spesso fortemente personalizzabili e consentono a chiunque di scegliere il piano previdenziale assicurativo che più si addice alle proprie esigenze, sia per quanto riguarda la data e l'entità dei premi da versare, che per quanto riguardano le modalità di rendita o riscatto nel caso si decida di recedere dal contratto prima della data di scadenza. Grazie a questi motivi, la presenza delle società di assicurazione sui mercati finanziari si è notevolmente ampliata e sta recentemente assumendo dimensioni considerevoli. Nonostante queste società trattino una quota marginale di titoli scambiati, le loro esigenze di mantenere costantemente nei propri portafogli titoli efficienti, le costringe ad una presenza costante che talvolta può rivelarsi sufficientemente elevata da influenzare in modo determinante l'intero sistema.

3. Investitori istituzionali

L'ultima importante categoria di soggetti operanti che merita particolare attenzione è quella degli *investitori istituzionali*. Con il termine *investitori istituzionali* si indicano tutte quelle associazioni aventi personalità giuridica che svolgono un'attività di intermediazione di carattere tipicamente pubblico orientata ad un elevato numero di investitori tanto privati quanto pubblici. Gli investitori istituzionali esplicano normalmente la loro attività mediante uno strumento che ha assunto notevole rilevanza e diffusione: il *fondo comune*. Già in precedenza, parlando delle assicurazioni e delle banche, si è accennato all'impiego di simili forme di investimento. I fondi comuni

possono essere di due categorie: ***fondi aperti*** e ***fondi chiusi***. Entrambi sono dotati di caratteristiche comuni.

I fondi comuni sono costituiti da capitali versati da società e privati suddivisi in quote ripartite proporzionalmente tra gli stessi aderenti in base a ciò che ciascun sottoscrittore ha effettivamente versato. La principale caratteristica che differenzia i fondi comuni aperti da quelli chiusi è la possibilità di rilevare la propria quota in qualunque momento successivo alla sottoscrizione. Mentre nei *fondi aperti* i partecipanti hanno diritto di rilevare la propria quota in qualunque momento chiedendo la restituzione del capitale versato valorizzato al momento della richiesta, i *fondi chiusi* non consentono di norma alcun rimborso anticipato. Alcune forme contrattuali derogano a questa disposizione e prevedono la restituzione della propria quota non prima di due o tre anni dalla data di scadenza dietro pagamento di una modesta penale.

Nonostante i ***fondi aperti*** garantiscano una maggiore liquidità all'investitore, non consentono al gestore del fondo di impiegare i fondi ricevuti in forme di investimento a lungo termine le quali offrirebbero maggiori rendimenti. Oggetto tipico di queste tipologie di investimento sono le società emergenti o di prossima espansione. La necessità di garantire il rimborso in ogni istante, impedisce all'investitore l'effettuazione di investimenti particolarmente rigidi con il risultato che il rendimento di tali fondi sarà prevedibilmente inferiore a quello che si potrebbe ottenere in altre circostanze.

Un ***fondo chiuso***, usufruendo di un capitale più stabile e costante, permette una migliore gestione di portafoglio garantendo normalmente maggiori rendimenti. Il dilemma liquidità e rendimento rappresenta la chiave di tutte le scelte operative riguardanti il portafoglio.

Si possono individuare due categorie di fondi particolarmente diffuse: quella dei ***fondi comuni di investimento*** e quella dei ***fondi pensione***. Mentre la prima categoria è rivolta a tutti gli investitori privati e a quelle società che desiderano impiegare efficientemente le proprie risorse in eccesso, la seconda categoria è rivolta esclusivamente alle persone in età lavorativa che stiano ancora esercitando la propria professione e lo scopo di questi fondi è tipicamente di carattere previdenziale. Come già in precedenza si è avuto modo di precisare, lo Stato ha attualmente in atto un'importante riforma previdenziale con lo scopo di riequilibrare l'intero sistema pensionistico. Per raggiungere tale scopo, è interesse dello Stato fare pressione

affinché i privati cittadini devolvano parte dei loro contributi e del loro *TFR* attualmente detenuto presso le imprese, a favore dei fondi pensione al fine di massimizzare i flussi finanziari futuri grazie ad una capitalizzazione mista che sfrutti al meglio i rendimenti dei mercati finanziari congiuntamente a quelli dei mercati reali. I fondi pensione gestiscono tali capitali in modo ottimale restituendo al lavoratore che ha raggiunto l'età pensionabile un flusso di denaro proporzionale a quanto versato durante la fase lavorativa maggiorato delle rendite derivanti dagli investimenti effettuati. I fondi pensione sono considerati vere e proprie pensioni integrative e rappresentano ormai uno dei principali pilastri del sistema previdenziale.

Nonostante i notevoli vantaggi che un fondo pensione può offrire ad un lavoratore, non si può sostenere che questi abbiano finora avuto un notevole successo. Alcune indagini condotte dal *CeRP*, rilevano che il vero problema alla base della scarsa diffusione di questi fondi sia localizzato nella natura del *TFR*. Pochi sono i lavoratori che si dimostrano favorevoli ad una devoluzione parziale del proprio *TFR* ai fondi pensione ad ancora meno sono quelli che accettano una devoluzione completa per scopi interamente pensionistici. Il ***TFR***, acronimo utilizzato per indicare il *Trattamento di Fine Rapporto*, rappresenta un credito che ciascun dipendente privato ha nei confronti della società per cui lavora. Dal lato impresa, il *TFR* rappresenta una fonte di finanziamento a basso costo e costituisce gran parte del passivo per un gran numero di società. Il motivo per cui i lavoratori si rivelano così riluttanti nel rinunciare al proprio *TFR* è probabilmente collegato con le tre funzioni principali a cui il *TFR* assolve e a cui i lavoratori prestano particolare attenzione. Innanzitutto esso rappresenta un ammortizzatore nel caso di perdita del lavoro; grazie a questo capitale, colui che perde il lavoro, dispone ancora di una certa autonomia economica durante il periodo che lo separa da un nuovo lavoro. In secondo luogo rappresenta un capitale che dopo otto anni di accumulo può essere utilizzato per importanti spese personali quali spese mediche o l'acquisto dell'abitazione principale. In ultimo costituisce un capitale la cui destinazione può essere scelta a piena discrezione del lavoratore stesso al termine della propria vita lavorativa.

E' difficile affermare con certezza quale possa essere il grado di influenza che questi soggetti hai sui mercati. Date le notevoli dimensioni dei portafogli titoli che questi investitori devono gestire, sarebbe lecito pensare che la loro presenza sui mercati sia decisamente elevata, ma considerando lo scarso sviluppo dimostrato dai fondi pensione, risulta altrettanto lecito pensare all'esistenza di una pressoché equa controtendenza in grado di controbilanciare la loro incidenza. Il loro grado di presenza

sui mercati dipenderà indubbiamente dallo sviluppo che i vari fondi mostreranno negli anni a venire.

2.2.2.2 Razionalità dei soggetti ed efficienza dei mercati

Si procede ora con la trattazione più dettagliata del secondo elemento determinante nella valutazione degli eventi borsistici: la *razionalità dei soggetti*. Le attuali teorie economiche presuppongono che gli esseri umani siano razionali. Non bisogna innanzitutto confondere ciò che gli economisti intendono per *razionalità* con il significato più grezzo di *ricerca del profitto*. Intendere la razionalità dei soggetti come semplice ricerca del profitto rappresenta un comune errore concettuale. I soggetti agiscono in base a ciò che loro stessi ritengono razionale. Un soggetto che accetta di effettuare una donazione per un ente benefico ad esempio può essere motivato da amor proprio o semplice altruismo che nulla ha a che fare con l'irrazionalità. Quando gli economisti parlano di razionalità, intendono l'insieme di scelte che i soggetti attuano guidati da propri interessi che non necessariamente assumono natura finanziaria.

La ***razionalità***, intesa in senso lato, è composta da due elementi separabili: gli *interessi individuali* e la *coerenza*. Valutati congiuntamente questi due elementi, i soggetti operano per ottenere i massimi vantaggi possibili. Molti studiosi sono attualmente impegnati in importanti studi sul comportamento umano ed in particolare su come i soggetti effettuino normalmente le proprie scelte. Questo studio richiede un'attenta analisi di molteplici aspetti che, anche in minima parte, influenzano le scelte dei soggetti ed aprono le porte a quelle che sono attualmente definite psicologia sperimentale e scienze cognitive.

Un aspetto particolarmente rilevante riguarda l'importanza che i soggetti attribuiscono ai tassi di rendimento nominali piuttosto che a quelli reali. Perché la gente spesso pensa in termini nominali anziché in termini reali? Forse perché risulta più semplice ed intuitivo oppure perché esiste un valido motivo alla base del ragionamento che conduce a ritenere questi più indicativi di quelli reali? Molti soggetti preferiscono avere maggior denaro pur sapendo di dover sostenere anche maggiori spese piuttosto che dover sostenere minori spese a scapito anche di minor denaro. E' forse questo un comportamento razionale? E' difficile fornire una spiegazione a questo comportamento sociale. Se i soggetti fossero completamente razionali, dovrebbero considerare

equivalenti le due situazioni ma, da quanto emerge, si potrebbe sostenere che *razionalità* significa soprattutto *ricchezza monetaria*.

Nella valutazione della razionalità dei soggetti non può essere trascurato l'aspetto informativo. I soggetti hanno la convinzione che il proprio comportamento sia tanto razionale quanto coerente ma nelle loro valutazioni non tengono debito conto di un aspetto determinante: la quantità e la qualità delle informazioni a loro disposizione. Molto spesso i soggetti tendono a prestare troppa attenzione alle informazioni recenti e poca attenzione a quelle più remote od alle aspettative future di lungo periodo. Questo elemento conduce i soggetti ad un comportamento collettivamente distorto che impone implicitamente loro una crescita aggregata trainata dai relativi trend economici. Questo potrebbe spiegare il motivo per cui i mercati tendono abitualmente ad essere dominati da modelli ricorsivi autocorrelati caratterizzati da bolle e crash. Il problema delle informazioni disponibili sul mercato merita particolare attenzione e necessita di essere integrato con un altro fondamentale concetto finanziario: l'***efficienza dei mercati***.

L'efficienza non può essere definita in modo univoco. Occorre effettuare una distinzione tra efficienza *allocativa*, *operativa* ed *informativa*. Nonostante l'efficienza di maggior rilevanza in questa sede sia quella informativa, è necessario fornire una breve definizione anche per gli altri due tipi di efficienza.

- Con il termine ***efficienza allocativa*** si indica la capacità di un mercato di distribuire adeguatamente le risorse finanziarie tra gli operatori interessati. Affinché l'ipotesi di efficienza allocativa possa concretizzarsi è necessario che l'intero processo allocativo ruoti attorno al meccanismo dei prezzi e che sia l'offerta che la domanda assumano caratteristiche coerenti sul piano razionale. Ogni investitore dovrebbe agire in modo razionale allo scopo di massimizzare i propri interessi e dovrebbe dotarsi di una struttura finanziaria ottimale in grado di ridurre al minimo i costi di competenza. Questo tipo di efficienza presuppone che tutte le informazioni siano liberamente ed economicamente disponibili a tutti i potenziali operatori (investitori e richiedenti).

- Con il termine ***efficienza operativa*** si indica la presenza di efficienza tecnica e funzionale nei mercati. Si ha *efficienza tecnica* quando gli operatori finanziari e razionalizzano la propria struttura dei costi in modo da limitare il peso degli oneri finanziari di transazione. Si ha *efficienza funzionale* quando un mercato è dotato di tutte le condizioni adatte ad agevolare l'incontro tra domanda ed

offerta e ad accrescere la significatività del sistema dei prezzi informando gli operatori interessati in modo rapido ed efficiente. L'analisi congiunta dell'*efficienza allocativa* ed *operativa* consente di analizzare l'ultimo tipo di efficienza che maggiormente interessa in questa sede: l'*efficienza informativa*.

- Con il termine ***efficienza informativa*** si indica la capacità di un mercato di distribuire tutte le informazioni provenienti da ogni fonte diffuse tramite i principali strumenti informativi. Un mercato può essere definito efficiente sotto il profilo informativo se i prezzi riflettono costantemente tutta l'informativa disponibile e gli operatori agiscono razionalmente in modo da massimizzare la propria funzione di utilità. Analizzato da un punto di vista dinamico, il livello di efficienza informativa di un mercato è inversamente proporzionale ai tempi di diffusione necessari al mercato per integrare nei prezzi le nuove informazioni rese disponibili. In base alla relazione tra informazioni e prezzi, si possono identificare tre diversi stadi di efficienza informativa:
  - *Efficienza debole*: si manifesta quando le attese di rendimento degli investitori, e quindi i prezzi, riflettono solo tutte le informazioni di tipo storico. In termini economici, la loro valenza è già stata scontata dal mercato tramite aggiustamenti progressivi già e soltanto chi disponesse di informazioni a carattere prospettico sarebbe in grado di trarre profitto da eventuali operazioni di arbitraggio.
  - *Efficienza semi-forte*: presuppone che le aspettative degli investitori in merito ai rendimenti ed ai prezzi si fondino su tutta l'informativa di dominio pubblico compresa quella in tempo reale. In questo caso soltanto gli *insider trader* potrebbero avvantaggiarsi di un canale di accesso ad informativa privilegiata. Sfruttando queste opportunità, gli *insider trader* si troverebbero nella condizione ideale per ottenere dei risultati economici superiori alla media.
  - *Efficienza forte*: presuppone che i prezzi incorporino tutta l'informativa disponibile, sia pubblica che privata, sia storica che prospettica. Queste condizioni impediscono la possibilità di anticipare il mercato e quindi di negoziare valori mobiliari a condizioni vantaggiose sul piano economico.

Se un mercato non è efficiente le informazioni non si distribuiscono in modo omogeneo e nascono quelle che comunemente prendono il nome di ***asimmetrie informative***. Con questo termine si indicano tutte quelle situazioni in cui le informazioni che dovrebbero essere disponibili all'intera collettività interessata, sono in realtà distribuite

asimmetricamente. Quando si parla di distribuzione asimmetrica occorre riferirsi tanto all'aspetto quantitativo quanto a quello qualitativo. Alcuni soggetti potrebbero disporre di maggiori informazioni ma di minor affidabilità mentre altri potrebbero disporre di minori informazioni ma di gran lunga più affidabili. La principale causa di queste asimmetrie va generalmente ricercata nella difficoltà e nel costo dell'acquisizione e valutazione delle informazioni disponibili nella collettività. Questo problema risulta particolarmente sentito soprattutto per quelle categorie di soggetti di minor rilevanza. In queste categorie rientrano indubbiamente i privati cittadini e le piccole istituzioni private. Questi soggetti, dati gli elevati costi di acquisizione che dovrebbero sopportare per ottenere le informazioni che desiderano, finiscono spesso con l'utilizzare strumenti e tecniche decisamente più semplici ma altrettanto meno efficienti. Il comportamento maggiormente utilizzata dalla collettività prende il nome di *imitazione* o, più precisamente, ***comportamenti imitativi***.

I soggetti, per prendere le proprie decisioni, osservano il comportamento degli altri ed agiscono di conseguenza. La tentazione da parte dei nuovi arrivati di seguire le orme di chi ha già operato diventa sempre più forte ed innesca un ciclo vizioso che conduce il mercato ad una situazione di instabilità caratterizzato da bolle e crash. Mano a mano che le bolle crescono, molti soggetti sono incentivati a seguire il trend e, anche una piccola ed insignificante informazione, può causare una reazione a catena che è in grado di alterare le scelte dei soggetti e quindi le condizioni dello stesso mercato. Gli operatori economici si comportano spesso come un gregge di pecore le quali si muovono per inerzia ciascuna seguendo colei che le sta di fronte. Gran parte delle scelte che messe in atto dai soggetti riflettono più che la propria volontà, quella indotta dagli altri membri della collettività. E' questo il caso di un comportamento irrazionale o si tratta semplicemente di una legge sociale imposta dalla stessa natura umana?

La sfida posta in essere riguarda proprio lo studio su quali siano le cause che maggiormente influenzano il comportamenti dei soggetti nel seguire le proprie preferenze individuali. Molte sono le teorie alla base di questi studi, ma pochi sono i risultati ottenuti finora. E' accertato che un soggetto razionale valuta molto di più un euro oggi piuttosto che un euro domani e questo è legato al rischio sull'incertezza del domani, ma gli stessi economisti ammettono che questa teoria non è comunque sufficiente per comprendere più dettagliatamente come i soggetti scontino il futuro e soprattutto quali siano tutti i motivi che li spingono ad operare ed agire in una certa direzione. Sulla base di questi fenomeni, molti sono i modelli informatici sviluppati e simulati al calcolatore allo scopo di ottenere delle plausibili spiegazioni.

2.2.2 Teoria di Portafoglio

Data l'elevata variabilità dei mercati, è lecito pensare all'investimento come ad un processo volto alla massimizzazione del rendimento e contemporaneamente alla riduzione del rischio. Il processo di investimento consiste normalmente in due operazioni sequenziali. La prima riguarda l'analisi generale dei mercati. La seconda consiste nell'individuazione di un portafoglio ottimo costituito da titoli disponibili sul mercato; un portafoglio in grado di minimizzare il rischio associato ad ogni singolo rendimento. Lo studio di queste due operazioni finanziarie prende il nome di ***teoria di portafoglio***.

Una prima fase della trattazione esaminerà alcuni concetti teorici quali il significato di rischio, avversione al rischio, funzione di utilità dei soggetti nonché il concetto di allocazione dei capitali ed il rapporto tra rischio e rendimento. Una seconda fase della trattazione analizzerà i più comuni strumenti utilizzati per la minimizzazione del rischio e la massimizzazione del rendimento. Parte di questi possono risultare un po' teorici, ma la loro analisi si rivela necessaria ai fini di una completa trattazione dell'argomento. In particolare saranno analizzati il metodo della *Frontiera efficiente* e quello del *Capital Asset Pricing Model* noto anche come *CAPM*.

2.2.2.1 Concetti generali

Occorre innanzitutto chiarire la distinzione tra due termini che, nonostante abbiano significati diversi, sono spesso oggetto di confusione: *speculazione* ed *azzardo*. Economicamente parlando, un ***azzardo*** è l'assunzione di un rischio senza finalità alcuna se non quella di aver assunto lo stesso rischio per spirito di divertimento. La ***speculazione*** si riferisce all'assunzione di un rischio dietro un adeguato premio in contropartita. L'elemento differenziale tra *speculazione* ed *azzardo* prende il nome di *premio al rischio*. Per mutare un semplice azzardo in una speculazione è sufficiente introdurre un adeguato premio al rischio in grado di compensare lo svantaggio derivante dall'assunzione del rischio stesso.

Il ***premio al rischio*** è il maggior rendimento che qualunque soggetto razionale domanda, in un mercato perfettamente efficiente, per detenere nel proprio portafoglio un titolo avente un particolare grado di rischio. Il grado di rischio di un titolo è dato dalla sua variabilità nel tempo e, espresso in termini matematici, è misurato dalla sua

varianza o, a scelta, dal suo scarto quadratico medio. Maggiore è il livello di rischio associato ad un titolo, e maggiore sarà il suo *premio al rischio*. Il *premio al rischio* deve essere valutato utilizzando come base il *tasso privo di rischio*. Il *tasso privo di rischio* è quel tasso che per definizione è assunto con varianza pari a zero. Nessun titolo è in realtà privo di rischio e, nonostante alcuni titoli siano caratterizzati da elevata affidabilità, non possono comunque essere considerati privi di rischio. Ai fini pratici si è comunque deciso di accettare come tali i titoli di Stato a breve termine (*BOT*), i certificati di deposito bancari (*CD*) ed i crediti commerciali (*CP*). In formule:

$$\Pr_i = R_i - R_f$$

Dove:
Pr_i è il premio al rischio del titolo i-esimo
R_i è il rendimento del titolo i-esimo
R_f è il rendimento dei titoli privi di rischio

Il premio al rischio è calcolato come la differenza tra il rendimento del titolo ed il rendimento dei titoli di stato a breve considerati di norma privi di rischio.

Un elemento di fondamentale importanza in ambito finanziario è rappresentato dalla tipologia di ***investitori***. Esistono principalmente tre tipologie di investitori:

- Gli *investitori avversi al rischio* sono coloro che seguono le regole razionali ed investono solo se il relativo premio al rischio offerto dall'investimento è sufficiente a coprire il rischio incorporato nello stesso.
- Gli *investitori neutrali al rischio* basano il proprio profilo di rischio solamente sulle proprie attese di rendimento. Per questa tipologia di investitori è totalmente irrilevante il grado di rischio di un investimento ed il rendimento certo equivalente del proprio portafoglio è dato semplicemente dal tasso di rendimento.
- Gli *investitori amanti del rischio* adattano la propria funzione di utilità alle diverse condizioni di rischio ottenendo sempre un certo equivalente che eccede le rispettive alternative in investimenti privi di rischio.

Il profilo di ciascun investitore può essere individuato teoricamente utilizzando quella che viene chiamata *funzione di utilità*. La ***funzione di utilità*** è una funzione matematica spesso utilizzata in ambito finanziario che, nonostante rappresenti

un'astrazione in campo pratico, è molto utile per comprendere la teoria di portafoglio. La funzione di utilità maggiormente utilizzata in queste circostanze è la seguente:

$$U = E(r) - 0.005 \cdot A \cdot \sigma^2$$

Il valore assunto dall'utilità dipende positivamente dal rendimento atteso e negativamente dall'avversione al rischio da parte del soggetto nonché dal rischio stesso dell'investimento. Maggiore è l'avversione al rischio da parte di un soggetto, e minore sarà l'utilità che egli otterrà da un investimento caratterizzato da un certo rendimento atteso ed un certo rischio. Il coefficiente 0.005 rappresenta un parametro correttivo. E' stato scelto come tale in quanto ritenuto maggiormente realistico rispetto ad altri possibili coefficienti ma avrebbe potuto valere qualsiasi valore appartenente allo stesso ordine di grandezza.

Dalla funzione di utilità si possono ricavare le *curve di indifferenza*. Le ***curve di indifferenza*** rappresentano l'insieme di tutte le combinazioni rischio-rendimento aventi la medesima utilità. Tutti i punti giacenti sulla stessa curva sono considerati equivalenti. Sull'asse delle ascisse è indicato il grado di rischio rappresentato dallo scarto quadratico medio mentre sull'asse delle ordinate è indicato il rendimento atteso. La curva di indifferenza rappresenta l'insieme di tutte le combinazioni rischio-rendimento per cui ciascun investitore è indifferente nella scelta del proprio portafoglio titoli.

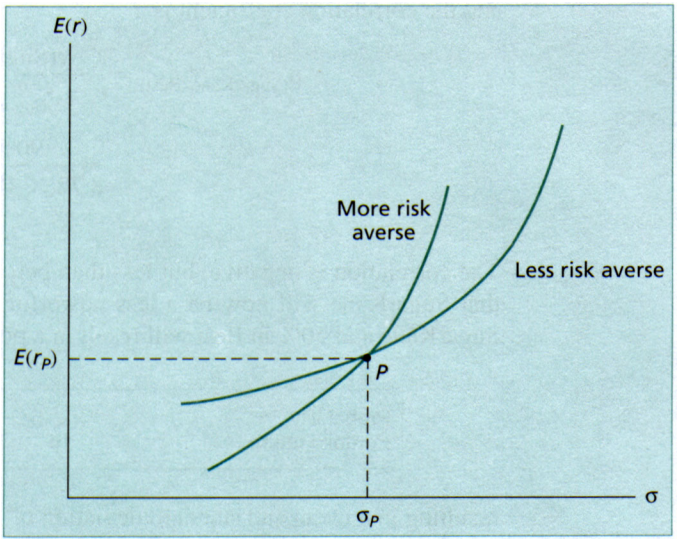


Figura 7 - Curve di indifferenza di due soggetti

Nella figura sono mostrate due curve aventi caratteristiche diverse. Una raffigura un soggetto maggiormente avverso al rischio mentre l'altra si riferisce ad un soggetto meno avverso. La curva di indifferenza del soggetto più avverso al rischio risulta

notevolmente più inclinata in quanto, a parità di rischio, è richiesto un rendimento decisamente maggiore. L'investitore meno avverso al rischio invece, a parità di rischio, è disposto ad accettare un rendimento inferiore. Questa logica vale per tutti i rendimenti superiori a quello fornito dal portafoglio di mercato rappresentato dal punto *P*. Senza indagare ulteriormente su cosa rappresenti il portafoglio di mercato che verrà analizzato meglio in seguito, si può affermare che una curva di indifferenza sarà tanto più inclinata quanto maggiore sarà l'avversione al rischio da parte del soggetto. Il fatto che alcuni investitori siano più avversi al rischio di altri non significa comunque che questi non accettino di investire in titoli più rischiosi: lo faranno purché ricevano in cambio un maggior rendimento. Gli investitori costruiscono generalmente i propri portafogli utilizzando titoli di diversa natura. Alcuni titoli potranno essere privi di rischio come i titoli di stato, altri invece particolarmente rischiosi come azioni emesse da società poco affidabili. Quando si desidera effettuare un investimento di carattere professionale, uno degli aspetti più importanti è certamente la costruzione di un buon portafoglio titoli mediante un'efficiente allocazione dei capitali.

L'allocazione dei capitali rappresenta sempre un'operazione complessa. Il principale problema risiede nella quantità di capitali da allocare in titoli privi di rischio e la quantità di capitali da allocare in titoli rischiosi. Tutte le combinazioni rischio-rendimento possono sempre essere disposte su un piano cartesiano. La retta di allocazione dei capitali si ottiene dall'insieme di punti derivanti dalla combinazione di due o più titoli. Dato un generico titolo privo di rischio e dato un generico titolo rischioso, il rendimento atteso del portafoglio titoli derivante dalla combinazione dei due assume la seguente forma algebrica:

$$E(r_p) = r_f \cdot \omega_f + E(r_r) \cdot \omega_r$$

Dove:
$E(r_p)$ è il rendimento atteso dell'intero portafoglio
$E(r_r)$ è il rendimento atteso del titolo con rischio
r_f è il rendimento del titolo privo di rischio
ω_f è la quota in portafoglio del titolo privo di rischio
ω_r è la quota in portafoglio del titolo con rischio.

Il rendimento atteso di portafoglio è dato dalla media ponderata del rendimento dei due titoli utilizzando come pesi le rispettive quote di portafoglio. La varianza del portafoglio è rappresentata nel seguente modo:

$$\sigma_p{}^2 = \omega_f{}^2 \cdot \sigma_f{}^2 + \omega_r{}^2 \cdot \sigma_r{}^2 + 2 \cdot \omega_f \cdot \omega_r \cdot Cov(r_f, r_r)$$

Dove:

r_f è il rendimento del titolo privo di rischio

$E(r_r)$ è il rendimento atteso del titolo con rischio

ω_f è la quota in portafoglio del titolo privo di rischio

ω_r è la quota in portafoglio del titolo con rischio

σ_p è la varianza dell'intero portafoglio

σ_f è la varianza del titolo privo di rischio

σ_r è la varianza del titolo con rischio.

La varianza dell'intero portafoglio è data dalla media pesata delle singole varianze più il doppio del prodotto dei pesi e la covarianza tra i due titoli. Dal momento che la varianza del titolo privo di rischio è pari a zero per definizione e la covarianza tra i due titoli è altrettanto pari a zero, la formula della varianza assume la seguente forma:

$$\sigma_p^2 = \omega_r^2 \cdot \sigma_r^2$$

La varianza di un portafoglio con due soli titoli di cui uno privo di rischio dipende esclusivamente dalla varianza del titolo con rischio ed il suo valore è direttamente proporzionale alla sua quota di portafoglio. L'insieme dei punti che si ottengono dalla combinazione risultante dal titolo privo di rischio e dal titolo con rischio forma una retta con intercetta pari al rendimento del titolo privo di rischio e coefficiente angolare pari a:

$$S = \frac{E(r_r) - r_f}{\sigma_p}$$

S non è altro che la tangente calcolata tramite il rapporto dei cateti. La retta risultante prende il nome di ***Capital Allocation Line*** (*CAL*) e rappresenta l'insieme di tutti i portafogli ottenibili dalla combinazione dei due titoli presi in considerazione.

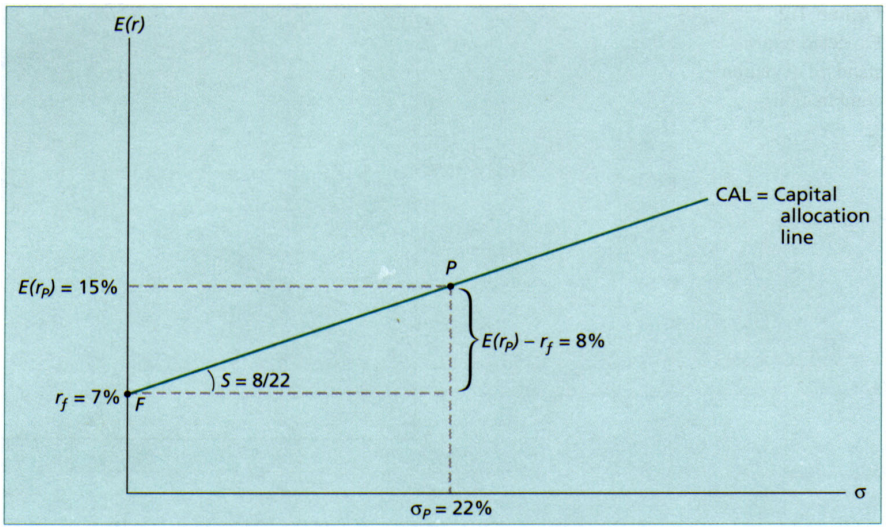


Figura 8 - Retta di allocazione dei capitali (CAL)

Se al posto di un generico titolo rischioso si fosse scelto un titolo rappresentativo dell'intero portafoglio di mercato, la retta di allocazione dei capitali (*CAL*) avrebbe assunto il nome di ***Capital Market Line*** (*CML*). Il portafoglio di mercato è un portafoglio al cui interno sono contenuti, in proporzione alle relative quote di mercato, tutti i titoli del mercato stesso. La principale caratteristica del portafoglio di mercato è quella di non possedere alcun rischio specifico, ma di questo si tratterà meglio in seguito.

Valutando congiuntamente la *Capital Allocation Line* e la curva di indifferenza di un soggetto, è possibile individuare la combinazione di rischio e rendimento che massimizza l'utilità attesa dall'investitore. Il punto di ottimo sarà dato dal punto di tangenza tra la curva di indifferenza e la retta dei capitali come appare in figura.

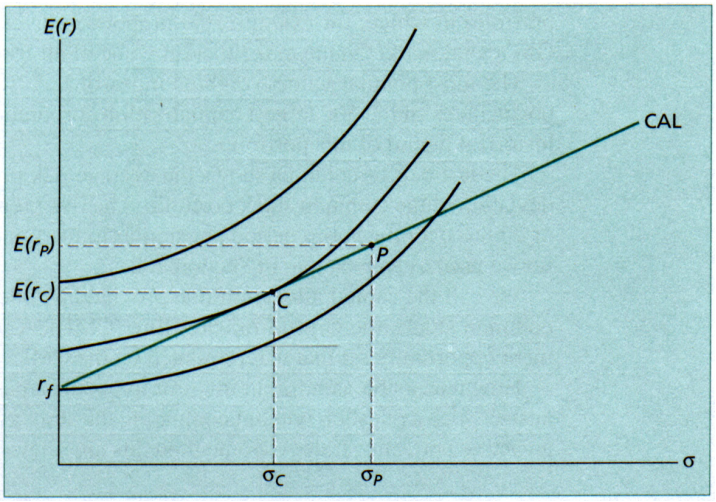


Figura 9 - Determinazione della combinazione ottima tra una curva di indifferenza e la CAL

Il punto *C* rappresenta la miglior combinazione rischio-rendimento ottenibile da un investitore avente una curva di indifferenza di tipo due.

2.2.2.2 Tecniche di minimizzazione del rischio

Nel paragrafo precedente si è analizzata la funzione di utilità e l'allocazione dei capitali sostenendo che la valutazione congiunta di questi due elementi conduce alla soluzione ottima per l'investitore. Non si è ancora illustrato come scegliere i titoli per formare un portafoglio e soprattutto se il portafoglio così ottenuto sia efficiente. Questo paragrafo focalizzerà la sua attenzione sulla scelta di un portafoglio efficiente ottenuto mediante diversificazione e sulla costruzione di un modello dotato di particolari caratteristiche finanziare chiamato *CAPM*.

1. Diversificazione e frontiera efficiente

Si è già esaminata la condizione in cui esistono due titoli in portafoglio, uno privo di rischio ed uno con un certo grado di rischio. Se in un portafoglio fossero inseriti più di due titoli, cosa muterebbe nelle caratteristiche del portafoglio? Il rendimento ottenibile a parità di varianza sarebbe maggiore o minore rispetto al caso precedente? Si presti attenzione alle seguenti formule:

$$E(r_p) = \sum_i \omega_i \cdot E(r_i)$$

$$\sigma_p^2 = \sum_i \sum_j \omega_i \cdot \omega_j \cdot \sigma_i \cdot \sigma_j \cdot \rho_{ij}$$

Le formule sono analoghe a quelle già analizzate nel caso di due soli titoli con la sola differenza di essere espresse in termini generali. Il rendimento atteso di un generico portafoglio composto da *n* titoli è pari alla media ponderata del rendimento atteso degli stessi titoli in portafoglio. La varianza sarà data dalla media pesata delle proprie covarianze moltiplicata per il rispettivo coefficiente di correlazione. Per quanto riguarda il rendimento atteso, nulla appare particolarmente ostico. Qualcosa in più occorre dire sulla la varianza del portafoglio. Si è detto che questa dipende dalle covarianze tra i vari titoli e soprattutto dal loro coefficiente di correlazione. In genere le covarianze tra più titoli non sono mai dello stesso segno e il più delle volte, se i titoli in portafoglio sono molti, si distribuiscono con distribuzione normale con media zero. Le covarianze generano una specie di compensazione algebrica che rende la varianza del portafoglio

tanto più piccola quanto maggiore è il numero di titoli in portafoglio. Questa caratteristica prende il nome di beneficio derivante dalla ***diversificazione di portafoglio*** e permette di isolare quello che è definito *rischio specifico* da quello che è chiamato *rischio sistematico* o di *mercato*.

- Il ***rischio specifico*** è il rischio incorporato in ogni singolo titolo. Una società ad alto rischio di fallimento o insolvenza avrà un rischio specifico maggiore rispetto ad un'altra società che avrà ottimi profitti. Il rischio specifico è implicitamente legato alla salute della società emittente e l'unico modo che un investitore ha a disposizione per ridurre il rischio specifico è quello di diversificare il portafoglio. Abbinando più titoli che abbiano correlazione fortemente negativa è possibile ottenere un portafoglio che abbia una varianza decisamente più bassa.

- Il ***rischio sistematico*** o di mercato rappresenta il rischio associato all'intero mercato e non può in alcun modo essere eliminato. Quando emergono alcune notizie che sconvolgono l'intero mercato, queste contribuiscono in modo negativo sul rischio sistematico. Il rischio di mercato è anche considerato come il rischio residuo che permane dopo la differenziazione e corrisponde di norma alla varianza del portafoglio di mercato. Se si tracciasse un grafico in cui si pongono in relazione la varianza di un portafoglio ed il numero di titoli presenti, si noterebbe che la varianza di portafoglio tende a diminuire progressivamente all'aumentare del numero di titoli presenti fino a raggiungere asintoticamente la varianza del portafoglio di mercato pari al rischio sistematico.

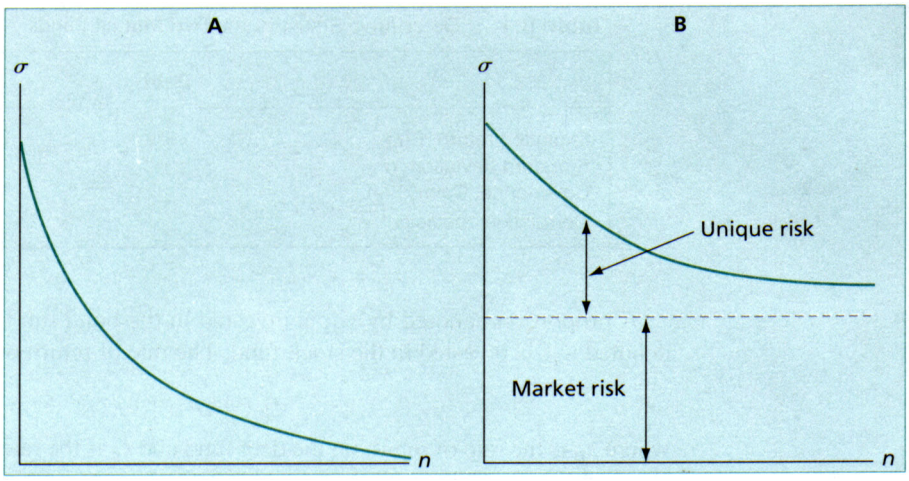


Figura 10 - Scomposizione del rischio: rischio specifico e rischio di mercato

La differenziazione di portafoglio consente di minimizzare il rischio specifico e permette di ottenere la cosiddetta *frontiera efficiente*. Dato un insieme generico di titoli rischiosi,

è definita ***frontiera efficiente*** l'insieme di tutte le combinazioni rischio-rendimento che massimizzano il rendimento per ogni livello di rischio. Il calcolo della frontiera efficiente è piuttosto semplice se il portafoglio contiene due soli titoli ma alquanto complesso se il portafoglio eccede i due titoli. In questo caso è necessario l'utilizzo di un calcolatore elettronico. Il modello basato sul calcolo della frontiera efficiente è chiamato ***modello di portafoglio secondo Markowitz*** e propone di individuare le quote di portafoglio da attribuire a singoli titoli per ottenere l'insieme di tutti i punti maggiormente efficienti.

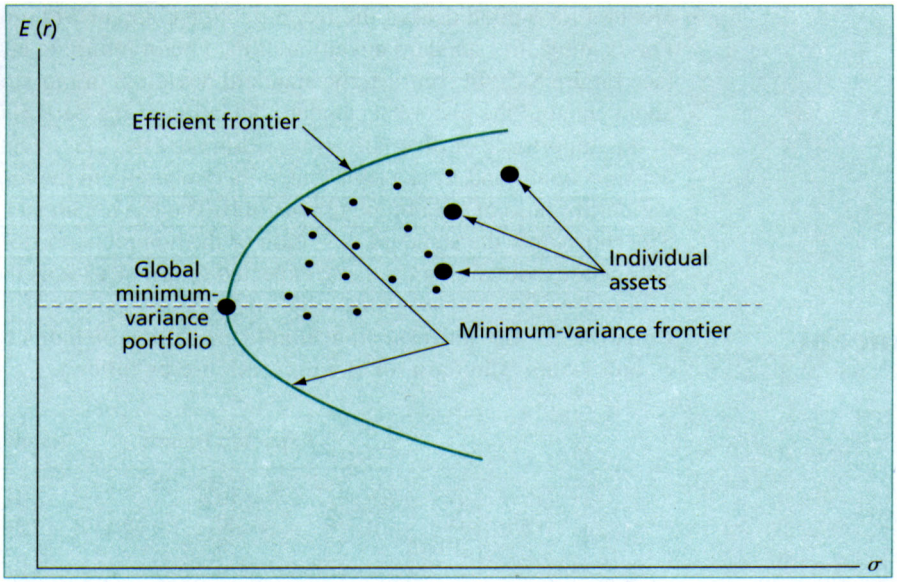


Figura 11 - Frontiera efficiente

Tutti i portafogli che si collocano sulla frontiera efficiente sono considerati a varianza minima e forniscono le migliori combinazioni rischio-rendimento candidate ad essere scelte per la costruzione del portafoglio ottimo. Quale sarà il portafoglio ottimo dipenderà esclusivamente dalla curva di indifferenza di ogni investitore e la tangenza tra le due curve rappresenterà il portafoglio ottimo.

2. Capital Assets Pricing Model

Un altro dei principali strumenti maggiormente utilizzati per controllare il rischio nelle proprie decisioni di investimento è rappresentato dal ***Capital Assets Pricing Model*** detto comunemente *CAPM*. Il *CAPM* è costruito sul concetto che l'esatto premio al rischio di un titolo sia determinato in base al contributo che questo fornisce al rischio di tutto il portafoglio. Il *CAPM* attribuisce a ciascun titolo del portafoglio un grado di rischiosità proporzionale al rischio dell'intero portafoglio. Questo rischio è espresso da un coefficiente chiamato *β*. Il coefficiente *β* indica il contributo al rischio di ogni singolo

titolo ed è dato dal rapporto tra la covarianza dello titolo rispetto al portafoglio di mercato e la varianza del portafoglio stesso. In formule:

$$\beta_i = \frac{Cov(r_i, r_M)}{\sigma_M^2}$$

Il coefficiente *β* può anche essere semplicemente calcolato come rapporto tra il premio al rischio del titolo ed il premio al rischio del portafoglio di mercato. In formule:

$$\beta_i = \frac{E(r_i) - r_f}{E(r_m) - r_f}$$

A numeratore si trova il premio al rischio del titolo i-esimo mentre a denominatore il premio al rischio del portafoglio di mercato. Il portafoglio di mercato ha un coefficiente *β* pari ad uno mentre tutti i titoli di un portafoglio avranno un *β* che potrà essere maggiore o minore di uno. Se un titolo ha un *β* maggiore di uno significa che è più rischioso dell'intero portafoglio di mercato e dovrà avere un rendimento atteso maggiore di quello fornito dal portafoglio stesso. Se un titolo ha un *β* minore di uno è considerato meno rischioso del portafoglio di mercato e, per essere in equilibrio, deve fornire un rendimento atteso minore di quello fornito dal portafoglio di mercato. Un titolo con *β* pari a zero è considerato privo di rischio e pagherà un rendimento uguale a quello dei titoli privi di rischio. E' anche possibile che un titolo abbia un *β* negativo. In tal caso sarà considerato meno rischioso dei *risk free bonds* e potrà garantire un rendimento atteso inferiore a quello garantito dagli stessi titoli privi di rischio.

Secondo la teoria del *CAPM*, i rendimenti dei singoli titoli si distribuiscono in base al loro grado di rischio su una retta definita *Securities Market Line (SML)*. Se il mercato è in perfetto equilibrio tutti i titoli del portafoglio devono collocarsi sulla *Securities Market Line* (*SML*). Se un titolo avente un certo *β* mostra un rendimento atteso superiore a quello di equilibrio, la differenza emergente prende il nome di eccesso ed è indicata da un altro coefficiente denominato *α*. Un coefficiente *α* diverso da zero non rappresenta una condizione di equilibrio e può condurre a possibilità di arbitraggi.

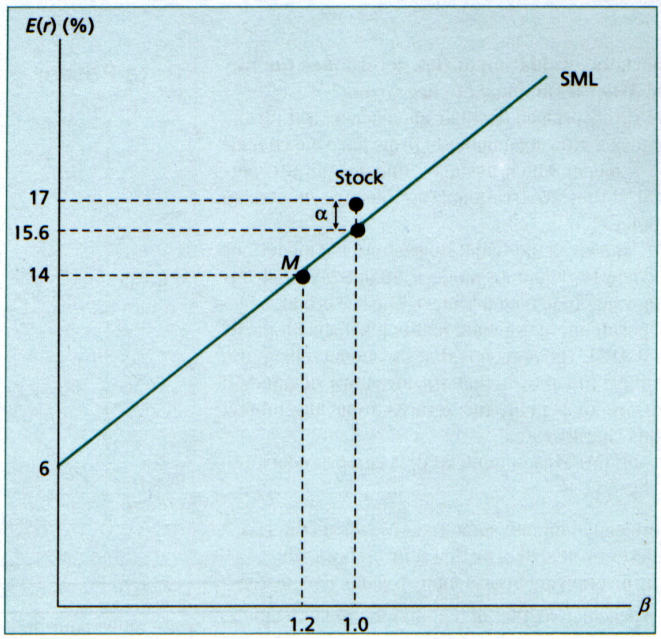


Figura 12 - Securities Market Line

Un titolo che garantisce un rendimento superiore di quanto dovrebbe attirerà molti investitori che grazie ai loro acquisti ne faranno aumentare il prezzo riducendone così il relativo rendimento. La stessa cosa si può affermare per il caso opposto. Se un titolo mostra un rendimento inferiore a quello dovuto, risulta poco appetibile e la sua continua vendita ne farà scendere il prezzo facendo così aumentare il relativo rendimento. Il processo di riallineamento in entrambi i casi proseguirà finché il titolo non si sarà nuovamente collocato sulla retta dei capitali.

Il *Capital Assets Pricing Model* basa il suo funzionamento sul presupposto della diversificazione di portafoglio. Se non si diversificasse il portafoglio, si rischierebbe di includere al suo interno titoli ancora fortemente soggetti al proprio rischio specifico ed il premio al rischio che si determinerebbe secondo la teoria del *CAPM* risulterebbe distorto rispetto al valore reale. Quando si utilizza il modello *CAPM* è pertanto necessario includere tutti i titoli del mercato e questo rappresenta un aspetto negativo del modello. Un modello finanziariamente meno perfetto ma certamente più pratico da analizzare è il ***Single Index Model***. In questo modello, anziché considerare tutti i titoli del mercato sottostante, si prendono in considerazione solo i principali titoli che normalmente formano l'indice caratteristico di ciascun mercato. In tal modo è possibile semplificare notevolmente il calcolo dei coefficienti e la diversificazione che si ottiene è sufficientemente elevata tale da poter considerare trascurabile l'effetto derivante dai singoli rischi specifici. Nonostante non si disponga delle covarianze che i singoli titoli hanno nei confronti del portafoglio di mercato, è possibile ricavare tutti i coefficienti β mediante una semplice regressione econometria sulle serie storiche dei singoli titoli. La

variabile indipendente sarà il premio al rischio dell'indice di mercato mentre la variabile dipendente il premio al rischio del titolo stesso. Il coefficiente della retta di regressione rappresenterà il β mentre l'intercetta il suo coefficiente α.

Capitolo 3
Programmazione e Swarm

Prima di esaminare gli aspetti caratteristici del modello di simulazione *SUM*, occorre addentrarsi brevemente in un argomento che rappresenta non solo la base delle simulazioni, bensì tutto il mondo informatico: la programmazione. La programmazione rappresenta il pilastro su cui poggia il funzionamento di tutti i calcolatori nonché dell'informatica stessa. Nonostante non sia questa la sede per fornire i dettagli più specifici circa la programmazione, pare adeguato fornire almeno le basi necessarie per comprendere il funzionamento del modello di simulazione nonché i risultati emergenti dallo stesso.

3.1 Programmazione ad oggetti

Un linguaggio di programmazione ad oggetti è in grado di rendere la stesura dei programmi molto intuitiva, veloce da sviluppare e soprattutto flessibile per eventuali successive modifiche. Questa tipologia di linguaggi non conduce solamente ad una nuova metodologia nella costruzione dei programmi, instaura un modo completamente nuovo ed innovativo di concepire la struttura della programmazione. I linguaggi di programmazione ad oggetti possono però rappresentare un ostacolo per coloro che desiderano avvicinarsi alla programmazione per la prima volta, ed introducono un modo di vedere le cose che a prima vista può risultare molto complesso soprattutto per un principiante.

3.1.1 Ambiente di Sviluppo

Tutti gli ambienti di sviluppo utilizzati per la programmazione ad oggetti constano di almeno tre elementi:

- Una o più biblioteche di oggetti che fungono da struttura per l'ambiente
- Alcuni strumenti di sviluppo
- Un linguaggio di programmazione ad oggetti.

Un insieme di librerie è sempre necessario per sviluppare un adeguato ambiente operativo. All'interno delle librerie sono normalmente contenute tutte le definizioni degli

oggetti più comuni che possono essere utilizzati con maggior frequenza. Gli strumenti di sviluppo sono prevalentemente utilizzati per assemblare le diverse applicazioni nonché accertarsi del corretto funzionamento delle applicazioni create o in fase di creazione. In termini informatici, questi strumenti sono noti con il nome di *compilatori* e *debugger*. Ultimo, e certamente più importante, è il linguaggio di programmazione. Il linguaggio adottato rappresenta l'elemento determinante per lo sviluppo dell'ambiente. I linguaggi di programmazione ad oggetti attualmente presenti sono molti, ma quelli considerati di maggior rilevanza sono soltanto due: *C++* e *Java*.

I linguaggi di programmazione tradizionali hanno diviso la logica informatica in due parti: i dati o informazioni da una lato, e le operazioni sugli stessi dall'altro. I linguaggi di programmazione ad oggetti utilizzano la stessa struttura. I due elementi sono mantenuti separati all'interno di semplici unità modulari chiamate *oggetti* in grado di interagire tra loro. In un linguaggio di programmazione ad oggetti, oggetti e loro interazione, rappresentano gli elementi base del funzionamento. Ogni oggetto contiene informazioni e comportamenti che in termini informatici sono rispettivamente definiti istanze e metodi.

3.1.2 Interfaccia ed Implementazione

Quando si analizzano i linguaggi di programmazione, occorre innanzitutto sottolineare la netta distinzione esistente tra l'*interfaccia* dell'applicazione e la sua *implementazione*. Un'applicazione può essere paragonata ad un orologio. La cassa e le lancette rappresentano l'interfaccia, gli ingranaggi contenuti all'interno, invisibili all'utilizzatore, rappresentano l'implementazione ossia il suo funzionamento.

- L'***interfaccia*** è l'aspetto esterno di un'applicazione. L'*interfaccia* è il mezzo tramite cui un'applicazione comunica con l'utilizzatore. Tramite l'*interfaccia* un'applicazione è in grado di mostrare i dati contenuti al suo interno e permettere il proprio controllo da parte dell'utilizzatore. Senza un'adeguata interfaccia, il compito dell'utilizzatore risulterebbe di gran lunga più articolato.

- L'***implementazione*** è rappresentata dalla *struttura dati* e dalle *funzioni*. Questi due elementi costituiscono il corpo di un oggetto e saranno analizzati meglio in seguito.

- Le ***strutture*** raggruppano diverse informazioni in grandi unità che possono essere trattate come singole entità. Una struttura può contenere al suo interno altre strutture così da creare una complessa gerarchia di informazioni che in termini informatici assume il nome di *struttura stratificata*. Ogni struttura è dotata di caratteristiche proprie ed è contenuta in uno spazio di memoria a se stante. Grazie a questa caratteristica, più strutture possono assumere la stessa denominazione senza che sorgano errori interpretativi da parte delle applicazioni.

- Le ***funzioni*** contengono i comportamenti che possono essere attuati durante l'esecuzione di un'applicazione. Le funzioni, una volta definite, possono essere richiamate tutte le volte che ne sorge la necessità. Le funzioni maggiormente utilizzate possono essere raggruppate in specifiche biblioteche in modo da poter essere riutilizzate anche in diverse applicazioni. A differenza delle strutture dati, ciascuna funzione deve essere caratterizzata da un nome univoco.

3.1.3 Il modello ad oggetti

Il principio alla base dei linguaggi di programmazione ad oggetti è quello di combinare le informazioni e le funzioni in modo da sviluppare un'unica unità di alto livello che prende il nome di *oggetto*. Un *oggetto* è un'entità che raccoglie al suo interno un gruppo di funzioni con una propria struttura dati. Le funzioni implementano i *metodi* mentre le loro strutture dati sono definite *istanze*. I *metodi* si collocano attorno alle *istanze* (in genere rappresentate da variabili) occultandole agli altri oggetti del modello. L'unico modo per manipolare le *istanze* di un oggetto è tramite l'utilizzo dei suoi *metodi*.

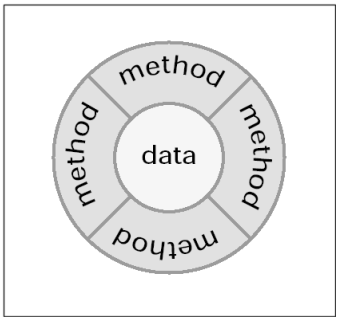


Figura 13 - Schema di un oggetto informatico

Un programma consiste normalmente in una rete di oggetti interconnessi. Ciascun oggetto ha uno specifico ruolo all'interno dell'intero programma ed è in grado di comunicare con tutti gli altri oggetti del modello. Gli oggetti comunicano tra loro mediante *messaggi* richiesti od inviati tramite i *metodi*.

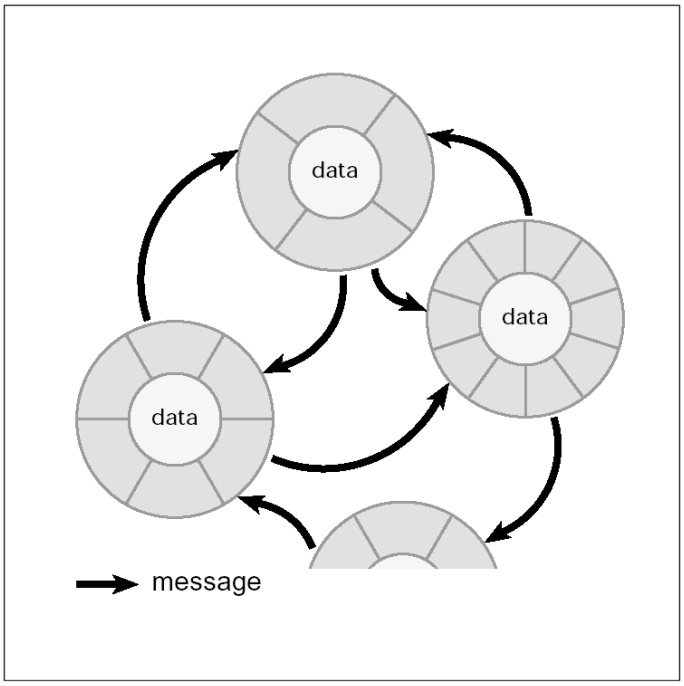


Figura 14 - Schema di una rete di oggetti

I messaggi inviati tramite i diversi metodi non necessariamente sono interpretati allo stesso modo da tutti gli oggetti. Dal momento che ogni oggetto ha caratteristiche proprie, uno stesso messaggio inviato a più oggetti può generare comportamenti diversi a seconda della tipologia di oggetto ricevente.

3.1.3.1 Classi

In un linguaggio di programmazione ad oggetti, tutti gli oggetti dello stesso tipo appartengono alla stessa *classe*. I membri di una stessa *classe* sono in grado di utilizzare gli stessi metodi e disporre delle stesse istanze. Una classe rappresenta un modello tramite il quale si possono forgiare una quantità indefinita di oggetti aventi tutti le stesse caratteristiche. In termini metaforici si può pensare ad una classe come ad una generica forma generatrice: da questa si possono generare tanti oggetti tutti identici quanti se ne desiderano. La definizione di una *classe* è simile alla definizione di una struttura con l'eccezione di comprendere anche i metodi che definiscono il comportamento degli oggetti. Due oggetti aventi la stessa struttura dati ma contenenti

metodi diversi, non appartengono alla stessa classe. Le classi sono caratterizzate da un'estrema unicità.

3.1.3.2 Modularità

Quando si trattano applicazioni di notevoli dimensioni, è bene considerare l'idea di dividere tali applicazioni in parti più piccole in grado di essere trattate e compilate singolarmente. Queste parti sono definite *moduli*. Espresso in termini informatici, un modulo non è altro che un file contenente del codice sorgente. Da un punto di vista formale, non rappresenta una vera unità logica del linguaggio bensì un semplice contenitore dove inserire il codice di un programma. I moduli sono spesso utilizzati per raggruppare parti di codice strettamente intercorrelate. La scelta della struttura dei moduli è sempre un'operazione difficile da intraprendere e l'identificazione di una struttura modulare chiara e completa in grado di semplificare il codice non sempre risulta di facile realizzazione . Un codice contenente troppi moduli può creare difficoltà interpretative soprattutto se tale codice non è più stato visionato da un po' di tempo mentre un codice contenente pochi moduli rischia di creare un problema analogo ma speculare. Una buona applicazione dovrebbe contenere un numero di moduli adeguato in base ai fini che intende perseguire. La corretta scelta dei moduli dipende, oltre che dai fini la cui applicazione intende perseguire, anche dalle capacità del programmatore.

3.1.3.3 Riutilizzabilità

Uno dei maggiori obiettivi raggiunti dai linguaggi di programmazione ad oggetti riguarda certamente la riutilizzabilità del codice. Grazie alla riutilizzabilità è possibile evitare la riscrittura dello stesso codice in più situazioni. La riutilizzabilità del codice è determinata da diversi fattori che ne influenzano la sua desiderabilità:

- *Affidabilità del codice*: L'affidabilità è intesa in questo contesto come sinonimo di stabilità. Chi desidera costruire applicazioni stabili ed efficienti, vedrà nell'impiego di un codice affidabile una scelta obbligata.
- *Chiarezza della documentazione*: Un codice con scarsa documentazione disincentiva l'utilizzo dello stesso. La documentazione espone le funzionalità del codice nonché il suo utilizzo. Una documentazione esaustiva e chiara favorisce l'apprendimento del codice anche da parte dei meno esperti.

- *Semplicità e flessibilità dell'interfaccia grafica*: L'interfaccia grafica implementata in un codice necessita di essere estremamente semplice e flessibile. Maggiore è la facilità e l'intuibilità di un codice e maggiore sarà la sua desiderabilità da parte degli utilizzatori.
- *Efficienza del codice*: Affinché un codice si possa riutilizzare, è necessario che oltre ad essere affidabile sia anche efficiente. Un codice è definito efficiente quanto ottimizza il più possibile la propria esecuzione all'interno di un calcolatore. Stabilità ed efficienza sono due concetti nettamente diversi. Un codice può essere stabile ma non efficiente o viceversa.
- *Applicabilità del codice*: Un codice deve essere particolarmente flessibile nella sua applicabilità. Un codice che descrive funzioni o comportamenti troppo specifici non può essere riutilizzato in diverse applicazioni.

Queste caratteristiche non si applicano solamente ai modelli ad oggetti; possono essere impiegate per valutare la riutilizzabilità di qualunque codice.

3.1.3.4 Caratteristiche degli oggetti

Finora si sono analizzati gli oggetti come unità di alto livello ciascuno contenente le proprie istanze ed i propri metodi tipici della classe di appartenenza. Si proceda ora con l'analisi delle principali caratteristiche degli oggetti. Gli oggetti godono di tre caratteristiche fondamentali:

- *Incapsulamento*
- *Poliformismo*
- *Ereditarietà*

1. Incapsulamento

L'*incapsulamento* riguarda un concetto che è già stato analizzato durante la trattazione generale dell'argomento: la separazione tra *interfaccia* ed *implementazione*. In un'applicazione sviluppata ad oggetti, la barriera tra l'interfaccia e l'implementazione deve essere assoluta. L'interfaccia di un oggetto deve di norma incapsulare l'implementazione in modo da nasconderla completamente alle altre parti del programma. L'*incapsulamento* può essere interpretato come una protezione nei confronti di azioni indesiderate provenienti da altre parti di codice. All'interno di un oggetto, gli elementi maggiormente incapsulati sono i dati definiti con il termine di

variabili d'istanza. Esiste una forma di incapsulamento anche per i metodi ma è meno rilevante e meno applicata. Quando delle istanze sono incapsulate, risultano completamente invisibili all'ambiente esterno; gli unici in grado di vederle e poterle utilizzare sono i metodi dell'oggetto in questione. L'incapsulamento di una variabile d'istanza assume anche il nome di *occultamento dell'informazione*.

2. Poliformismo

Il *poliformismo* è la capacità di un oggetto di ricevere come argomenti informazioni di natura diversa. Lo stesso oggetto è in grado di ricevere come stesso argomento variabili aventi tipologia diversa (*float, int, ecc.*) Questa caratteristica si estende anche ai metodi ed alle istanze di ciascuna classe consentendo che classi diverse possano contenere variabili d'istanza e metodi aventi eguali denominazioni. Il principale beneficio derivante dal *poliformismo* è la possibilità di utilizzare lo stesso oggetto in modalità diverse. Questa caratteristica consente una maggiore flessibilità del codice che conduce a notevoli semplificazioni strutturali dei programmi; il risultato che ne deriva è un aumento dell'applicabilità del codice.

3. Ereditarietà

Il modo più facile per illustrare qualcosa di nuovo è partendo da qualcosa di vecchio. La stessa affermazione è vera se si desidera definire un nuovo tipo di oggetto: la sua descrizione sarà molto più semplice se si parte dalla definizione di un oggetto già esistente. I linguaggi di programmazione ad oggetti sfruttano questo principio e permettono la definizione di una nuova classe facendo uso di una classe già definita. La classe di base è definita in termini informatici *superclasse* mentre la nuova classe prende il nome di *sottoclasse*. La definizione di una *sottoclasse* comprende esclusivamente le differenze che si intendono apportare rispetto alla relativa *superclasse*; tutto ciò che non è nuovamente dichiarato è inteso uguale alla classe originaria. Le due classi (*superclasse* e *sottoclasse*) sono connesse tra di loro in modo che la *sottoclasse* erediti tutti i metodi e le variabili d'istanza della sua *superclasse*. Se la definizione della *sottoclasse* fosse vuota (senza quindi ulteriori dichiarazioni di metodi od istanze), le due classi risulterebbero esattamente identiche. Qualunque classe può essere utilizzata come *superclasse* per la definizione di nuove classi. Grazie a questa caratteristica, ciascuna classe può essere contemporaneamente una *sottoclasse* ed una *superclasse* di altre classi. La struttura che emerge dall'ereditarietà

delle classi può assumere forme estremamente complesse. Un esempio è rappresentato in figura.

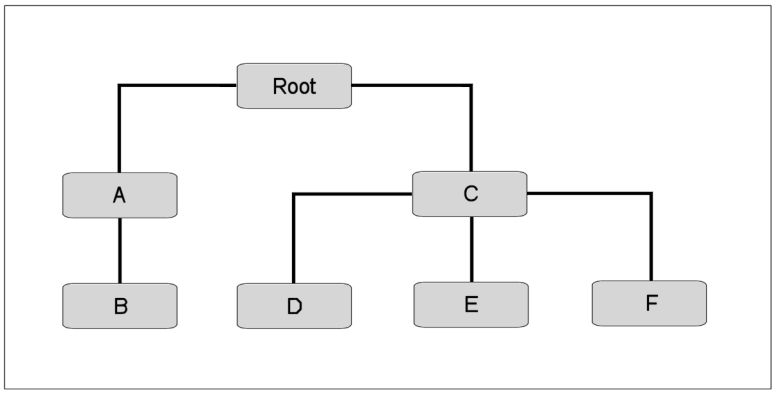


Figura 15 - Struttura gerarchica delle classi

Tutte le classi derivano da una classe radice definita *Root*. La classe *Root* non deriva da alcuna *superclasse* e rappresenta l'estremo gerarchico della struttura. I rami della gerarchia che discendono da esso si possono sviluppare in vari modi a seconda del tipo di applicazione che si intende sviluppare. Ciascuna nuova classe contiene tutte le caratteristiche delle classi gerarchicamente collocate al di sopra di essa e generalmente deriva da una sola *superclasse*. Sebbene questa struttura sia considerata abituale, nulla vieta ad una nuova classe di ereditare le caratteristiche di due o più classi; in questo caso si parlerà di più *superclassi*. Una *sottoclasse* può contenere tre tipologie di modifiche rispetto alla sua *superclasse*:

- *Espansione:* è possibile espandere la definizione di una classe aggiungendo nuovi metodi e nuove variabili d'istanza. Questa rappresenta la motivazione più comune per la quale si utilizza la definizione di nuove sottoclassi.
- *Modifica:* è possibile modificare l'implementazione di alcuni metodi od il contenuto di alcune variabili, mediante la sostituzione con delle versioni più aggiornate. Questa operazione è attuabile implementando un nuovo metodo avente la stessa denominazione di quello obsoleto appena ereditato. La nuova versione sovrascriverà completamente quella vecchia.
- *Modifica parziale:* è possibile modificare l'implementazione di alcuni metodi sostituendoli con delle versioni più aggiornate in grado di mantenere però gli elementi ereditati. I vecchi metodi saranno incorporati nella nuova versione ma saranno attivabili solo mediante l'invio di un messaggio specifico ad essi diretto.

Le *sottoclassi* possono essere concepite come delle integrazioni alle relative *superclassi* allo scopo di renderle più specifiche e definite. Il loro utilizzo è molto vario e dipende strettamente dalle applicazioni che si intendono sviluppare.

3.1.4 Struttura dei programmi

I linguaggi di programmazione ad oggetti sono caratterizzati da due tipi di struttura: una *struttura gerarchica* esistente tra le classi ed una *struttura funzionale* esistente tra i diversi oggetti.

- La ***struttura gerarchica*** mostra come gli oggetti sono interconnessi in base alle loro caratteristiche (*variabili d'istanza* e *metodi*). Questo argomento è già stato in parte discusso nel paragrafo precedente.

- La ***struttura funzionale*** mostra come gli oggetti sono interconnessi in base al sistema di messaggistica. Questo tipo di struttura illustra il vero funzionamento del programma evidenziando le interrelazioni esistenti a livello funzionale tra i singoli oggetti.

3.1.4.1 Struttura delle classi

Il primo aspetto fondamentale da prendere in considerazione durante la fase di progettazione riguarda la struttura delle classi. La domanda che occorre porsi è: "Quando aggiungere una nuova funzionalità ad una classe già esistente e quando creare invece una nuova classe o sottoclasse?" Si immaginino due casi estremi:

- Da un lato lo sviluppo di un programma consistente in un solo oggetto appartenente ad una sola classe. Dal momento che sarebbe l'unico oggetto presente nel modello, potrebbe inviare messaggi solo a se stesso. L'intera struttura del programma sarebbe contenuta all'interno della classe di appartenenza senza che si possa trarre alcun vantaggio né dal poliformismo né tanto meno dalla modularità, caratteristiche entrambe fondamentali per questa tipologia di programmazione.

- Dall'altro lato lo sviluppo di un programma avente centinaia di oggetti diversi, ciascuno con i propri metodi e le proprie variabili. Ogni oggetto avrebbe le caratteristiche della propria classe di appartenenza e la struttura risulterebbe talmente complessa tale da rendersi praticamente incomprensibile. Si tratterebbe di un vero e proprio labirinto di oggetti.

Il problema della scelta di quante classi costruire ed utilizzare risulta sempre di notevole rilevanza. Evitando i due casi estremi, le scelte migliori consistono normalmente nel mantenere le classi sufficientemente ampie da poter essere riutilizzate in altre applicazioni ma anche sufficientemente piccole tali da non perdere il loro ruolo predefinito, motivo per cui sono state concepite.

Uno dei problemi di maggior rilevanza è certamente quello di scegliere tra la modifica di una classe già esistente o la costituzione di una classe in tutto o in parte nuova. La scelta dipende da molteplici fattori tra i quali gli stessi obiettivi che si intendono perseguire mediante la realizzazione del programma. La scelta delle classi deve essere effettuata in base al sistema che si intende riprodurre all'interno del calcolatore mediante realtà artificiale. Non bisogna dimenticare che lo scopo delle simulazioni è quello di riprodurre la realtà, e gli oggetti inseriti all'interno del modello devono rappresentare gli elementi del mondo reale. Nello sviluppo di un modello deve esistere una stretta interrelazione tra mondo reale e classi. Se tale relazione non esiste, il modello di simulazione rappresenta una realtà che è distorta rispetto a quella effettiva.

3.1.4.2 Struttura degli oggetti

Una parte delle operazioni che devono essere svolte in fase di implementazione del codice (*design time*) riguarda la progettazione di un'adeguata struttura di oggetti. Questa struttura deve essere concepita in maniera dinamica in modo da potersi adattare in base alle esigenze richieste dal programma. Esistono connessioni tra oggetti appartenenti allo stesso livello oppure tra oggetti gerarchicamente diversi. Le connessioni tra gli oggetti possono essere di due tipologie:

- ***Estrinseche**:* in questa categoria rientrano tutte le connessioni esplicitamente dichiarate. Rappresentano il sistema di organizzazione del programma ad alto livello e tengono traccia di tutte le connessioni tra i componenti indipendenti del sistema.

- ***Intrinseche**:* in questo gruppo rientrano tutte le connessioni tra oggetti non dichiarate esplicitamente nel codice. Quando un oggetto è scaricato dalla memoria o salvato in un file su disco, gli oggetti che con questo interagiscono intrinsecamente, decadono a loro volta.

La rete di oggetti così strutturata non avrebbe molta utilità se poi non fosse attivata. L'attivazione di qualunque struttura di oggetti è effettuata tramite un impulso esterno. Nel caso di un'applicazione interattiva, l'attivazione può avvenire tramite interfaccia grafica o mediante l'utilizzo di una tastiera o di un mouse. Alcuni programmi possono essere attivati anche tramite linea di comando mentre altri addirittura tramite chiamate telefoniche o chiamate effettuate da altre applicazioni. I modi per attivare un'applicazione sono diversi ma non è questa la sede per intraprendere una simile trattazione.

3.2 Objective C

Questo paragrafo descrive i principi base del linguaggio di programmazione ad oggetti denominato *Objective C*. La sintassi dell'*Objective C* è in gran parte analoga a quella del *C* standard ma la maggioranza dei compilatori *C* esistenti su piazza non sono in grado di compilare entrambi i linguaggi. L'unico compilatore attualmente disponibile in grado di compilare entrambi i linguaggi è *GCC*: un ottimo compilatore rilasciato sotto licenza *GNU*. I file del *C* standard hanno tipicamente estensione *".c"* mentre i file di *Objective C* hanno generalmente estensione *".m"* anche se l'*Objective C* è anche in grado di utilizzare come estensioni quelle utilizzate dal *C++*.

3.2.1 Oggetti

In *Objective C* gli oggetti rispettano la struttura generale già descritta nel precedente paragrafo. Ciascun oggetto contiene le proprie informazioni ed i propri comportamenti. I comportamenti sono rappresentati dai *metodi* mentre i dati sono costituiti dalle *variabili d'istanza*. Le variabili d'istanza sono contente all'interno dell'oggetto e l'unico modo per accedervi è tramite i metodi dell'oggetto stesso. In *Objective C* gli oggetti sono identificati da un particolare tipo di dichiarazione rappresentata dalla voce **id**. Questo simbolo indica quello che in termini informatici è definito puntatore. Tutti gli oggetti sono implicitamente dichiarati di tipo **id**. La dichiarazione avviene nel seguente modo:

```
id anObject;
```

La definizione di tipo **id** non fornisce alcuna informazione sull'oggetto in se eccetto il fatto che si tratta di un oggetto. Gli oggetti non sono però tutti uguali. Un programma potrebbe aver bisogno di maggiori informazioni sull'oggetto in questione, soprattutto su cosa contiene al suo interno: quali sono le sue *variabili d'istanza* e quali i suoi *metodi*. Dal momento che la dichiarazione di un oggetto tramite la tipologia generica **id** non è in grado di fornire queste informazioni, ciascun oggetto dovrà essere in grado di fornirle su richiesta durante l'esecuzione dell'applicazione (*runtime*). Questa operazione è possibile grazie ad una variabile d'istanza definita **isa** che ogni oggetto porta con se. Tramite questa variabile è possibile identificare la classe di appartenenza dell'oggetto in questione. Gli oggetti dotati di tali caratteristiche sono definiti dinamici ed il sistema è in grado di identificare la loro tipologia di appartenenza mediante semplice interrogazione.

3.2.2 Messaggi

Anche per quanto riguarda il sistema di messaggistica, l'*Objective C* non si scosta molto da quelle che sono le regole generali. Per consentire l'esecuzione dei metodi è sufficiente inviare loro un messaggio tramite il quale si specifica il metodo ed i relativi argomenti. La sintassi adottata in è la seguente:

```
[receiver message];
```

Il ricevente è rappresentato da un oggetto mentre il messaggio descrive ciò che l'oggetto deve compiere ed è rappresentato dal nome di un metodo con i relativi argomenti, se ne possiede:

```
[myObject setOption:1 :2];
```

Il metodo **setOption**, appartenente all'oggetto *myObject*, possiede due argomenti: al primo è associato il numero uno, al secondo il numero due. Sono anche ammessi metodi che utilizzano un numero di argomenti variabile ma il loro utilizzo è estremamente raro. I metodi sono anche utilizzati per restituire i valori provenienti dalle variabili interne dell'oggetto. La sintassi in questo caso è la seguente:

```
myVariable = [myObject myVariable];
```

Come si può notare dall'esempio, una variabile ed un metodo possono anche avere la stessa denominazione, ma di questa caratteristica si è già discusso a sufficienza. In *Objective C* è anche ammessa la nidificazione dei messaggi.

```
[myObject setOption: [myNewObject setNewOption]];
```

3.2.3 Classi

Un programma comprende generalmente una notevole varietà di oggetti e spesso accade che più di un oggetto appartenga alla stessa classe. In *Objective C*, gli oggetti sono definiti tramite le classi. La definizione di una classe rappresenta la forma generatrice per un particolare tipo di oggetto: dichiara le variabili d'istanza che divengono parte di ciascun membro della classe e definisce un insieme di metodi che tutti gli oggetti appartenenti a quella classe ed alle eventuali sottoclassi potranno utilizzare. Gli oggetti creati da una classe prendono il nome di *istanze della classe*.

Le classi scritte in *Objective C* sono additive e ciascuna nuova classe eredita le caratteristiche della relativa superclasse. Ciascuna classe può essere considerata una specializzazione od un'estensione di un'altra classe. Anche in *Objective C*, così come avviene per tutti i linguaggi di programmazione ad oggetti, esiste una classe considerata radice. La superclasse alla quale fanno capo tutte le classi prende il nome di ***NSObject***. Questa classe definisce la struttura base di tutti gli oggetti comprese le loro interazioni. La sua attribuzione principale è quella di impartire alle classi ed alle istanze che interagiscono con esse, la capacità di comportarsi come oggetti in grado di cooperare tra loro durante la fase di esecuzione (*runtime*). Grazie all'ereditarietà tra le classi, un oggetto ha accesso non solo ai metodi definiti dalla propria classe ma anche a tutti i metodi definiti dalla sua superclasse la quale a sua volta avrà ereditato altri metodi dalle sue classi gerarchicamente superiori.

L'Objective C utilizza delle classi particolari definite *astratte*. Le classi *astratte* raggruppano i metodi e le variabili d'istanze che saranno utilizzati da diverse sottoclassi in modalità comune. Le classi astratte sono formalmente incomplete per un corretto funzionamento indipendente, ma contengono il codice necessario per ridurre l'implementazione delle relative sottoclassi.

La definizione di una classe non è altro che la specificazione di un tipo di oggetto e, quando si desidera definire un oggetto, è sufficiente utilizzare il nome di una classe al posto del generico identificativo **id**.

```
Class *myClass;
```

Questa metodologia di dichiarazione è nota come *assegnazione statica*. Così come **id** rappresenta il puntatore ad un oggetto, gli oggetti sono staticamente assegnati come puntatori a delle classi. Gli oggetti, in un modo o nell'altro, sono sempre definiti come dei puntatori. L'assegnazione statica rende il puntatore esplicito mentre **id** lo rende nascosto. Il grande vantaggio che presenta l'assegnazione statica rispetto a quella generica è che permette di eseguire alcuni controlli specialmente in fase di compilazione. Grazie alla dichiarazione esplicita, il compilatore è in grado di individuare immediatamente la coerenza dei vari oggetti ed istanze contenuti nel modello.

Ci sono alcuni casi in cui le classi necessitano di essere inizializzate. Quando il sistema è in esecuzione, un messaggio di inizializzazione è inviato tramite il metodo **initialize** a ciascuna classe. Questa operazione imposta l'ambiente della classe prima che questo sia utilizzato.

3.2.4 Definizione di una classe

In *Objective C* le classi sono definite in due parti:

- L'***interfaccia*** che dichiara tutti i metodi e le variabili d'istanza della classe (compresi i nome delle proprie superclassi nel caso ce ne sia più di una).

- L'***implementazione*** tramite la quale si definiscono i metodi e le variabili della classe.

Nonostante i compilatori non lo richiedano, l'interfaccia e l'implementazione sono generalmente separati in due file distinti. Per convenzione è bene associare ad ogni file una ed una sola classe ma nulla vieta che all'interno dello stesso file siano dichiarate più di una classe. Il motivo per cui è preferibile abbinare ad ogni file una sola classe, risiede nella maggior chiarezza logistica. Troppe classi contenute nello stesso file rischiano di complicare notevolmente il lavoro e la chiarezza, soprattutto in fase post-

scrittura. Generalmente, i file dell'interfaccia e dell'implementazione sono nominati con lo stesso nome attribuito alla classe. Il motivo di questa scelta è identico a quello appena citato: maggiore chiarezza strutturale.

In *Objective C*, i file di interfaccia hanno di norma estensione *".h"* e contengono il nome di tutte le variabili e di tutti i metodi che si intendono implementare nella classe. Oltre a queste informazioni, contengono anche il nome di tutte le superclassi necessarie per generare le interdipendenze. I file di implementazione assumono tipicamente estensione *".m"*. In questi file è implementato tutto il codice che descrive il contenuto delle variabili ed il comportamento imposto a ciascun metodo.

3.2.4.1 L'interfaccia

La dichiarazione dell'interfaccia di una classe inizia tramite l'istruzione **@interface** e termina con l'istruzione **@end**.

```
@interface ClassName: ItsSuperClass
{
      instance variable declarations
}
method declarations
@end
```

La prima riga rappresenta il nome della classe con la rispettiva superclasse di riferimento. Come già accennato più volte, la superclasse definisce la posizione della nuova classe all'interno della gerarchia. Se non è specificata alcuna superclasse, la nuova classe sarà dichiarata come classe radice rivale della classe *NSObject*. Per questa ragione occorrerà prestare molta attenzione quando si dichiarano nuove classi. I primi elementi ad essere dichiarati sono le variabili d'istanza. Queste sono inserite all'interno delle parentesi graffe e sono definite a seconda della natura che le si vuole attribuire.

```
float width;
float height;
int filled;
```

Dopo le variabili d'istanza, fuori dalle parentesi graffe ma prima dell'istruzione **@end**, sono dichiarati i metodi relativi alle classi. I metodi che possono essere utilizzati dagli oggetti della classe sono definiti *metodi della classe* e sono preceduti dal simbolo "+" mentre i metodi che possono essere utilizzati dalle istanze di una classe sono definiti *metodi delle istanze* e sono preceduti dal simbolo "-". Prima del nome di ciascun metodo è anche possibile specificare il tipo di dati che gli potranno essere comunicati.

```
+ alloc;
- (void)display;
- (float)options;
```

Può accadere che un file di interfaccia debba disporre di alcuni metodi o variabili presenti in un'altra classe. In tal caso è necessario procedere con l'importazione dell'interfaccia richiesta. Questa operazione è facilmente attuabile tramite l'istruzione **#import**:

```
#import "OtherClass.h"
```

In tal modo, l'interfaccia contenuta nel file *"OtherClass.h"*, è importata all'interno della nuova classe. Questa istruzione è identica all'istruzione **#include** eccetto per il fatto che lo stesso file non è mai incluso più di una volta. Nonostante le due istruzioni assolvano alla stessa funzione, l'istruzione **#import** è spesso preferita all'istruzione **#include**.

Il file d'interfaccia contiene tutte le informazioni necessarie affinché la classe funzioni correttamente in accordo con gli scopi perseguiti. Schematizzando, si possono individuare tre funzioni attribuibili ad un file d'implementazione:

1) Il file d'interfaccia comunica agli utilizzatori come la classe sia connessa con le altre classi indicando le relative *superclassi* e *sottoclassi*.
2) Il file di interfaccia permette al compilatore di sapere quante e quali variabili d'istanza contiene un oggetto.
3) Attraverso la lista dei propri metodi, il file d'interfaccia permette agli altri moduli di sapere quali messaggi possono essere inviati alla classe ed alle sue istanze.

3.2.4.2 L'implementazione

La definizione di una classe è strutturata esattamente come la sua dichiarazione. L'eccezione principale risiede ovviamente nei suoi contenuti e nell'istruzione utilizzata per iniziare l'implementazione. L'istruzione con cui si inizia l'implementazione è **@implementation** mentre l'istruzione tramite la quale si termina è **@end**.

```
@implementation ClassName: ItsSuperClass
{
      instance variable declarations
}
method definitions
@end
```

Prima di procedere con l'implementazione di una classe è necessario importare la propria interfaccia tramite l'istruzione **#import** già trattata. Dal momento che l'interfaccia importa implicitamente tutte le istanze ed i metodi delle relative superclassi, risulta del tutto inutile ripetere ulteriormente la dichiarazione delle suddette. Questa caratteristica semplifica notevolmente l'implementazione della nuova classe e rende il compito del programmatore molto più semplice ed intuitivo.

```
#import "ClassName.h"

@implementation ClassName
      method definitions
@end
```

Ciascun metodo della classe è definito come una funzione scritta in linguaggio *C* ed il codice è compreso tra due parentesi graffe che ne delimitano l'estensione. Anche in *Objective C* la definizione dei metodi ammette l'utilizzo di argomenti.

```
+ alloc
{
      .  .  .
}
- (void)setOptions: (BOOL)flag
{
      .  .  .
```

```
}
- (float)getState
{
      .   .   .
}
```

I metodi non sono solo utilizzati per eseguire qualcosa di specifico all'interno dell'applicazione, ma anche per impostare le variabili d'istanza contenute all'interno dell'oggetto o per recuperare i valori ivi contenuti. Nel primo caso il codice utilizzato sarà il seguente:

```
- (void)setOptions: (BOOL)value
{
      options = value;
}
```

Il valore contenuto nella variabile *value*, quando sarà passato attraverso il metodo *setOptions*, sarà attribuito alla variabile d'istanza *options* contenuta all'interno dell'oggetto in questione. Se l'oggetto ha nome *myObject*, il codice da utilizzare per eseguire questa operazione sarà il seguente:

```
[myObject setOptions: value];
```

Per recuperare i valori contenuti in una variabile d'istanza di un oggetto, il codice utilizzato sarà il seguente:

```
- (float)getState
{
      return state;
}
```

Il valore restituito dal metodo *getState* sarà quello contenuto nella variabile *state*. Il codice da utilizzare per svolgere l'operazione indicata è il seguente:

```
stateValue = [myObject getState];
```

Il valore restituito dall'oggetto *myObject* sarà inserito nella variabile *stateValue*.

3.3 Swarm

A differenza di molti strumenti di simulazione presenti sul mercato, *Swarm* rappresenta uno strumento di simulazione unico nel suo genere. Consente lo sviluppo di potenti simulazioni senza introdurre i classici elementi restrittivi tipici di altri strumenti maggiormente commerciali. I pacchetti che spesso si trovano in circolazione forniscono degli strumenti informatici caratterizzati da un'estrema facilità d'uso ma da una minore flessibilità nelle possibilità d'impiego. *Swarm*, nonostante rappresenti uno strumento decisamente articolato e complesso, probabilmente non adatto per un utente poco esperto, è certamente più completo e permette di realizzare diverse tipologie di modelli. I campi di applicazione sono molteplici e le istituzioni che attualmente utilizzano Swarm sono in numero sempre crescente. Per citarne alcune: l'economia, la finanza, la sociologia, la psicologia, la matematica, la statistica e molte altre ancora.

3.3.1 Cenni storici

Il progetto *Swarm* fu originariamente avviato da Chris Langton nel 1994 presso l'istituto di Santa Fe in Messico. Assieme a lui, molti furono gli sviluppatori che fornirono il proprio contributo a questo progetto. Per citarne alcuni: Roger Burkhart, Nelson Minar, Glen Ropella, Manor Askenazi, Irene Lee, Vladimir Jojic, ed Alex Lancaster. L'organizzazione alla quale fa capo il progetto è attualmente promossa come *non profit*. *Swarm* è un software completamente libero e l'attuale distribuzione è rilasciata sotto licenza ***GNU*** (*General Pubblic License - GPL*).

L'obiettivo del progetto è di sviluppare tanto un vocabolario quanto una collezione di strumenti informatici atti alla realizzazione di modelli di simulazione multiagenti. Armati di queste idee, gli sviluppatori del progetto sono stati in grado di posare la propria attenzione sulla vera sostanza dei modelli simulativi evitando alcune delle complicazioni derivanti dai più comuni linguaggi di programmazione. Originariamente il progetto *Swarm* fu concepito per sistemi operativi *Unix* in grado di supportare l'ambiente grafico *X Windows*. La prima versione beta di *Swarm* fu rilasciata nel 1995. Nel Gennaio del 1997 fu finalmente rilasciata la versione 1.0 la quale però funzionava solamente su sistemi *Solaris* e *Linux*. Nell'Aprile del 1998 fu rilasciata la versione 1.1 e, grazie al pacchetto *Cygnus Win32*, *Swarm* poteva finalmente essere utilizzato anche su sistemi operativi *Windows 95/NT*. Verso la fine del 1999 fu rilasciata la versione 2.0 nella quale si introduceva il supporto per il linguaggio *Java* che fino a quel momento

era stato trascurato. Questa innovazione permise ai programmatori *Java* l'introduzione di tutte le librerie *Swarm* all'interno dei propri modelli.

3.3.2 Caratteristiche fondamentali

Swarm è una collezione di librerie informatiche in grado di fornire un ottimo supporto per le simulazioni effettuate tramite linguaggi di programmazione. Le principali caratteristiche di *Swarm* confermano quanto già sostenuto riguardo i linguaggi di programmazione ad oggetti.

- *Il codice Swarm è un codice orientato agli oggetti*: le librerie di *Swarm* sono scritte con linguaggi di programmazione ad oggetti e le ultime versioni utilizzano tanto *Objective C* quanto *Java*.
- *Il programmi Swarm sono gerarchici*: la maggioranza dei programmi *Swarm* utilizzano una struttura gerarchica come quella già analizzata nei paragrafi precedenti relativa ai linguaggi di programmazione ad oggetti.
- *Swarm fornisce diversi strumenti di implementazione*: le librerie di *Swarm* forniscono una moltitudine di funzioni congegnate appositamente per facilitare lo sviluppo di modelli basati su agenti. Questi strumenti facilitano la gestione della memoria, la programmazione delle azioni e tutte le operazioni necessarie per una buona simulazione. Gli utenti che intendono sviluppare i propri modelli simulativi non devono far altro che incorporare gli oggetti *Swarm* all'interno dei loro programmi.

3.3.3 Funzionamento di Swarm

Swarm è stato sviluppato per la creazione di oggetti informatici su base gerarchica. Il primo oggetto ad essere creato è generalmente l'*Observer* il quale crea a sua volta l'interfaccia utente ed inizializza le istanze del *ModelSwarm*; il *ModelSwarm* crea a sua volta i livelli sottostanti compresa la programmazione delle azioni e delle attività. I modelli *Swarm* utilizzano una sintassi abbastanza semplice in grado di fornire agli utilizzatori un valido strumento per comprendere il modo in cui le varie parti del modello interagiscono tra loro. L'*ObserverSwarm* ed il *ModelSwarm* sono sviluppati con la stessa metodologia. L'oggetto *ModelSwarm* è definito come una sottoclasse dell'oggetto *Swarm* e rappresenta l'oggetto al quale è attribuito il compito di costruire

gli agenti. Il *ModelSwarm* è anche in grado di fornire a ciascun agente un indirizzo di memoria nonché programmare le proprie attività.

L'allocazione e la deallocazione della memoria è un requisito assolutamente necessario per la realizzazione di un progetto di simulazione. *Swarm*, grazie alla sua elevata flessibilità ed affidabilità, è in grado di risolvere egregiamente questo problema. In *Swarm*, gli oggetti sono creati mediante un processo che in termini informatici prende il nome di *allocazione in zone di memoria* e l'arduo compito di allocare memoria per i singoli oggetti è gestito direttamente dalle librerie. Gli oggetti sono creati ed allocati in memoria solo quando ne è richiesta la loro presenza; quando non sono più necessari, il programma invierà all'oggetto un messaggio di **drop** che provvederà a rimuoverlo dalla memoria.

In *Swarm* esistono due metodi considerati fondamentali: il metodo **buildObjects** ed il metodo **buildActions**.

- Nell'implementazione del metodo **buildObjects** si introducono normalmente le istruzioni necessarie alla classe per creare i propri oggetti. Tramite il metodo **buildObjects** si possono anche eseguire istruzioni dedicate alla creazione di oggetti grafici nonché alla creazione del pannello grafico necessario per il controllo della simulazione.

- Il metodo **buildActions** crea generalmente gli oggetti di due classi fondamentali: l'*ActionGroup* e lo *Schedule*. L'*ActionGroup* è l'oggetto che contiene l'insieme di azioni ed eventi che devono essere svolti simultaneamente durante la fase di simulazione. Lo *Schedule* è l'oggetto che contiene la sequenza tramite la quale è controllata l'esecuzione delle azioni inseriti nell'*ActionGroup*.

3.3.3.1 La creazione degli oggetti

Quando si desidera sviluppare una simulazione in un ambiente *Swarm*, occorre innanzitutto creare gli oggetti necessari alla simulazione. I metodi che in *Swarm* sono maggiormente utilizzati per la creazione gli oggetti sono **createBegin**, **createEnd** e **create**.

- Il metodo **createBegin** è utilizzato per creare gli oggetti ed attribuire loro una specifica zona di memoria. Nel metodo **createBegin** si collocano normalmente le istruzioni che definiscono in modo permanente le caratteristiche degli oggetti.

- Il metodo **createEnd** è utilizzato per contenere tutte le istruzioni che integrano l'inizializzazione di un oggetto. La sua funzione principale è di assicurare che tutte le variabili inizializzate siano impostate correttamente.

- Il metodo **create** svolge simultaneamente entrambi i metodi. Esistono alcuni casi in cui le variabili ed i metodi che devono essere impostati sono contenuti all'interno dell'oggetto: in questo caso è richiesto l'utilizzo dei due metodi in forma separata. Là dove non sono necessarie particolari impostazioni su variabili e metodi, è possibile utilizzare il metodo più semplice denominato **create**.

3.3.3.2 L'interfaccia grafica

Swarm è in grado di fornire un elevato numero di classi e protocolli volti alla creazione di un'adeguata interfaccia grafica durante la fase di simulazione. Il primo elemento da prendere in considerazione è il *probe* grafico.

Il *probe* grafico permette all'utente di osservare le caratteristiche di un oggetto selezionato durante la fase di simulazione. In *Swarm* esistono due tipi distinti di visualizzazione che l'utente può utilizzare:

1) ***DefaultProbMaps:*** è il tipo di *probe* utilizzato nel caso in cui non si specifica un particolare tipo di *probe*. E' considerato *probe* predefinito.

2) ***CustomProbMaps:*** è un tipo di probe personalizzato. Quando si adotta questo metodo, l'interfaccia grafica visualizzerà solamente ciò che l'utente ha specificato. Tramite questo metodo si può impedire la visualizzazione di alcune variabili e consentire la visualizzazione di altre che altrimenti risulterebbero nascoste.

La scelta dell'interfaccia è un elemento fortemente condizionato dalle caratteristiche della simulazione.

3.3.3.3 Lo Schedule

L'apparato di scheduling rappresenta uno dei più importanti elementi di un modello *Swarm*. Consente di integrare le azioni di molti agenti in diversi livelli di una simulazione. Le azioni che si svolgono durante una simulazione sono dirette da un oggetto denominato *Schedule* che utilizza un particolare protocollo definito appunto *Schedule Protocol*. Lo *Schedule* è un oggetto creato mediante il metodo **buildActions** e può essere considerato come un'agenda dove ad ogni giorno è attribuito un certo impegno od una certa azione da eseguire. Ciò che l'utente deve fare è disporre le azioni che ad ogni istante della simulazione dovranno essere eseguite. Un esempio di *Schedule* può essere il seguente:

```
- buildActions
{
      modelSchedule=[Schedule createBegin: self];
      [modelSchedule setRepeatInterval: 1];
      modelSchedule = [modelSchedule createEnd];

      [modelSchedule at: 0 createActionTo: aBug message:
      M(step)];

      return self;
}
```

Le prime tre righe sono dedicate alla creazione dell'oggetto *Schedule*. Tra i metodi **createBegin** e **createEnd** sono specificate le caratteristiche dello *Schedule* che in questo caso sono rappresentate dall'impostazione del solo intervallo di ripetizione fissato pari ad uno; in tal caso tutte le azioni inserite nello *Schedule* saranno eseguite ad ogni istante della simulazione (*tic*).

Una volta che l'oggetto è stato creato ed impostato, è possibile inserire tutte le operazioni che si desidera siano eseguite. A tal fine si adottano normalmente due metodi a seconda che si intenda controllare un singolo oggetto oppure un'intera lista di oggetti. Nel primo caso si utilizzerà il metodo **at:createActionTo:message**, nel secondo si utilizzerà il metodo **at:createActionForEach:message**.

3.3.3.4 Le liste

Un elemento di fondamentale importanza per l'ambiente *Swarm* è senz'altro l'utilizzo delle liste. In *Swarm*, le liste sono una tipologia di oggetto appartenente alla classe *List*. Una lista è un oggetto all'interno del quale è contenuta una precisa struttura di singoli oggetti: per ciascuno di essi sono definite la propria zona di memoria e la propria posizione nei confronti degli altri oggetti. Le liste in *Swarm* possono avere diversi utilizzi:

1) Sono utilizzate per gestire collezioni di oggetti mediante la programmazione delle rispettive attività.
2) Possono essere utilizzate per trasferire informazioni tra diversi livelli di una simulazione.
3) Possono essere utilizzate da singoli agenti per tener traccia delle proprie esperienze e per gestire le informazioni in loro possesso.

In una lista è possibile aggiungere degli oggetti, recuperare le caratteristiche di un oggetto collocato in una particolare posizione e rimuovere tutti o solo in parte gli oggetti già presenti al suo interno. La sintassi utilizzata per la creazione di un oggetto *Lista* è la seguente:

```
id myList;
myList = [List create: self];
[myList addFirst: myName];
[myList addLast: yourName];
[myList removeFirst];
[myList removeLast];
[myList addFirst: myName];
[myList deleteAll];
```

L'oggetto *Lista* è creato utilizzando il semplice metodo **create**. Dopo che l'oggetto *Lista* è stato creato, è possibile eseguire su di esso tutte le operazioni che si desiderano. I metodi caratteristici di quest'oggetto sono:

- **addFirst**: aggiunge un elemento all'inizio della lista
- **addLast**: aggiunge un elemento alla fine della lista
- **removeFirst**: rimuove il primo elemento della lista
- **removeLast**: rimuove l'ultimo elemento della lista
- **removeAll**: estrae tutti gli elementi della lista senza cancellarli
- **deleteAll**: cancella tutti gli elementi della lista.

Capitolo 4
Il modello *SUM* (*Surprising Unrealistic Market*)

4.1 Introduzione

Il modello *SUM* è un modello virtuale implementato su calcolatore con la finalità di simulare un mercato borsistico a scopo di ricerca. Il funzionamento del modello *SUM* è rappresentato da una struttura computerizzata che riproduce il comportamento del sistema borsistico computerizzato di un reale mercato dei capitali. In termini economici è definito *book*.

Degli agenti artificiali, endogeni al modello, inviano al book i propri ordini di acquisto o vendita con i relativi prezzi. Il book, se le condizioni di contropartita lo permettono, esegue immediatamente gli ordini presenti procedendo così all'aggiornamento del listino e quindi del prezzo. Se le condizioni non permettono abbinamenti sugli ordini, questi vengono registrati rispettivamente nella sezione denaro o lettera accodandoli a quelli già esistenti. Al termine della giornata virtuale il *book* viene chiuso. All'inizio della giornata successiva, il book è nuovamente azzerato e gli ordini presenti e non ancora eseguiti sono rimossi in modo da consentire l'immissione di nuovi ordini. Il modello si sviluppa su due fasi: una prima fase definita di *preapertura* ed una seconda fase definita continua. Nella fase di preapertura, gli agenti possono inserire i propri ordini tutti contemporaneamente; al termine dell'inserimento ordini, il *book* eseguirà i contratti abbinabili. Nella fase di contrattazione *continua*, gli ordini possono essere inseriti sequenzialmente uno alla volta ed il *book* abbina gli ordini ad ogni nuovo inserimento.

Gli agenti introdotti nel modello possono appartenere a diverse tipologie che in seguito saranno illustrate dettagliatamente ed ogni agente opera in base alle caratteristiche determinate in fase di programmazione. L'intera popolazione di agenti determina l'andamento del prezzo del titolo ed una combinazione di agenti sufficientemente diversificata è anche in grado di generare una notevole volatilità nelle quotazioni.

Swarm, grazie alla sua struttura particolarmente elastica e flessibile, rappresenta uno dei migliori strumenti adatti alla costruzione di una sistema multilivello come quello in esame.

4.2 Struttura del modello

Il modello è sviluppato utilizzando le librerie di *Swarm* ma il linguaggio di programmazione adottato è l'*Objective C*. Di seguito sono esposte le principali parti del modello. Per una analisi più dettagliata del codice si rinvia il lettore alla relativa appendice.

4.2.1 L'oggetto principale *(Main)*

Il modello poggia il proprio funzionamento su di una struttura a blocchi. Ogni blocco assolve ad una funzione particolare ed è associato ad una certa classe. Il blocco principale su cui poggia l'intera struttura del programma è rappresentato dall'oggetto *Main*. Al suo interno è contenuta tutta la sequenza base per l'avvio di qualunque modello *Swarm*:

1) Creazione dell'*Observer*
2) Creazione degli oggetti (tra cui il *ModelSwarm*)
3) Creazione di un piano di esecuzione (*Schedule*)
4) Attivazione del modello
5) Avvio della simulazione.

4.2.2 L'Observer *(ObserverSwarm)*

L'*ObserverSwarm* è situato a valle dell'oggetto *Main*. Al suo interno si trovano tutte le informazioni necessarie per controllare il modello durante la simulazione. Una parte di codice contenuto in questi file permette la visualizzazione su schermo e su file di tutti i dati emergenti dalla simulazione. Il modello prevede di norma delle impostazioni standard stabilite in fase di programmazione, ma l'utilizzatore è libero di modificare alcuni parametri per una propria personalizzazione delle variabili maggiormente rilevanti. Le variabili che possono essere modificate dall'utilizzatore sono:

- *displayFrequency:* determina la frequenza con cui le informazioni sono visualizzate sullo schermo tramite i relativi grafici.
- *stopAtDayNumber:* determina l'intera durata della simulazione.
- *displayPreviousDayMean:* permette di scegliere se visualizzare o no il prezzo medio di negoziazione del giorno precedente.

Le altre variabili permettono di visualizzare il grafico relativo al *book*, alla ricchezza degli agenti ed alle previsioni nonché la possibilità di salvare i dati emergenti dalla simulazione su file separati. Oltre a queste funzioni, un'altra parte di codice è dedicata allo sviluppo dei controlli necessari alla simulazione. Questa parte è definita pannello di controllo. I pulsanti disponibili su questo pannello consentono l'avvio e la pausa della simulazione, il salvataggio dei dati su file e l'uscita dal programma.

4.2.3 Il Nucleo del modello *(ModelSwarm)*

Il *ModelSwarm* è il nucleo di tutto il modello. Gerarchicamente è collocato a valle dell'*ObserverSwarm*, ma ad esso fanno capo tutte le altre classi e gli altri oggetti. Il *ModelSwarm* è direttamente collegato al *Book*, all'*EventGenerator* ed agli agenti. Al suo interno sono dichiarate tutte le variabili necessarie al modello compresi tutti gli oggetti e le azioni necessarie per eseguire la simulazione. Le funzioni del *ModelSwarm* possono essere sintetizzate in tre punti:

1) Dichiarazione ed impostazione di tutte le principali variabili del modello relative ai singoli oggetti.
2) Creazione di tutti gli oggetti, ognuno in base classe di appartenenza.
3) Impostazione della sequenza di azioni necessarie alla simulazione.

1. Dichiarazione ed impostazione delle variabili

La prima funzione attribuita al *ModelSwarm* è quella di definire ed impostare tutte le variabili comuni utilizzate dai singoli oggetti. Quando si parla di oggetti ci si riferisce ovviamente al *Book*, all'*EventGenerator* ed ai singoli agenti. Ad ogni variabile, dopo essere stata dichiarata, è attribuito un valore di base che può essere modificato dall'utente secondo le proprie esigenze. Le variabili sono così dichiarate sono pronte per essere trasmesse ai relativi oggetti. Questo compito spetta alla seconda funzione attribuita al *ModelSwarm*: la creazione e l'impostazione degli oggetti.

2. Creazione ed impostazione degli oggetti

La seconda funzione attribuita al *ModelSwarm* è la creazione e l'impostazione di tutti gli oggetti del modello. Anche in questo caso, quando si parla di oggetti, ci si riferisce al *Book*, all'*EventGenerator* ed ai singoli agenti. Innanzitutto sono create tutte le liste di agenti necessarie per la simulazione. Per ogni oggetto se ne esegue la creazione e

l'impostazione dei relativi parametri necessari al suo funzionamento ed i valori che sono forniti ai singoli oggetti sono quelli contenuti e fissati nelle variabili durante la fase precedente. Il primo oggetto ad essere creato è il *Book* seguito dall'*EventGenerator* ed infine dalle varie categorie di agenti. Il numero di agenti creati dipende dal numero fissato dall'utilizzatore in fase di presimulazione. Ogni volta che è creato un nuovo agente, questo è immediatamente aggiunto nell'apposita lista degli agenti.

3. Impostazione della sequenza delle azioni

L'ultima funzione attribuita al *ModelSwarm* riguarda l'impostazione delle azioni che devono necessariamente essere svolte durante la fase di simulazione. L'oggetto che si occupa di gestire la sequenza delle azioni prende il nome di *Schedule*. Le azioni che sono svolte ogni giorno sono divise in quattro gruppi, uno di carattere generale e tre relativi alle tre fasi della giornata di borsa:

- **Azioni di inizio giornata:** in questo gruppo sono inserite tutte quelle procedure che devono necessariamente essere eseguite tutte le volte ad inizio giornata. Riguardano essenzialmente il corretto svolgimento di tutta la simulazione:
  - *Incremento del giorno*
  - *Azzeramento del book*
  - *Incremento delle previsioni*
  - *Incremento della quota*

- **Azioni in fase di preapertura:** le procedure inserite in questo gruppo sono svolte solo in fase di preapertura e riguardano direttamente la simulazione della giornata borsisitica:
  - *Determinazione dell'evento*
  - *Azioni in preapertura di tutti gli agenti*

- **Azioni in fase di contrattazione continua:** le procedure appartenenti a questo gruppo riguardano esclusivamente la fase di contrattazione continua e sono pressoché analoghe a quelle già esposte nella fase di preapertura salvo qualche variante:
  - *Determinazione dell'evento*
  - *Azioni in continua di tutti gli agenti*

- **Azioni in fase di chiusura:** in questo gruppo sono inserite tutte quelle azioni che devono essere eseguite al termine della giornata virtuale ossia:
  - *Determinazione del prezzo medio della giornata*
  - *Determinazione della ricchezza dei singoli agenti*

Questi quattro gruppi di azioni sono eseguiti in base all'ordine prescritto dallo *Schedule*. Lo *Schedule* gestisce le azioni in base all'istante raggiunto dalla simulazione, istante che in termini informatici è definito *tic:*

- All'istante *0* lo *Schedule* esegue le azioni appartenenti al primo ed al secondo gruppo. Avanza di un giorno la simulazione, pulisce il *book* ed interroga tutti gli agenti della lista per sapere se intendono o no agire in fase di preapertura; in tal caso consentirà loro di agire.
- Negli istanti compresi tra l'istante *0* e l'istante (*numero agenti – 1)*, lo *Schedule* esegue le azioni appartenenti al terzo gruppo. Interroga l'agente di turno per sapere quale azione desidera intraprendere (acquisto o vendita), quale prezzo offrire e quale quantità.
- Nell'ultimo istante della giornata, lo *Schedule* esegue il quarto ed ultimo gruppo di azioni. Calcola il prezzo medio di contrattazione realizzato durante la giornata corrente e la ricchezza dei singoli agenti.

4.2.4 Il Generatore di Eventi *(EventGenerator)*

L'*EventGenerator* è il generatore di eventi introdotto nel modello. Il generatore di eventi, nonostante sia direttamente legato al *ModelSwarm*, rappresenta un elemento caratterizzato da un proprio funzionamento autonomo.

Una rappresentazione realistica del modello dovrebbe prevedere la generazione di eventi sociali di rilevanza economica tramite espressioni verbali. Una simile impostazione risulterebbe però molto complessa da realizzare mediante una simulazione al calcolatore, sia per quanto riguarda l'implementazione stessa del codice, che per quanto riguarda i problemi derivanti da una simile impostazione. La generazione di eventi in modalità puramente verbale creerebbe inoltre innumerevoli problemi di carattere interpretativo da parte degli agenti simulati. Il problema sarebbe: "Come spiegare ai singoli agenti il modo di interpretare gli eventi espressi tramite frasi verbali?" Non rappresenterebbe di certo un problema di facile soluzione. Per ovviare a questo problema, si è preferito rappresentare gli eventi in forma numerica.

Il generatore di eventi è inizializzato una sola volta all'inizio simulazione ed il suo stato iniziale è fissato pari a *0* (evento nullo). Ad ogni istante della simulazione (*tic*), il generatore di eventi è azionato dallo *Schedule* tramite il metodo *setEventState.* Il generatore di eventi calcola un numero casuale compreso tra *0* ed *1* e, grazie ad una probabilità fissata nella variabile *eventChangeProbability*, è in grado di determinare il cambio di stato. Dopo aver determinato il cambio di stato, il generatore di eventi sceglie con probabilità pari a *0.50* se tornare allo stato nullo oppure ad un nuovo stato attivo; in tal caso si procederà alla generazione di un numero casuale che rappresenterà l'entità del nuovo evento. I numeri casuali rappresentativi degli eventi possono essere generati secondo due modalità.

- Una ***prima modalità*** prevede che si possano generare solo numeri interi compresi tra *–1* e *+1*. In tal caso gli unici valori numerici possibili saranno *-1*, *0* e *+1*. L'evento attribuito al valore *–1* rappresenterà il peggior evento concepibile viceversa, l'evento associato al valore *+1*, rappresenterà il migliore evento possibile. L'evento associato al valore *0* prende il nome di *evento nullo* e rappresenta una situazione di neutralità.

- Una ***seconda modalità***, considerata molto più realistica, prevede che i numeri casuali rappresentativi degli eventi siano generati tramite una distribuzione normale con media pari a *0* e varianza pari a *0.15*. La scelta di generare eventi caratterizzati da una distribuzione normale risiede nella maggiore coerenza che il modello dimostra nei confronti della realtà che intende rappresentare. Osservando la realtà quotidiana, è decisamente più probabile riscontrare eventi di rilevanza contenuta che eventi di rilevanza notevole. Questa caratteristica vale sia per gli eventi considerati positivi che per gli eventi considerati negativi. L'introduzione di questa caratteristica complica leggermente il modello e fa sorgere un problema che merita particolare attenzione. Dal momento che una distribuzione normale può teoricamente generare tutti i valori compresi tra meno infinito e più infinito, come si potranno rappresentare gli eventi estremamente negativi e gli eventi estremamente positivi? La domanda è lecita. Una distribuzione normale genera effettivamente numeri casuali compresi tra meno infinito e più infinito, ma i risultati emergenti da alcune prove effettuate con una varianza pari a *0.15*, hanno dimostrato che raramente i valori generati eccedono entità pari ad *1* in valore assoluto. Questa caratteristica permette di considerare anche in questo caso il numero *1* come un valore soglia necessario

per la determinazione formale degli eventi. Gli eventi con entità maggiore di *+1* saranno considerati estremamente positivi mentre gli eventi con entità minore di *–1* saranno considerati estremamente negativi. Tutti i valori contenuti all'interno di tale intervallo rappresenteranno delle situazioni intermedie che, con incidenza diversa, influenzeranno le azioni degli agenti.

La variabile che contiene lo stato attuale dell'evento è contenuta all'interno del generatore ed è denominata *eventState*. Ogni volta che un oggetto avrà necessità di conoscere l'entità dell'evento in corso, sarà lui stesso che si rivolgerà al *generatore di eventi* per ottenere le informazioni di cui necessita. Questa operazione è effettuata tramite il metodo *getEventState* contenuto all'interno del generatore stesso. Le variabili caratteristiche del *generatore di eventi* sono impostate dal *ModelSwarm* e sono le seguenti:

- *eventType:* determina la modalità tramite la quale si desidera generare gli eventi. Le scelte disponibili riguardano la generazione di numeri discreti oppure di numeri reali distribuiti normalmente.
- *eventChangeProbability:* determina la soglia di probabilità sotto la quale il generatore di eventi deve cambiare di stato. Con una probabilità di *0.01* ad esempio, il generatore cambierà di stato mediamente una volta ogni *100 tic*.

4.2.5 Il Book

Il *Book* rappresenta il vero elemento funzionale della simulazione e rappresenta il mezzo tramite il quale la simulazione può essere realizzata. Il *Book* interagisce con il *ModelSwarm* solo per l'impostazione delle proprie variabili ed accoglie tutti gli ordini immessi dai singoli agenti durante le varie fasi della giornata virtuale. Il suo compito è quello di abbinare i contratti in base alle caratteristiche determinate in fase di programmazione. In *SUM* esistono tre momenti della giornata in cui sono svolte azioni diverse: la *fase di preapertura*, la *fase di contrattazione continua* e la *fase di chiusura*. Il *Book*, così come gli altri oggetti appartenenti al modello, si comporta in modo diverso a seconda del momento in cui si trova la simulazione.

4.2.5.1 Fase di preapertura

1. Calcolo del prezzo teorico di apertura

Il modello *SUM*, al momento attuale, non calcola alcun prezzo teorico di apertura. Nonostante possa sembrare un particolare apparentemente trascurabile, rappresenta in realtà un elemento essenziale per garantire una simulazione realistica e completa in ogni suo particolare. Il comportamento che il *book* di *SUM* dovrebbe attuare per la determinazione del prezzo teorico di apertura, sarebbe sintetizzabile in quattro regole sequenziali fissate dalla *Consob*.

Regola I

Il prezzo teorico è quello che consente la negoziazione della maggior quantità di titoli.

Le operazioni necessarie per ottemperare a questa regola sono le seguenti:

1) Per ogni proposta a prezzo limitato viene calcolata la quantità complessiva in denaro e in lettera.
2) Per ogni prezzo viene calcolata la quantità negoziabile ossia la maggiore tra la quantità in denaro e quella in lettera per quel dato livello del *book*.
3) Infine viene rilevata la quantità negoziabile maggiore tra tutte quelle presenti.

Regola II

Il prezzo teorico di apertura è quello tramite il quale è possibile negoziare la quantità che produce la minor differenza fra il volume delle quantità in denaro ed il volume delle quantità in lettera.

Nel caso di più proposte con la stessa quantità, occorre verificare quale livello di prezzo rende minima, in valore assoluto, la differenza tra la quantità in lettera e la quantità in denaro.

Regola III

Il prezzo teorico di apertura corrisponde al prezzo più vicino a quello di chiusura della seduta precedente.

In tal caso è necessario valutare il prezzo che in valore assoluto differisce meno dall'ultimo prezzo di chiusura disponibile.

Regola IV

Il prezzo teorico di apertura corrisponde al maggiore tra quelli equidistanti.

2. Validazione

Affinché il prezzo teorico di apertura sia ritenuto valido, deve rispettare tre vincoli prestabiliti:

- Consentire la negoziazione di una percentuale prefissata della quantità complessivamente presente per il segno prevalente. La percentuale di norma fissata per questo tipo di validazione è pari al *20%*.
- La differenza tra il prezzo teorico di apertura ed il prezzo di riferimento del giorno precedente non deve superare una percentuale prefissata. Di norma queste percentuale è fissata al *10%*.
- La proporzione delle proposte a prezzo di apertura sul totale delle proposte non deve superare un'altra determinata percentuale.

4.2.5.2 Fase di contrattazione continua

Durante la fase di contrattazione continua, il *book* di *SUM* si comporta esattamente come un reale *book* di Borsa. Ad ogni istante della simulazione, l'agente di turno invia al *book* un ordine in acquisto o in vendita ad un determinato prezzo fissato in base a regole tipiche della categoria di appartenenza. Il *book* agisce di conseguenza con la seguente procedura:

1) L'ordine ricevuto è inserito nella lista di appartenenza (*denaro* per l'acquisto, *lettera* per la vendita) pronto per essere valutato.
2) Se tra gli ordini già presenti nel *book* ne esiste almeno uno compatibile con quello appena inserito, il *book* li abbina automaticamente eseguendo il contrattato ed aggiornando la lista.
3) Il prezzo di esecuzione modifica il prezzo corrente di negoziazione il quale risulterà aggiornato in base alle nuove condizioni.

L'esecuzione di queste operazioni avviene in modo del tutto automatico tramite l'implementazione di alcune semplici condizioni *if* all'interno dell'oggetto *Book*.

4.2.5.3 Fase di chiusura

Al termine della giornata virtuale, dopo che tutti gli agenti della popolazione hanno operato, sono aggiornati il prezzo medio della giornata e la ricchezza detenuta dai singoli agenti. Un'osservazione di rilievo deve essere effettuata sulle attuali caratteristiche del modello *SUM*. Le attuali procedure di chiusura attuate dal modello non risultano infatti conformi all'attuale regolamento vigente. In base all'ultimo regolamento sulla Borsa emanato dalla *Consob*, la fase di chiusura prevede un'opportuna asta che, sebbene non sia analizzata in questa sede, va certamente considerata di grande rilevanza ai fini di una corretta simulazione.

4.2.6 Gli Agenti

Finora si sono analizzate le parti principali del modello illustrando come interagiscono tra di loro e quali sono le funzioni a loro attribuite. Si è analizzato il funzionamento dell'*ObserverSwarm*, del *ModelSwarm*, dell'*EventsGenerator* e del *Book*, ma non sono ancora stati presi in considerazione gli elementi determinanti del modello: gli agenti.

SUM dispone di diverse tipologie di agenti, ciascuna dotata di caratteristiche e comportamenti propri. Ogni categoria di agente fa capo ad un oggetto contenente l'insieme di regole che determinano la scelta delle proprie azioni. Questi oggetti prendono il nome di *ruleMaster*. I *ruleMaster* contengono l'insieme di regole comportamentali che caratterizzano i singoli agenti durante la simulazione. Così come avviene per altri oggetti appartenenti al modello, anche i *ruleMaster* associano diverse tipologie di azioni e comportamenti a ciascuna fase della simulazione (*apertura*, *continua* e *chiusura*). Sia gli agenti che i *ruleMaster* utilizzano due classi di base che prendono rispettivamente il nome di *BasicSumAgent* e *BasicSumRuleMaster*. Queste due classi contengono gli elementi comuni a tutti gli agenti e a tutti i *ruleMaster*. Al loro interno si trovano, oltre ad alcune variabili, anche i metodi che descrivono le azioni da svolgere durante le tre fasi della giornata.

- All'interno del ***BasicSumAgent*** sono contenuti i seguenti metodi:
  - *setNumber:* imposta il numero di agenti inseriti nel modello.
  - *setBook:* imposta l'indirizzo in memoria del *Book*.
  - *setEventGenerator:* imposta l'indirizzo in memoria del *Generatore di eventi*.

- *setAsymmetricBuySellProb:* imposta lo scostamento simmetrico della probabilità di acquisto e vendita per gli agenti casuali.
- *setMaxOrderQuantity:* imposta il massimo numero di ordini inseribili da ciascun agente.
- *act0*: azione da compiersi durante la fase di apertura.
- *act1*: azione da compiersi durante la fase di continua.
- *act2*: azione da compiersi durante la fase di chiusura.

- Per quanto riguarda il ***BasicSumRuleMaster***, i metodi implementati sono i seguenti:
  - *setAgentProbToActBeforeOpening:* imposta la probabilità che un agente casuale agisca in fase di apertura.
  - *setMinCorrectingCoeff / setMaxCorrectingCoeff:* imposta i coefficienti di correzione per gli agenti con caratteristiche casuali.
  - *setAsymmetricRange:* imposta il range di asimmetria per gli agenti con caratteristiche casuale.
  - *setEventSensibility:* imposta la sensibilità degli agenti sensibili agli eventi.
  - *setFloorP:* imposta la probabilità di agire se il prezzo è sceso sotto un certo valore.

Oltre queste caratteristiche comuni, ogni classe è caratterizzata da altri elementi propri di uso esclusivo.

4.2.6.1 Agenti Casuali *(RandomAgent)*

Gli agenti casuali sono la tipologia di agenti più semplice presente nel modello. Per motivi che ora verranno illustrati, rappresentano l'elemento maggiormente stabilizzante di tutto il modello. Il funzionamento degli agenti casuali è relativamente semplice ed utilizza un proprio *ruleMaster* denominato *randomRuleMaster*.

1. Fase di preapertura

Durante la fase di preapertura, gli agenti decidono casualmente se agire oppure no. La scelta è determinata dal confronto tra un numero casuale ed il valore contenuto all'interno della variabile *agentProbToActBeforeOpening*. Se il numero casuale è inferiore a quello prestabilito l'agente opera, in caso contrario agirà durante la fase di

contrattazione continua. Nel caso l'agente intenda operare, si procederà con la determinazione del tipo di azione (acquisto o vendita) e del prezzo. L'agente che opta per l'azione in fase di preapertura, non potrà più agire durante la fase di contrattazione continua. Questa regola vale per tutte le tipologie di agenti presenti nel modello.

2. Fase di contrattazione continua

Durante la fase di contrattazione continua, gli agenti che non hanno operato durante la fase di apertura, immettono i propri ordini non appena sono interrogati. Il prezzo è determinato utilizzando come riferimento l'ultimo prezzo inserito nel *book*. Ottenuto l'ultimo prezzo inserito nel *book*, il *ruleMaster* calcola il nuovo prezzo moltiplicando l'ultimo prezzo per un fattore moltiplicativo *(1+x)* determinato casualmente in modo proporzionale ai coefficienti correttivi analizzati durante l'analisi del *basicSumRuleMaster* e del *basicSumAgent*. Dopo aver determinato il prezzo, un'altra scelta casuale determina il tipo di ordine da immettere. Occorrerà infatti scegliere se immettere un ordine di acquisto oppure di vendita.

Questa tipologia di agenti, grazie al suo comportamento prevalentemente casuale, rappresenta un elemento decisamente stabilizzante per il modello. La generazione di un numero casuale è generalmente distribuita con legge uniforme. Grazie a questa caratteristica, gli agenti casuali, nel prendere le proprie decisioni di prezzo, si distribuiranno uniformemente attorno al valore centrale di riferimento. Se il numero di agenti che segue questa regola è sufficientemente elevato, il prezzo non dovrebbe mai discostarsi troppo da quello iniziale o perlomeno dovrebbe mostrare una varianza relativamente bassa. Gli agenti casuali rappresentano una categoria di agenti molto importante che necessita un'attenta valutazione in fase di presimulazione. Una scelta errata del loro numero può condurre a notevoli alterazioni dei risultati derivanti dalla simulazione. Una simulazione con bolle o crash di dimensioni eccessive perde di realismo e soprattutto di utilità. Condizioni troppo distorte ed alterate rispetto alla realtà che si intende studiare, rappresentano una notevole divergenza da quelle che sono le finalità orientate alla ricerca.

4.2.6.2 Agenti Limitatori di Perdite *(StopLossAgent)*

Lo scopo degli agenti limitatori di perdite riguarda, come suggerisce la definizione stessa, la limitazione delle possibili perdite finanziarie. Nonostante possa sembrare

una categoria di agenti alquanto inusuale da analizzare, è sufficiente osservare alcuni comportamenti attuati in un'ordinaria giornata di Borsa per comprendere che la loro presenza nei mercati reali è decisamente maggiore di quanto si possa immaginare. Il comportamento di questi agenti è simile a quello degli agenti casuali e le loro specifiche sono contenute nel rispettivo *ruleMaster*.

1. Fase di preapertura

Durante la fase di preapertura, il comportamento degli agenti limitatori di perdite è completamente analogo a quello degli agenti casuali. Gli agenti decidono casualmente se agire oppure no in base al valore ottenuto da un generatore di numeri casuali. Se tale valore è inferiore a quello fissato nella variabile *agentProbToActBeforeOpening*, l'agente opera in apertura altrimenti agirà nella fase di contrattazione continua. Nel caso intenda agire, si procederà alla relativa determinazione del prezzo nonché dell'azione da intraprendere. L'agente limitatore di perdite che agisce in apertura, sarà escluso dall'azione durante la fase di contrattazione continua.

2. Fase di contrattazione continua

In questa fase della simulazione, il comportamento degli agenti limitatori è determinato dalla valutazione di una perdita consentita. Se le condizioni di perdita sono rispettate, l'agente si comporterà come un agente casuale, in caso contrario adotterà un comportamento difensivo tipico della propria categoria di appartenenza. Il comportamento di un agente limitatore può essere così riassunto:

a) **Vincoli di perdita rispettati:** Se tutti i vincoli di perdita sono rispettati, il prezzo è determinato in maniera completamente casuale con lo stesso algoritmo già adottato in precedenza per gli agenti casuali. Tramite lo stesso algoritmo si procede anche alla determinazione della tipologia di azione da intraprendere (*acquisto* o *vendita*).

b) **Vincoli di perdita non rispettati:** Se i vincoli di perdita non sono rispettati, l'agente opera in modo da ridurre le proprie perdite e massimizzare i propri profitti. Queste finalità possono essere perseguite secondo due modalità:
   - *Senza controllo su posizioni lunghe o corte*
   - *Con controllo su posizioni lunghe o corte.*

- Tramite la ***prima modalità***, l'agente limitatore acquisterà all'ultimo prezzo inserito nel *book* tutte le volte che questo risulterà maggiore di quello medio moltiplicato per un fattore di capitalizzazione determinato in fase di presimulazione. Venderà all'ultimo prezzo inserito nel *book* tutte le volte che questo risulterà minore di quello medio moltiplicato per un fattore di sconto anche questo determinato in fase di presimulazione.

- Tramite la ***seconda modalità***, l'agente si comporta nello stesso modo ma, nella valutazione dell'azione, considera anche la propria posizione riguardo la quantità di titoli condivisa il giorno precedente. L'agente acquisterà all'ultimo prezzo inserito nel *book* se questo sarà maggiore di quello medio moltiplicato per un fattore di capitalizzazione, a patto che la sua posizione sia *corta* ossia la quantità condivisa di titoli sia inferiore a zero. Il ragionamento è analogo per il caso opposto. L'agente venderà all'ultimo prezzo inserito nel *book* se questo sarà minore di quello medio moltiplicato per un fattore di sconto, a patto che la sua posizione sia *lunga* ossia la quantità condivisa di titoli sia maggiore di zero.

La scelta tra le due modalità è devoluta completamente all'utilizzatore e deve essere accuratamente intrapresa in base alle finalità che si desidera perseguire tramite la simulazione. In condizioni normali, all'avvio di ogni simulazione, la variabile *checkingIfShortOrLong* che controlla le due modalità è di norma fissata ad uno.

4.2.6.3 Agenti Imitatori *(MarketImitatingAgent e LocallyImitatingAgent)*

Gli agenti imitatori agiscono imitando qualcuno o qualcosa. Non si è specificato ulteriormente l'oggetto di imitazione in quanto, nel modello *SUM*, esistono due tipologie di agenti imitatori: ***agenti imitatori del mercato*** ed ***agenti imitatori locali***. Nonostante la diversa denominazione attribuita agli agenti, il loro comportamento diverge pochissimo. La differenza tra le due categorie risiede nella diversa asimmetria adottata durante la scelta dell'azione da intraprendere (*acquisto* o *vendita*). Entrambe le categorie di agenti imitatori utilizzano come *ruleMaster* il *randomRuleMaster* e, per maggiore chiarezza espositiva, saranno analizzate separatamente.

1. Fase di preapertura

Durante la fase di preapertura, il comportamento assunto da questa categoria di agenti è del tutto analogo al comportamento già esaminato relativo alle categorie già trattate. La scelta di agire in fase di preapertura è sempre determinata casualmente mediante l'algoritmo già illustrato e colui che agisce in preapertura, esclude la propria azione in fase di contrattazione continua.

2. Fase di contrattazione continua

Durante la fase di contrattazione continua, il comportamento attuato dalle due tipologie di agenti imitatori si differenzia per il modo con cui è determinata l'azione da eseguire ossia la scelta tra acquisto e vendita. Entrambe le categorie apportano questa modifica comportamentale tramite la modifica della variabile *buySellSwitch*. La differenza tra le due categorie risiede però nel modo con cui questa modifica è attuata.

2.a MarketImitatingAgent

Gli agenti imitatori del mercato determinano il fattore di asimmetria tra acquisto e vendita eseguendo un'analisi sul prezzo medio di contrattazione degli ultimi due giorni di seduta. Se i prezzo medio è aumentato, la scelta tra acquisto e vendita sarà fortemente sbilanciata verso l'acquisto. In formule:

$$buySellSwitch = asymmetricBuySellProb$$

Se il prezzo medio è diminuito, il ragionamento sarà speculare e l'asimmetria introdotta favorirà maggiormente la vendita piuttosto che l'acquisto. In formule risulta del tutto analogo:

$$buySellSwitch = 1 - asymmetricBuySellProb$$

Le decisioni di acquisto o vendita sono quindi governate dal grado di asimmetria che dipende a sua volta dall'andamento del prezzo di mercato. Questa logica di funzionamento è abbastanza coerente con quanto realmente accade nei mercati ogni giorno. E' legittimo pensare di vendere i propri titoli se per due giorni consecutivi il prezzo medio di negoziazione è diminuito; tutto lascia pensare ad un breve periodo di

crisi che probabilmente trascinerà il mercato verso il basso introducendo quel fenomeno che in termini economici assume il nome di previsioni auto avveranti.

2.b LocallyImitatingAgent

Gli agenti imitatori locali, a differenza di quanto accade per gli agenti imitatori del mercato, determinano il fattore di asimmetria tra acquisto e vendita mediante un algoritmo che, anziché prendere in considerazione l'andamento del prezzo medio delle giornate precedenti, prende in considerazione il tipo di azioni intraprese dagli altri agenti. Per eseguire questa operazione è sufficiente valutare l'istanza *getLocallyHistory* contenuta all'interno del *book*. In tal modo, l'agente è in grado di conoscere quali sono le maggiori tipologie di ordini immesse nel *book* (*acquisto* o *vendita*). Se nello storico del *book* sono presenti più ordini di acquisto rispetto a quelli di vendita, l'agente varierà il suo grado di asimmetria rappresentato dalla variabile *buySellSwitch* in modo da favorire statisticamente gli acquisti. Se nello storico del *book* sono presenti più ordini di vendita rispetto a quelli di acquisto, l'agente muterà il proprio grado di asimmetria in modo da favorire statisticamente le vendite.

4.2.6.4 Agenti sensibili agli eventi *(EventAgent)*

Una categoria di agenti che, per ovvi motivi, merita particolare attenzione, è certamente quella degli agenti sensibili agli eventi, argomento di questa tesi. Gli agenti sensibili agli eventi rappresentano una tipologia di agenti che sta assumendo una maggiore rilevanza all'interno della compagine sociale operante sui mercati finanziari. Nel modello *SUM*, gli agenti sensibili agli eventi determinano il proprio comportamento in base ad alcune regole introdotte nell'*eventRuleMaster* e le loro azioni riflettono l'andamento degli eventi. Durante la stesura del codice, si è cercato di strutturare un agente che, benché artificiale, si comportasse comunque in un modo molto simile a quello adottato da un vero operatore finanziario. Nonostante il comportamento utilizzato implichi una struttura prevalentemente matematica, il modello rispecchia abbastanza bene il reale comportamento dei soggetti appartenenti a questa categoria operanti sui mercati.

1. Fase di preapertura

Durante la fase di preapertura, gli agenti possono comportarsi in base a due modalità diverse. La variabile che determina le modalità di comportamento è denominata *eventAgentChoiceToActBeforeOpening*:

- La ***prima*** modalità prevede che la decisione di operare dipenda esclusivamente dallo stato dell'evento in corso. Gli agenti interrogano il generatore di eventi per sapere lo stato attuale. Se il generatore di eventi fornisce in risposta un evento di qualsiasi entità diverso da quello nullo, gli agenti operano e procedono alla determinazione del prezzo. Dal momento che il *generatore di eventi* determina l'evento una sola volta per ciascun *tic*, e la fase di apertura dura un solo *tic* in cui possono agire contemporaneamente tutti gli agenti della popolazione, appare intuitivo che, nel caso esista un evento non nullo, tutti gli agenti appartenenti a questa categoria opereranno assieme escludendosi dalla contrattazione continua che avverrà in seguito.

- La ***seconda*** modalità prevede che la decisione di operare in fase di preapertura sia proporzionale all'entità dell'evento. Gli agenti hanno facoltà di scegliere se operare in fase di preapertura oppure attendere la fase di contrattazione continua, ma la probabilità con cui si determina la scelta è direttamente proporzionale all'entità dell'evento. Maggiore è l'entità dell'evento e maggiore sarà la probabilità che un agente operi in fase di preapertura, minore è l'entità dell'evento e minore sarà la probabilità che un agente operi in fase di preapertura. Se l'evento è nullo, l'agente sensibile alle notizie si comporterà in questa fase di negoziazione esattamente come un agente casuale. La scelta di utilizzare questa modalità deriva dalla necessità di fornire al modello maggiore coerenza con la realtà che si intende riprodurre. Un evento molto rilevante, positivo o negativo che sia, è in grado di incentivare maggiormente la necessità di operare in fase di preapertura.

2. Fase di contrattazione continua

Durante la fase di contrattazione continua, se gli agenti non hanno già operato in fase di preapertura, sarà loro chiesto di agire. Dovranno prendere una decisione sull'azione da intraprendere (*acquisto* o *vendita*) e a quale prezzo. Il loro comportamento

dipenderà esclusivamente dall'entità dell'evento esistente in quel particolare istante. Si possono distinguere tre casi:

a) **Evento nullo:** Se l'evento è pari a zero, l'agente opera esattamente come un agente casuale e la sua scelta di acquisto o vendita, con relativo prezzo, è determinata con le stesse modalità già enunciate durante l'illustrazione degli agenti casuali.

b) **Evento positivo:** Se l'evento è positivo, l'agente opera in acquisto ed il prezzo che è fissato risulta proporzionale all'entità numerica dell'evento. Su questo punto è necessario fornire ulteriori spiegazioni. Così come avviene per le altre tipologie di agenti, anche in questo caso, il prezzo di riferimento per la valutazione del nuovo prezzo, è l'ultimo inserito nel *book*. Il nuovo prezzo inserito dall'agente sarà calcolato tramite una maggiorazione casuale proporzionale all'entità dell'evento ed alla sensibilità degli agenti applicata sull'ultimo prezzo inserito. In formule:

   Nuovo prezzo = Ultimo prezzo x (1 + (Numero casuale [0, Evento x Sensibilità])

   Il prezzo immesso da ogni singolo agente, benché casuale, sarà direttamente proporzionale all'entità dell'evento ed alla sensibilità degli agenti. Maggiore sarà la positività dell'evento e maggiore sarà la disponibilità ad aumentare il prezzo. Gli andamenti azionari di tutti i giorni sono un'evidente conferma di questo abituale comportamento.

c) **Evento negativo:** Se l'evento è negativo, l'agente agisce esattamente in modo speculare. Il prezzo adottato come riferimento è sempre l'ultimo inserito nel *book*, ma il comportamento dell'agente non implica più un acquisto bensì una vendita. Il calcolo del prezzo avviene in maniera del tutto analoga alla precedente con l'unica eccezione di essere determinato tramite una minorazione casuale sempre proporzionale all'entità dell'evento ed alla sensibilità degli agenti applicata sull'ultimo prezzo inserito. In formule:

   Nuovo prezzo = Ultimo prezzo x (1 + (Numero casuale [Evento x Sensibilità, 0])

   In questo caso la formula vede il range del numero casuale invertito. Questo è dovuto al semplice fatto che l'evento negativo renderà il prodotto con la

sensibilità altrettanto negativo rendendolo così un limite inferiore. L'analogia con il contesto reale si fonda sullo stesso ragionamento e perviene alle stesse conclusioni: maggiore sarà la negatività degli eventi e maggiore sarà l'incentivo ad abbassare il prezzo.

4.2.6.5 Agenti previsori *(forecastingAgent e ANNForecastingAppAgent)*

Il funzionamento degli agenti previsori è molto complesso e merita particolare attenzione. Innanzitutto occorre precisare che nel modello esistono due tipologie di agenti che possono essere definiti previsori. Una rappresenta gli agenti veri e propri, l'altra è costituita da un agente virtuale che non opera direttamente sul mercato ma fornisce esclusivamente previsioni. Data la complessità dell'argomento è bene analizzarli separatamente.

1. ForecastingAgent

Il *forecastingAgent* non è un vero agente e, nonostante si trovi nel modello, non agisce direttamente sul mercato. Questo agente è costituito da una rete neurale artificiale e produce ogni giorno una previsione sul possibile prezzo assunto del titolo nei giorni successivi. La lunghezza della previsione è indicata all'interno della variabile *nAheadForecasting* e rappresenta il numero di giorni futuri per cui si richiede la previsione. Di norma è fissata a dieci giorni ma l'utilizzatore può modificarla a proprio piacimento. L'agente previsore, per svolgere la propria funzione, utilizza una rete neurale artificiale nella quale i nodi di input sono fissati in base alla dimensione della finestra e sono contenuti nella variabile *dataWindowLength* che di norma è fissata a *30*. I nodi di output sono pari ad *1* mentre quelli nascosti sono determinati automaticamente dal sistema come la metà dei nodi di input. Il numero di sequenze di dati (*pattern*) in input è fissato a *100*. La lunghezza di una singola epoca è contenuta nella variabile *forecastingTrainingSetLength* mentre il numero di epoche utilizzate per l'apprendimento è contenuto in *epochNumberInEachForecastingTrainingCycle*; anche questa di norma fissata pari a *100*. Il numero di cicli totali, in condizioni predefinite, sarà pertanto pari a *10.000*; al termine dei quali il processo salverà il valore ottenuto in output e, dopo aver azzerato l'intera matrice dei pesi (definita anche matrice di apprendimento), la rete neurale sarà nuovamente pronta per una nuova previsione.

2. ANNForecastingAppAgent

Il vero agente in grado di poter usufruire delle previsioni effettuate dal *forecastingAgent* è l'*ANNForecastingAgent*. Questo agente opera sul mercato seguendo le previsioni fornite dal *forecastingAgent*. Il suo funzionamento è del analogo a quello degli agenti imitatori ed il *ruleMaster* di riferimento è ancora una volta il *randomRuleMaster*.

2.a Fase di preapertura

La fase di preapertura è identica a quella già analizzata per gli altri agenti eccetto quelli sensibili agli eventi che invece sono caratterizzati da un processo decisivo a proprio. La scelta di operare è determinata secondo le stesse modalità già adottate per gli agenti casuali e come avviene per tutte le altre categorie di agenti, colui che opera in preapertura è inibito dall'operare in continua.

2.b Fase di contrattazione continua

Durante la fase di contrattazione continua, gli agenti che non hanno agito in fase di preapertura, saranno chiamati ad agire. Il processo di decisione sull'azione da intraprendere è molto simile a quello già analizzato per gli agenti imitatori e si basa nuovamente su una variazione introdotta sul coefficiente di asimmetria. La determinazione del coefficiente si fonda sul confronto tra la previsione effettuata dal *forecastingAgent* ed un parametro di riferimento denominato *ANNInactivityRange*. Tale parametro esprime un range di inattività oltre il quale, l'agente che segue le previsioni, opera in base al proprio comportamento caratteristico:
Se la previsione è *superiore* al range, si prospettano possibilità di guadagno ed il coefficiente di asimmetria è sbilanciato a favore degli acquisti.
Se la previsione è *inferiore* al range, si prospettano possibilità di perdita ed il coefficiente di asimmetria è sbilanciato a favore delle vendite.

4.2.6.6 Agenti Cognitivi *(BPCTagent)*

L'ultima categoria di agenti che occorre prendere in considerazione è quella relativa agli *agenti cognitivi*. Questi agenti basano il proprio funzionamento su reti neurali artificiali implementate con il metodo dei *cross target* ed utilizzano delle previsioni *"ragionate"* per determinare le proprie azioni. Un agente cognitivo, operando in un

ambiente economico, deve sviluppare ed adattare le proprie capacità di valutazione in maniera coerente. Deve apprendere cosa fare per ottenere uno specifico risultato e cosa dedurre dalle conseguenze delle proprie azioni. La stessa regola deve ovviamente valere anche nel caso in cui l'agente si trovi in un ambiente nel quale interagisce con altri agenti non necessariamente della stessa tipologia. Oltre alla coerenza interna, gli agenti possono anche sviluppare altre caratteristiche come la capacità di intraprendere decisioni e valutazioni di effetti suggeriti dall'ambiente stesso o da altri agenti. Il scopo principale di questi agenti riguarda lo sviluppo di un'adeguata coerenza interna tra quale azione intraprendere e le relative conseguenze derivanti dall'azione stessa. L'agente deve produrre delle congetture sulle proprie azioni e sui relativi effetti in base alle informazioni disponibili. I target utilizzati nel processo di apprendimento sono: da un lato, gli effetti derivanti delle azioni intraprese dall'agente, dall'altro lato, le azioni necessarie per ottenere gli effetti congetturati.

All'interno del modello *SUM* sono disponibili due tipologie di agenti cognitivi: una tipologia definita *A* ed una tipologia definita *B*. In questa sede sarà analizzata soltanto la tipologia di agenti *B* dal momento che è considerata aggiornamento della tipologia *A*. Ciò non esclude comunque che gli agenti delle due categorie possano tranquillamente coesistere all'interno della stessa simulazione. Gli agenti di tipo *B* sono dotati di *8* nodi di input: il *prezzo medio dei 5 giorni antecedenti* quello in corso (da t_{-5} a t_{-1} considerando che la simulazione si trova in *t*), la *liquidità* dell'agente, il *valore condiviso* al termine del giorno t_{-1} nonché il *prezzo stimato* del giorno *t+n-1* dove *n* rappresenta il numero di giorni per i quali si richiede la previsione. Il numero di nodi nascosti è fissato pari a *6*. Ci sono *4* nodi di output dal lato degli *effetti* ed *1* nodo di output dal lato delle *azioni*. Gli *effetti* sono la *liquidità*, la *quantità condivisa*, la *ricchezza determinata dall'ultimo prezzo giornaliero* e la *ricchezza determinata considerando il prezzo previsto*. L'azione in output prevede solo due alternative: *acquisto* o *vendita*. La decisione tra acquisto e vendita è rappresentata da un numero reale all'interno dell'intervallo *–maxOrderQuantity* e *+maxOrderQuantity.* Se la variabile *bs* risulta maggiore di zero, l'agente si comporta da acquirente mentre se *bs* è minore di zero l'agente si comporta da venditore. Nel caso *bs* fosse pari a zero, caso estremamente raro, l'agente non effettuerà alcuna operazione. La quantità che deve essere acquistata o venduta (in senso unitario) è determinata dall'approssimazione del valore assoluto attribuito alla variabile *bs*. La rete neurale che gestisce questa tipologia di agenti prevede che possano anche avere due obiettivi esterni. Le situazioni che si possono presentare sono sostanzialmente quattro:

EO=0: Nessun obiettivo esterno.

EO=1: Obiettivo esterno di aumentare la propria ricchezza utilizzando come valutazione l'ultimo prezzo inserito.
EO=2: Obiettivo esterno di aumentare la propria ricchezza utilizzando come valutazione il prezzo previsto.
EO=3: Entrambi gli obiettivi esterni.

L'utilizzo di obiettivi esterni è molto utile se si desidera osservare il comportamento degli agenti nel caso sia chiesto loro di incrementare la propria ricchezza.

Capitolo 5
Esperimenti ed analisi dei risultati

La parte più originale di tutto il lavoro è costituita dagli esperimenti condotti e dall'analisi dei risultati emersi. In primo luogo saranno analizzati gli obiettivi che si sono voluti perseguire tramite questa ricerca. In seguito saranno analizzati i parametri che in questo contesto sono ritenuti di maggior rilevanza, giustificando anche i motivi che hanno condotto a tali preferenze. Per ultimo saranno analizzate le singole fasi degli esperimenti al cui interno saranno illustrati i diversi casi presi in esame.

5.1 Obiettivi degli esperimenti

Definiamo gli obiettivi che si sono voluti perseguire tramite gli esperimenti svolti. Sono fondamentalmente quattro:

1) ***La presenza di agenti sensibili agli eventi altera il mercato borsistico?***
La presenza di agenti sensibili agli eventi all'interno di un mercato borsistico è in grado di alterare sostanzialmente i risultati? Gli effetti che emergono in un mercato operante in simili condizioni, derivano solamente da un comportamento comune adottato da tutti gli agenti attivi o dipendono strettamente dalla presenza di agenti particolarmente sensibili alle notizie? Tramite alcuni esperimenti condotti in un ambiente simulato, è possibile comprendere se la presenza di alcuni, anche pochi, agenti appartenenti a questa tipologia, sia sufficiente per destabilizzare l'intero mercato.

2) ***Esiste un numero di tali agenti in grado di alterare il mercato?***
Nel caso si accerti che la presenza di tali agenti sia in grado di alterare significativamente le condizioni di stabilità mostrate dal mercato, è possibile individuare una percentuale di questi agenti considerata critica oltre la quale il mercato risulterà significativamente alterato? Questa informazione risulta di notevole rilevanza e rappresenta lo scopo principale di questa tesi.

3) ***Che influenza hanno le altre categorie di agenti in un simile mercato?***
Individuata la soglia massima di tali agenti per cui la stabilità dell'intero mercato ne risulta fortemente compromessa, quale influenza avranno le altre categorie di

agenti introdotte all'interno di un simile mercato? L'effetto potrebbe essere tanto enfatizzante quanto riduttivo. Tramite la simulazione si cercherà di attribuire ad ogni tipologia di agente il proprio grado di influenza su un mercato così strutturato.

4) Quale incidenza hanno i parametri di rilievo sui risultati ottenibili?

Analizzati gli aspetti principali riguardanti direttamente la tipologia di popolazione introdotta all'interno del modello, l'ultimo obiettivo che tramite queste simulazioni si desidera perseguire riguarda l'incidenza che i principali elementi di rilievo, elementi che nel successivo paragrafo saranno analizzati in dettaglio, hanno nei confronti del mercato. Riguarderanno principalmente il numero di agenti introdotti nel modello, la sensibilità attribuita agli agenti sensibili agli eventi, la possibilità concessa o meno a questi agenti di agire discrezionalmente in fase di apertura o in fase di contrattazione continua, la tipologia degli eventi e la durata della simulazione.

5.2 Parametri di rilievo

Gli esperimenti sono stati strutturati e condotti in modo da poter perseguire nel miglior modo possibile gli obiettivi appena esposti. La scelta della struttura e dei parametri alla base degli esperimenti rappresenta un elemento determinante da cui dipende fortemente la bontà dei risultati ottenuti. Se non si presta adeguata attenzione al modo in cui saranno condotti gli esperimenti, si corre il rischio di ottenere risultati interessanti che forniscono però informazioni diverse da quelle che il ricercatore si era proposto di ottenere. Quando si progetta una qualsiasi struttura di esperimenti, è innanzitutto necessario assicurarsi che questa consenta di conseguire l'obiettivo prestabilito.

5.2.1 Tipo di evento

Il primo elemento da prendere in considerazione riguarda il tipo di evento. Durante l'esposizione del modello si è già illustrato il funzionamento del generatore di eventi e si è sottolineata la possibilità di scegliere, nella fase precedente alla simulazione, di generare solo eventi interi oppure eventi su scala reale. In questo contesto si è preferito optare per la seconda alternativa in quanto maggiormente coerente con la realtà borsistica sottostante. Gli eventi, in tutte le simulazioni prese in esame, saranno

caratterizzati da numeri reali distribuiti normalmente tra meno infinito e più infinito con valore centrale pari a zero.

5.2.2 Probabilità degli eventi

La probabilità che un evento si verifichi rappresenta un elemento determinante per la bontà dei risultati ottenuti dalla simulazione. Dalla frequenza con cui gli eventi si verificano, possono derivare diversi effetti. La scelta di una probabilità adatta è essenziale per ottenere dei risultati coerenti con la realtà sottostante e questo parametro deve riflettere le caratteristiche dell'ambiente all'interno del quale la simulazione si sviluppa, mantenendo con gli altri elementi del modello la maggior coerenza possibile. Una probabilità troppo elevata o troppo bassa renderebbe il modello troppo astratto e troppo lontano dalla realtà. Un evento che nella realtà cambiasse mediamente ogni dieci minuti oppure ogni mese rappresenterebbe una condizione troppo improbabile da verificarsi. In entrambi i casi si otterrebbero dei risultati distorti la cui utilità risulterebbe pressoché nulla.

La scelta della probabilità relativa al cambio di stato da parte di un evento dipende strettamente dal numero di agenti presenti nel modello. Durante l'esposizione del modello *SUM*, si è detto che la giornata borsistica virtuale dipende dal numero di agenti introdotti all'interno del modello. Concentrando l'attenzione sulla realtà borsistica di ogni giorno, è facile rendersi conto che gli eventi principali hanno periodicità mediamente giornaliera. Esistono eventi o notizie che nel mondo finanziario e sociale permangono per più di una giornata oppure meno di una mattinata, ma rappresentano casi particolari che raramente si verificano. Nella scelta della probabilità si è dunque preferito seguire questa logica e si è fissata una probabilità di cambio stato proporzionale alla durata della giornata. La probabilità di cambio stato è pari al reciproco del numero di agenti presenti nel modello. Nel caso in esame la probabilità è stata fissata pari a *0.003*.

5.2.3 Sensibilità degli agenti

Un altro elemento di estrema importanza nella determinazione dei risultati riguarda la sensibilità che gli agenti hanno nei confronti delle notizie. Il parametro che determina la sensibilità degli agenti deve essere scelto con molta attenzione e da esso dipendono i

risultati di tutte le simulazioni. La scelta della sensibilità da attribuire agli agenti rappresenta un elemento di notevole complessità da determinare. La causa principale si ricollega a quanto affermato durante la trattazione delle simulazioni. Tra le critiche maggiormente diffuse sulle simulazioni, ne esiste una che riguarda il collegamento tra simulazione e realtà. La critica afferma che le simulazioni non sono in grado di riprodurre esattamente la realtà che si desidera rappresentare in quanto non esiste perfetta conoscenza di tutti gli elementi e di tutte le caratteristiche che la compongono. Il problema in questo caso è analogo. Non essendo a perfetta conoscenza della realtà sottostante, non siamo in grado di determinare a priori l'entità di alcuni parametri necessari alla simulazione e, la sensibilità degli agenti, rappresenta proprio uno di questi parametri. Lo studio della sensibilità rappresenterebbe un ottimo argomento per lo sviluppo di una tesi alternativa ma ciò che questa tesi si propone di fare non è l'individuazione di questo parametro, bensì l'individuazione di una percentuale di agenti oltre la quale il mercato subisce dei mutamenti radicali nel proprio comportamento. La scelta di questo parametro è stata dunque effettuata con la stessa metodologia già adottata nella scelta della probabilità: scelta che tiene conto della coerenza tra realtà e simulazione. Dopo una serie di prove, si è ritenuto opportuno fissare questo parametro pari a *0.20* ma, per maggiore completezza espositiva, uno degli esperimenti condotti al termine della trattazione, riguarderà proprio la modifica di questo parametro. Dai risultati emersi si cercherà di comprendere l'influenza che questo ha nei confronti dell'intero mercato.

5.2.4 Facoltà di azione in preapertura

Un parametro che caratterizza gli agenti sensibili agli eventi è la facoltà concessa loro di agire in fase di preapertura. Gli agenti possono, a discrezione dell'utilizzatore esterno, comportarsi secondo due modalità in base agli esperimenti che si desiderano condurre. Una prima modalità prevede che gli agenti appartenenti a questa categoria agiscano obbligatoriamente quando, durante la fase di preapertura, sussiste un evento di entità non nulla: in tal caso gli agenti opereranno tutti contemporaneamente. Una seconda modalità prevede che gli agenti appartenenti a questa categoria abbiano la facoltà di scegliere se operare in fase di preapertura oppure in fase di contrattazione continua anche se in quel momento sussiste un evento di entità non nulla. La scelta del comportamento dipenderà principalmente dall'entità dell'evento. Gli esperimenti condotti durante le fasi di sperimentazione hanno sempre adottato questa seconda modalità ritenuta più coerente con il reale funzionamento di borsa ma uno degli

esperimenti appartenenti all'ultima fase della trattazione ha utilizzato la prima modalità allo scopo di individuare eventuali correlazioni tra prezzo e fase di apertura.

5.2.5 Popolazione di agenti

L'ultimo elemento fondamentale che occorre prendere in considerazione è la popolazione di agenti utilizzata negli esperimenti. Quando si parla di popolazione occorre fare una netta distinzione tra popolazione in termini quantitativi e popolazione in termini qualitativi. Con il primo termine si indica il numero di agenti introdotti nel modello, con il secondo il tipo di agenti utilizzati.

5.2.5.1 Numero di agenti

La primo parametro che occorre determinare prima di iniziare gli esperimenti riguarda il numero di agenti che si desidera interagiscano nel modello. Il numero di agenti introdotti all'interno di un modello non può essere troppo basso altrimenti il mercato non dispone di una sufficiente liquidità ma neanche troppo elevato altrimenti, date le potenzialità degli strumenti a nostra disposizione, si corre il rischio di rallentare o addirittura bloccare le macchine su cui la simulazione è mandata in esecuzione. Il numero di agenti da introdurre nel modello deve essere scelto in modo da evitare entrambi i problemi ed allo stesso tempo consentire una corretta ripartizione tra le categorie di agenti introdotte all'interno di ciascuna simulazione. Per tali motivi, la popolazione dovrebbe sempre essere fissata in quantità superiore a *100*. Nel caso in esame, per garantire una sufficiente liquidità al mercato, gli esperimenti sono stati condotti con una popolazione di *300* agenti ma, nell'ultima fase di sperimentazione, uno degli esperimenti sarà condotto portando il numero di agenti presenti all'interno del modello pari a *600*. Eventuali differenze rispetto al caso base rappresenteranno un'interessante scoperta.

5.2.5.2 Tipi di agenti

Il secondo elemento di fondamentale importanza nella determinazione dei risultati riguarda la composizione della popolazione introdotta all'interno della simulazione. La determinazione della popolazione, da un punto di vista qualitativo, rappresenta un

elemento fondamentale per tutta la simulazione: da essa dipende direttamente la natura dei risultati.

Prima di eseguire gli esperimenti è necessario innanzitutto prestare molta attenzione alla realtà che si intende studiare e, dopo averne determinato in modo sufficientemente preciso la struttura di base, occorre riprodurre all'interno del modello le stesse condizioni riscontrate nella realtà; nel caso questo non sia possibile, si cercherà di riprodurle nel modo più fedele possibile. La scelta sbagliata degli agenti può condurre a risultati non desiderati o completamente inutili. Nel caso in esame, tutti gli esperimenti sono stati condotti dividendo la popolazione in due categorie: una categoria base alla quale appartengono tutti gli agenti che non sono generalmente considerati trainanti per il mercato, ed una categoria alla quale appartengono le classi di agenti considerate trainanti per il mercato. Alla prima categoria sono stati attribuiti gli agenti casuali (*randomAgent*) e gli agenti limitatori di perdite (*stopLossAgent*). Alla seconda categoria appartengono tutte le tipologie di agenti residuali (*forecastingAgent, eventAgent, BPCTAgent, marketImitatingAgent* e *locallyImitatingAgent*). Una critica potrebbe essere sollevata riguardo l'appartenenza degli agenti limitatori di perdite alla categoria degli agenti meno trainanti nei confronti del mercato. Nonostante tali agenti possano operare tanto in acquisto quanto in vendita, alterando così il mercato in maniera anche significativa, i loro interventi sono attuati solo quando il mercato sia già stato sufficientemente destabilizzato da altre categorie di agenti caratterizzate da maggiori capacità trainanti.

5.2.6 Durata della simulazione

Un elemento di estrema rilevanza riguarda la durata delle simulazioni. Il problema della durata si ricollega a quello già enunciato riguardo la probabilità degli eventi e la sensibilità degli agenti. Una durata troppo breve può fornire dei risultati distorti mentre una durata eccessiva può condurre a risultati corretti ma che hanno richiesto molto più tempo di quanto ne sarebbe stato necessario. Uno dei modi più pratici per risolvere il problema è quello di osservare attentamente la realtà e fissare i parametri in base agli obiettivi che si intendono perseguire. Quando occorre determinare la durata di una simulazione, occorre innanzitutto individuare la periodicità con cui il fenomeno che si desidera studiare si ripete. Avendo a disposizione la periodicità con cui i fenomeni si ripetono all'interno di un ambiente, è possibile determinare con una discreta precisione l'arco temporale entro il quale si verificano la maggior parte dei fenomeni verificabili.

Da alcune indagini emerge che anche i mercati azionari sono dotati di una periodicità avente dimensioni temporali stimabili approssimativamente attorno alla decina di anni. Con l'intento di non prolungare troppo la simulazione, si è ritenuto opportuno adottare una soluzione di compromesso utilizzando per tutti gli esperimenti una durata che pare adeguata di *8* anni virtuali.

5.3 Esperimenti ed analisi dei risultati

Gli esperimenti relativi alla simulazione in questione, sono stati condotti seguendo tre fasi ben distinte:

- La ***prima fase*** di esperimenti riguarda il perseguimento degli obiettivi fissati al punto uno e due. Si desidera innanzitutto individuare gli effetti che alcuni agenti sensibili alle notizie introducono in un mercato azionario. In seguito si desidera individuare la percentuale di agenti appartenenti a questa categoria considerata critica nei confronti della quale il mercato subisce rilevanti alterazioni.

- La ***seconda fase*** di esperimenti riguarda l'identificazione degli effetti che le altre categorie di agenti trainanti hanno su un mercato in cui siano già presenti agenti sensibili agli eventi. In questo modo si persegue al terzo obiettivo prefissato.

- La ***terza fase*** di esperimenti concerne l'individuazione degli effetti derivanti dalla modifica di alcuni parametri rilevanti per la simulazione. Questa fase di sperimentazione è considerata complementare e permette di raggiungere il quarto ed ultimo obiettivo prefissato.

5.3.1 Fase I

La prima fase degli esperimenti riguarda il perseguimento dei primi due obiettivi. Gli agenti sensibili agli eventi sono veramente in grado di influenzare il mercato e soprattutto, qual è la percentuale di agenti appartenente a questa categoria oltre la quale il mercato risulta decisamente destabilizzato? Gli esperimenti condotti in questa fase della sperimentazione cercano di fornire una risposta a queste domande ed hanno utilizzato le seguenti configurazioni:

Esperimento 1

Agenti Totali : *300*
Agenti Casuali : *280*
Agenti Limitatori : *20*

Esperimento 2a	*Esperimento 2b*	*Esperimento 2c*
Agenti Totali : *300*	Agenti Totali : *300*	Agenti Totali : *300*
Agenti Casuali : *274*	Agenti Casuali : *268*	Agenti Casuali : *265*
Agenti Limitatori : *20*	Agenti Limitatori : *20*	Agenti Limitatori : *20*
Agenti Eventi : *6*	Agenti Eventi : *12*	Agenti Eventi : *15*

L'esperimento *1* utilizza solamente la popolazione appartenente alla classe poco trainante ed è utilizzato come esperimento campione. Gli esperimenti *2* introducono gli agenti sensibili alle notizie. Lo scopo di queste simulazioni è rivolto al secondo obiettivo formulato e riguarda l'individuazione della soglia critica di agenti sensibili agli eventi.

5.3.1.1 Esperimento con soli agenti non trainanti (*Esperimento 1*)

Il primo esperimento riguarda l'effetto derivante esclusivamente dall'interazione di una popolazione tipicamente non trainante costituita da agenti casuali ed agenti limitatori di perdite. Al termine dell'esperimento, l'andamento del prezzo rilevato è il seguente.

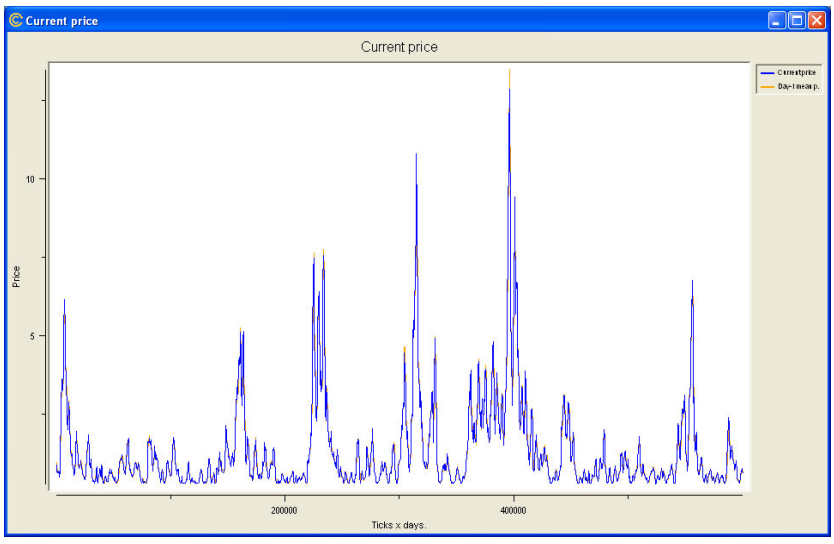


Figura 16- Prezzo in un mercato privo di agenti trainanti

Il mercato non è in grado di generare bolle di dimensione notevole e quelle che si creano raramente eccedono i *10* punti ed ancor meno i *13*. Un fenomeno che merita particolare attenzione riguarda il numero delle bolle generate. Nonostante siano di modesta entità, le bolle compaiono con una discreta frequenza e la volatilità del mercato risulta particolarmente elevata. Questo esperimento è stato condotto per poter essere utilizzato come campione nei confronti degli altri esperimenti eseguiti.

5.3.1.2 Esperimenti con agenti sensibili agli eventi (*Esperimenti 2*)

Il secondo gruppo di esperimenti introduce la presenza degli agenti sensibili agli eventi allo scopo di valutarne la loro incidenza e soprattutto individuare la soglia considerata critica oltre la quale il mercato risulti notevolmente destabilizzato. Gli esperimenti sono stai condotti in base alle configurazioni precedentemente esposte e per ogni esperimento è preso in considerazione il relativo prezzo formato sul mercato.

1. Agenti sensibili agli eventi pari al 2% della popolazione

L'esperimento *2a* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *2%*. L'andamento del prezzo è il seguente:

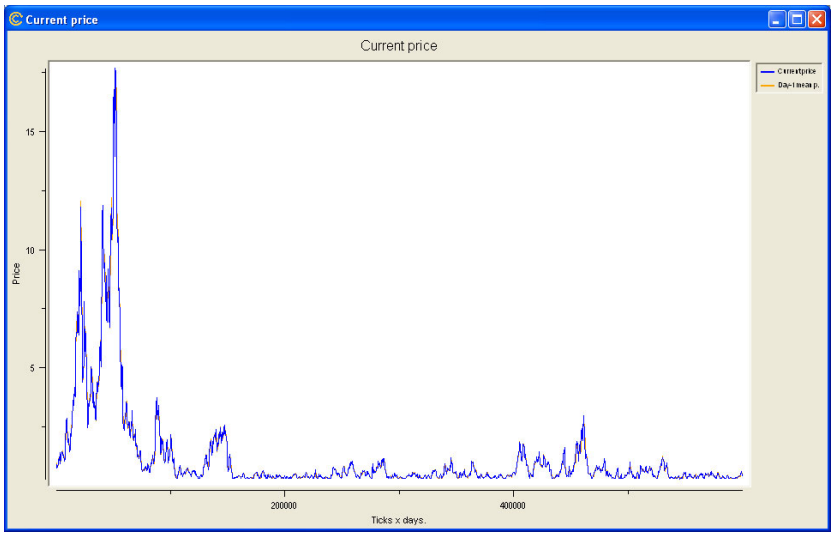


Figura 17 - Prezzo in presenza di agenti sensibili agli eventi in percentuale del 2%

I risultati che emergono da questo esperimento risultano molto interessanti. L'aspetto particolarmente interessante dell'esperimento riguarda il numero di bolle: decisamente inferiore rispetto al caso di riferimento. Con una percentuale pari al *2%* di agenti sensibili agli eventi, l'effetto che si ottiene è una riduzione della variabilità con

conseguente miglioramento della stabilità di mercato. Il motivo alla base di questo fenomeno è probabilmente localizzato nel comportamento degli agenti sensibili. L'effetto impresso al mercato da parte dagli agenti sensibili alle notizie è scindibile in due componenti separabili: una *direzione* ed un'*intensità*. La *direzione* dell'effetto dipende esclusivamente dalla natura dell'evento (*positiva* o *negativa*) ma l'*intensità* dell'effetto, oltre che dipendere dall'entità dell'evento, dipende anche dalla quantità di agenti sensibili presenti sul mercato. Si è già visto che la distribuzione degli eventi all'interno di un arco temporale sufficientemente lungo è caratterizzata da una distribuzione normale con media *0* e varianza *0.15*. La frequenza con cui si ripetono gli eventi positivi e negativi, determina negli agenti sensibili alle notizie un comportamento mediamente simmetrico. La direzione dell'effetto sarà mediamente nulla ma l'intensità dipenderà esclusivamente dalla quantità di agenti. Dai risultati emersi dall'esperimento in questione, si può affermare che il *2%* di tali agenti sia in grado di stabilizzare quasi perfettamente il mercato. L'intensità dell'effetto generato da questi agenti è circa equivalente all'intensità degli effetti generati dagli altri agenti del modello. Quando il mercato mostra regolarità uniformi, gli agenti appartenenti a questa categoria introducono un effetto correttivo che contribuisce alla stabilizzazione del mercato, ma quando gli eventi risultano estremamente positivi per una lungo periodo di tempo durante il quale il prezzo è già collocato su valori piuttosto elevati, il mercato enfatizza l'effetto dell'evento e genera bolle di entità ancora superiore rispetto al caso base. L'effetto riscontrabile sul mercato sarà il sorgere di un minor numero di bolle caratterizzate da un'entità più elevata.

2. Agenti sensibili agli eventi pari al 4% della popolazione

L'esperimento *2b* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *4%*. I risultati ottenuti sul prezzo sono i seguenti:

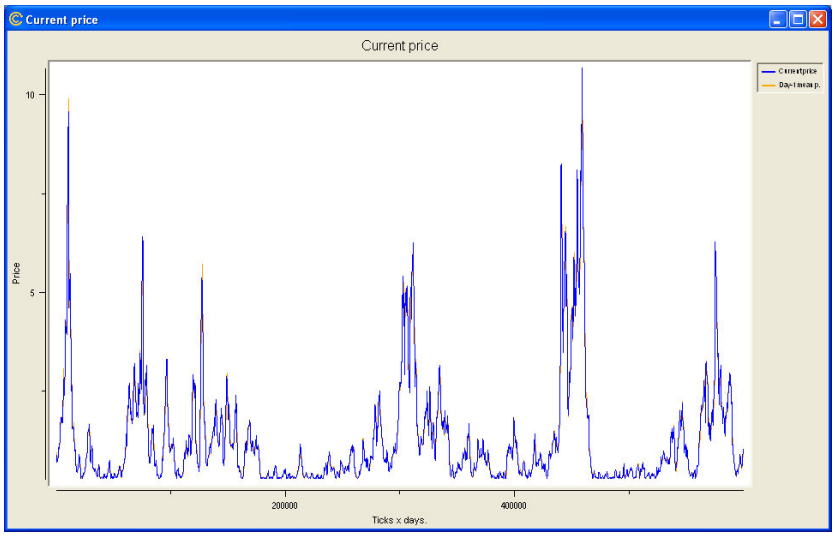


Figura 18 - Prezzo in presenza di agenti sensibili agli eventi in percentuale del 4%

La variabilità del mercato risulta maggiore ed il numero di bolle è notevolmente aumentato rispetto al caso precedente in cui gli agenti in esame rappresentavano solo il *2%* della popolazione totale. Nonostante la maggiore variabilità, l'entità assunta dalle bolle risulta però sensibilmente ridotta. La causa di questo fenomeno deve probabilmente essere ricercata ancora una volta nell'effetto che gli agenti sensibili agli eventi hanno nei confronti del mercato. La direzione dell'effetto che questi agenti introducono nel mercato è sempre mediamente pari a zero ma l'intensità che ne risulta è decisamente maggiore. Mentre una percentuale del *2%* sembra equilibrare in modo ottimale le forze che agiscono sul mercato, contribuendone alla sua stabilizzazione, una percentuale del *4%* fornisce un'intensità eccessiva che tende nuovamente a destabilizzare il mercato. L'effetto introdotto dalla direzione con cui agiscono gli agenti si può ancora riscontrare nella minore entità raggiunta dalle bolle, ma l'eccessiva intensità di tale effetto è altrettanto riscontrabile nella maggiore variabilità assunta dal mercato.

3. Agenti sensibili agli eventi pari al 5% della popolazione

L'esperimento *2c* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *5%*. L'andamento degli eventi ed i risultati ottenuti sul prezzo sono i seguenti:

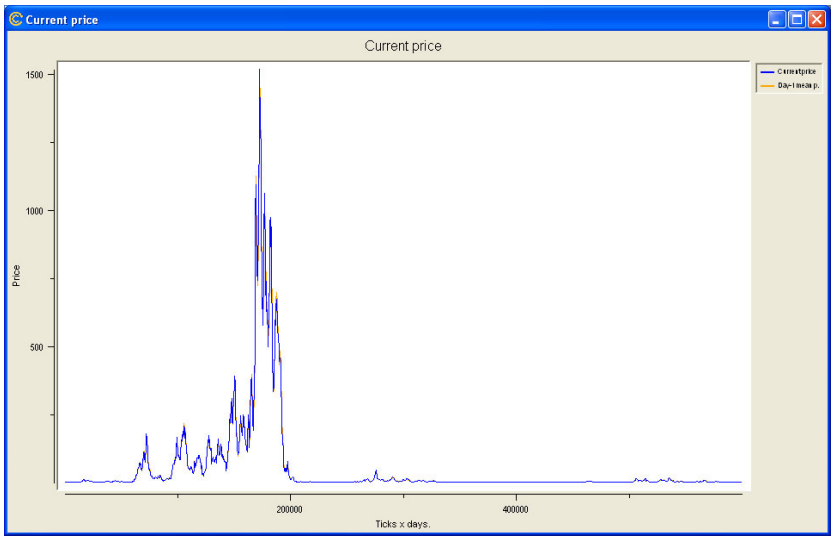


Figura 19 - Prezzo in presenza di agenti sensibili agli eventi in percentuale del 5%

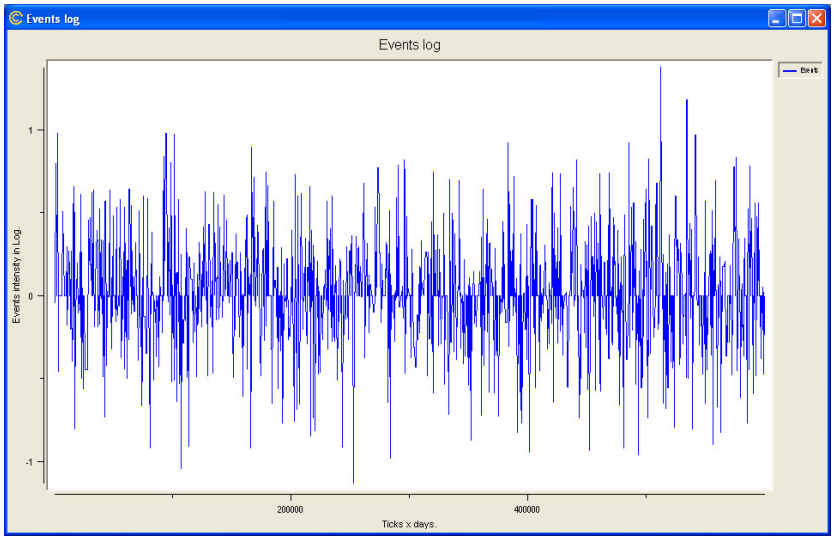


Figura 20 – Eventi in presenza di agenti sensibili agli eventi in percentuale del 5%

L'esperimento eseguito utilizzando una percentuale di agenti sensibili agli eventi pari al 5% della popolazione totale ha condotto ad un risultato molto particolare in grado di fornire un'adeguata risposta alla domanda che caratterizza il secondo obiettivo. Esiste una percentuale di tali agenti considerata critica in grado di destabilizzare il mercato? Sì, un mercato composto per il 5% da agenti sensibili agli eventi presenta una variabilità elevatissima e le bolle che in esso si generano possono raggiungere i *1500* punti. L'aspetto rilevante di questo fenomeno riguarda un'eventuale connessione tra l'entità dell'evento e la dimensione delle bolle. Da un'attenta analisi del prezzo e degli eventi si può facilmente notare che tra i due esiste meno correlazione di quanto non si potesse immaginare. La bolla di maggior rilievo è emersa approssimativamente poco prima del tic *200.000* ma osservando il grafico relativo agli eventi si può notare che nello stesso istante di tempo l'intensità degli eventi, benché fosse mediamente positiva,

non si differenzia in modo così significativo rispetto ad altri istanti di tempo anche successivi nei quali però non si sono formate bolle di entità così elevata. Esiste dunque una relazione tra entità degli eventi ed andamento del prezzo? Trattandosi di un sistema complesso è difficile fornire una spiegazione adeguata. Un'possibile ipotesi alla base del fenomeno giace probabilmente nella casualità con cui operano gli agenti, nell'intensità delle loro azioni e nella durata degli eventi. Dagli eventi generati emerge un'altra importante caratteristica: l'entità delle bolle non dipende tanto dall'entità dell'evento bensì dalla durata dello stesso. Eventi molto intensi ma di breve durata formano bolle meno rilevanti di quelle che si possono formare nel caso in cui l'evento sia di entità più modesta ma permanga per un periodo di tempo più lungo. Se il mercato si trova nelle condizioni per cui molti agenti casuali o limitatori di perdite agiscono contemporaneamente nella stessa direzione ed in quel momento gli eventi sono positivi per un lungo periodo di tempo, una percentuale di agenti sensibili agli eventi pari al *5%* è sufficiente per fornire al mercato l'enfasi necessaria per spingere il prezzo a livelli elevatissimi.

5.3.2 Fase II

La seconda fase della sperimentazione riguarda il perseguimento del terzo obiettivo. Quali effetti introducono altri agenti appartenenti alle categorie più trainanti in un mercato in cui sono già presenti agenti sensibili agli eventi? Gli esperimenti condotti in questa fase della sperimentazione riguardano questo argomento e sono stati condotti utilizzando le seguenti configurazioni:

Esperimento 3a	*Esperimento 3b*
Agenti Totali : 300	Agenti Totali : 300
Agenti Casuali : 270	Agenti Casuali : 261
Agenti Limitatori : 20	Agenti Limitatori : 20
Agenti Eventi : 6	Agenti Eventi : 15
Agenti Imitatori : 4	Agenti Imitatori : 4

Esperimento 4a	*Esperimento 4b*
Agenti Totali : 300	Agenti Totali : 300
Agenti Casuali : 270	Agenti Casuali : 261

Agenti Limitatori : 20	Agenti Limitatori : 20
Agenti Eventi : 6	Agenti Eventi : 15
Agenti Previsori : 4	Agenti Previsori : 4

Esperimento 5a	*Esperimento 5b*
Agenti Totali : 300	Agenti Totali : 300
Agenti Casuali : 270	Agenti Casuali : 261
Agenti Limitatori : 20	Agenti Limitatori : 20
Agenti Eventi : 6	Agenti Eventi : 15
Agenti Cognitivi : 4	Agenti Cognitivi : 4

Gli esperimenti sono stati suddivisi in base al tipo di agente trainante introdotto e le configurazioni adottate comprendono i casi in cui le percentuali di agenti sensibili agli eventi sono rispettivamente il *2%* ed il *5%*. La scelta di queste configurazioni è stata determinata in modo da rilevare l'effetto che ogni singola categoria introduce in un mercato nel quale siano già presenti alcuni agenti sensibili agli eventi mentre la scelta di effettuare due esperimenti per ciascuna categoria di agenti trainanti, relativi alle diverse percentuali di agenti sensibili, è giustificata dalla necessità di valutare disgiuntamente gli effetti che si possono generare in un mercato dove una diversa presenza di agenti sensibili è in grado di alterare significativamente i risultati.

5.3.2.1 Esperimenti con agenti imitatori (*Esperimenti 3*)

Il terzo gruppo di esperimenti introduce la presenza degli agenti imitatori con lo scopo di valutare la loro incidenza in un mercato in cui siano già presenti alcuni agenti sensibili agli eventi. Gli esperimenti sono stati condotti in base alle configurazioni sopra illustrate e per ogni esperimento ne è preso in considerazione il prezzo.

1. Agenti sensibili agli eventi pari al 2% della popolazione

L'esperimento *3a* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *2%* ed un numero di agenti imitatori pari a *4*. I risultati ottenuti sul prezzo sono i seguenti:

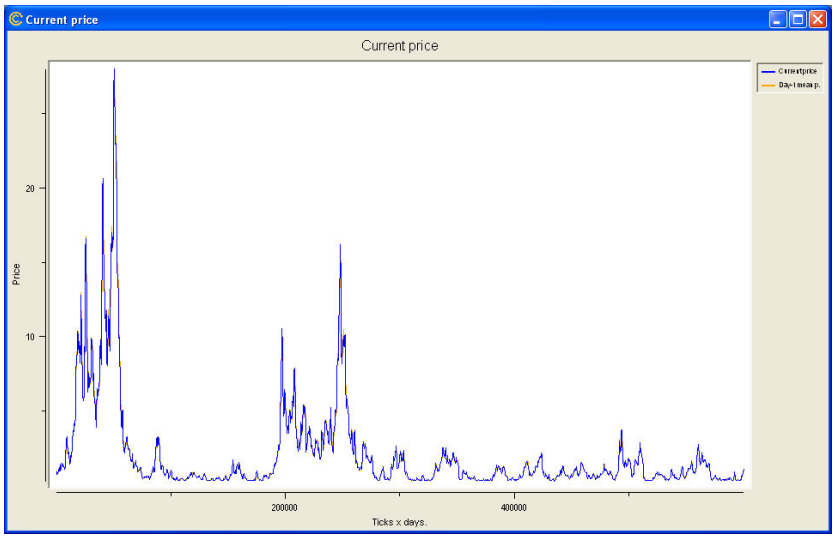


Figura 21 - Prezzo in presenza di agenti sensibili agli eventi pari al 2% con 4 agenti imitatori

La presenza di agenti imitatori all'interno del modello non altera sostanzialmente l'andamento del prezzo rispetto al caso base analizzato nell'esperimento *2a*. Com'era prevedibile, la presenza di agenti che imitano il mercato produce un effetto enfatizzante che ne compromette leggermente la stabilità. Le bolle di notevole rilevanza sono due ma l'effetto stabilizzante generato dalla bassa percentuale di agenti sensibili alle notizie continua a permanere ed è riscontrabile nella discreta stazionarietà mostrata dal prezzo nei tratti estranei alle bolle. In questi tratti il prezzo raramente eccede i *5* punti.

2. Agenti sensibili agli eventi pari al 5% della popolazione

L'esperimento *3b* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *5%* ed un numero di agenti imitatori pari a *4*. I risultati ottenuti sul prezzo sono i seguenti:

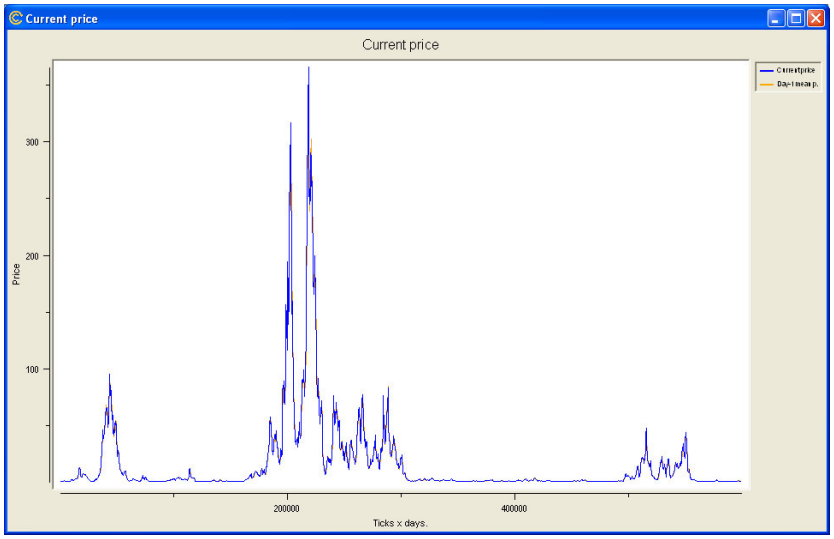


Figura 22 - Prezzo in presenza di agenti sensibili agli eventi pari al 5% con 4 agenti imitatori

Dall'esperimento in esame emergono risultati imprevedibili di notevole rilevanza. Nonostante la presenza del *5%* di agenti sensibili agli eventi fosse più che sufficiente a destabilizzare il mercato creando una bolla che raggiungeva i *1500* punti, l'introduzione di alcuni agenti imitatori sembra essere in grado di contrastare l'effetto distorsivo introdotto dagli agenti sensibili. La complessità del sistema impedisce la formulazione di sicure interpretazioni ma alcune ipotesi rivolte ad una minima comprensione del fenomeno possono comunque essere formulate. La causa principale di tale andamento è probabilmente localizzata nella casualità con cui gli agenti agiscono all'interno del modello durante le varie fasi della simulazione. Il fatto che in questo caso non si sia sviluppata una bolla di dimensioni molto elevate come nel caso *2c*, dimostra che la precedente bolla era probabilmente generata da una particolare sequenza positiva di eventi che non necessariamente richiedeva un'elevata entità degli stessi. L'ipotesi che una percentuale di agenti sensibili pari al *5%* sia sufficiente a destabilizzare il mercato, è confermata dalle bolle di notevole rilevanza ed entità che si generano anche in questo caso. In via principale se ne possono individuare quattro di cui una in grado di assumere valori superiori ai *350* punti.

L'effetto introdotto da questa categoria di agenti è decisamente poco definito. Sotto determinate condizioni, come avviene ad esempio nell'esperimento *3a*, il mercato risulta destabilizzato più di quanto non accada in assenza di tali agenti. In altre condizioni, come nell'esperimento *3b*, il mercato appare meno variabile e le bolle risultano meno elevate del rispettivo caso in cui vi è assenza di agenti imitatori. L'elemento determinante è probabilmente localizzato nel diverso numero di agenti sensibili agli eventi introdotti nel modello.

5.3.2.2 Esperimenti con agenti previsori (*Esperimenti 4*)

Il quarto gruppo di esperimenti introduce la presenza degli agenti previsori con lo scopo di valutarne la loro incidenza sul mercato. Gli esperimenti sono stati condotti in base alle configurazioni precedentemente esposte e per ogni esperimento è preso in considerazione il relativo prezzo.

1. *Agenti sensibili agli eventi pari al 2% della popolazione*

L'esperimento *4a* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *2%* ed un numero di agenti previsori pari a *4*. I risultati ottenuti sul prezzo sono i seguenti:

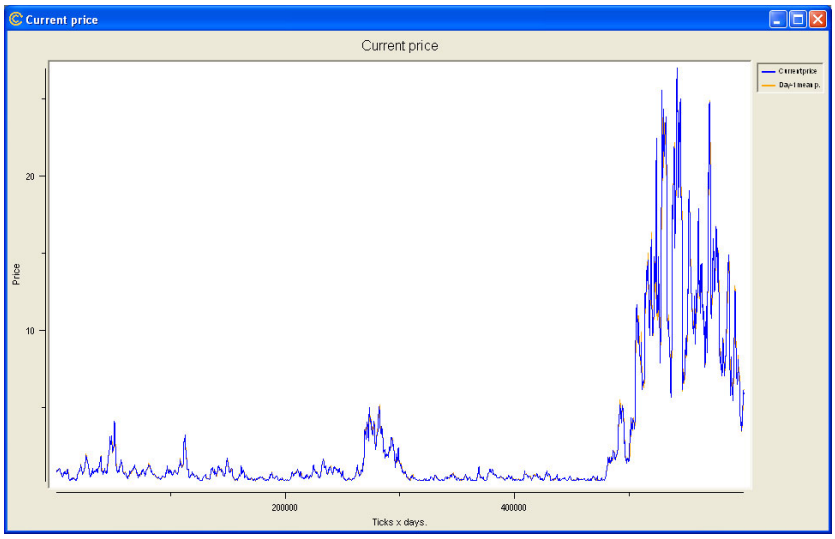


Figura 23 - Prezzo in presenza di agenti sensibili agli eventi pari al 2% con 4 agenti previsori

In un mercato avente il *2%* di agenti sensibili agli eventi e *4* agenti previsori, il prezzo mostra caratteristiche del tutto particolari. Da un lato si verifica un drastico calo delle bolle rispetto al caso base, dall'altro, l'unica vera bolla che si verifica risulta di discreta entità e decisamente prolungata nel tempo. Per comprendere meglio quale possa essere l'incidenza degli agenti previsori sul funzionamento di tutto il mercato, si prenda in considerazione il grafico delle previsioni.

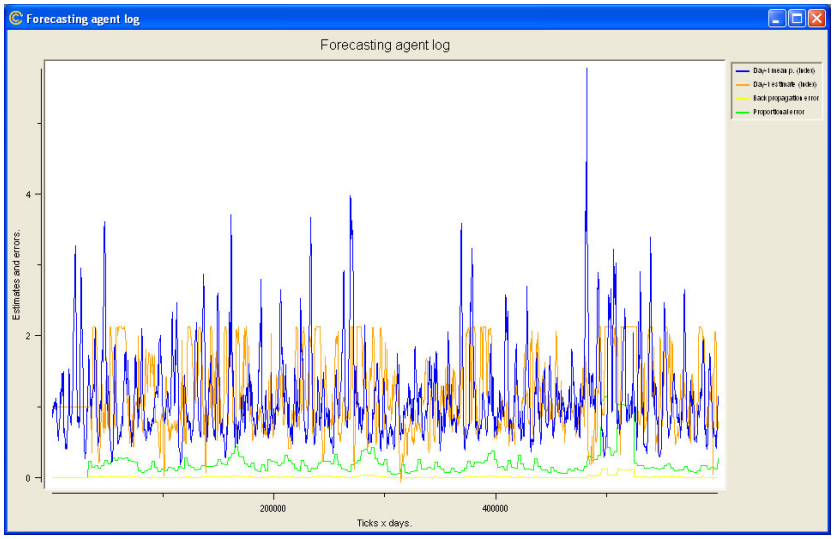


Figura 24 – Previsioni in un mercato con agenti sensibili agli eventi in percentuale del 2%

L'effetto che gli agenti previsori introducono nel modello è probabilmente duplice. Da un lato, grazie alle loro previsioni prevalentemente stabili, contribuiscono alla stabilizzazione del mercato sommando il proprio effetto a quello generato dalla piccola popolazione di agenti sensibili agli eventi i quali continuano anche in questo caso ad agire come stabilizzatori del mercato, dall'altro lato introducono un elemento distorsivo

che in alcune condizioni può rappresentare uno stimolo al rialzo o al ribasso per l'intero mercato. Osservando attentamente il grafico si può notare che in corrispondenza dell'inizio della bolla, corrisponde una previsione valutata sopra la media che raggiunge i *6* punti. L'ipotesi più accreditata tenderebbe ad affermare che questa previsione azzardata sia la reale causa della nascita della bolla e, in un mercato dotato di tali caratteristiche, quattro agenti previsori sono probabilmente sufficienti per innescare un processo in grado di alterare significativamente il prezzo. Nella prima parte dell'esperimento, il comportamento dimostrato dal mercato può essere considerato decisamente stabile e raramente il prezzo supera i *2* punti.

2. Agenti sensibili agli eventi pari al 5% della popolazione

L'esperimento *4b* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *5%* ed un numero di agenti previsori pari a *4*. I risultati ottenuti sul prezzo sono i seguenti:

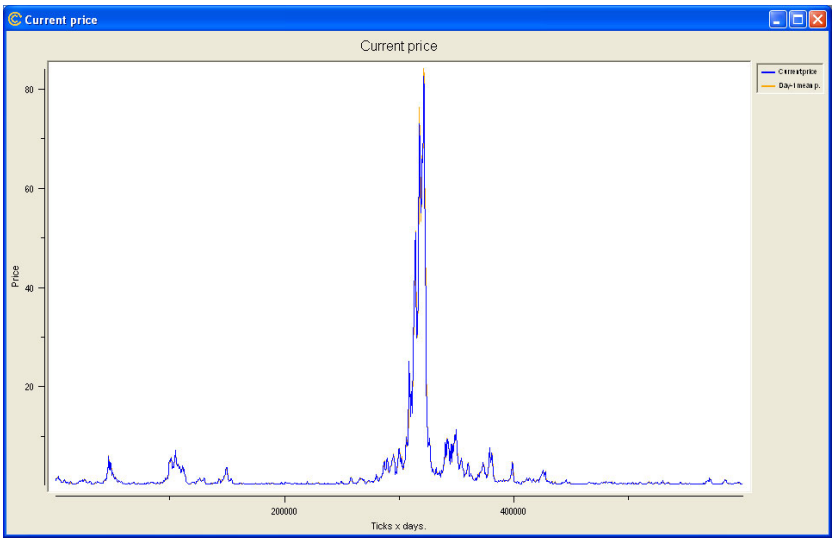


Figura 25 - Prezzo in presenza di agenti sensibili agli eventi pari al 5% con 4 agenti previsori

Le bolle che si verificano sono poche e l'effetto stabilizzante che gli agenti previsori hanno in un mercato dotato di queste caratteristiche è elevatissimo. Grazie all'azione degli agenti previsori, un mercato avente il 5% di popolazione caratterizzata da agenti sensibili alle notizie risulta molto più stabile rispetto al caso base. Si prenda in considerazione il grafico relativo alle previsioni.

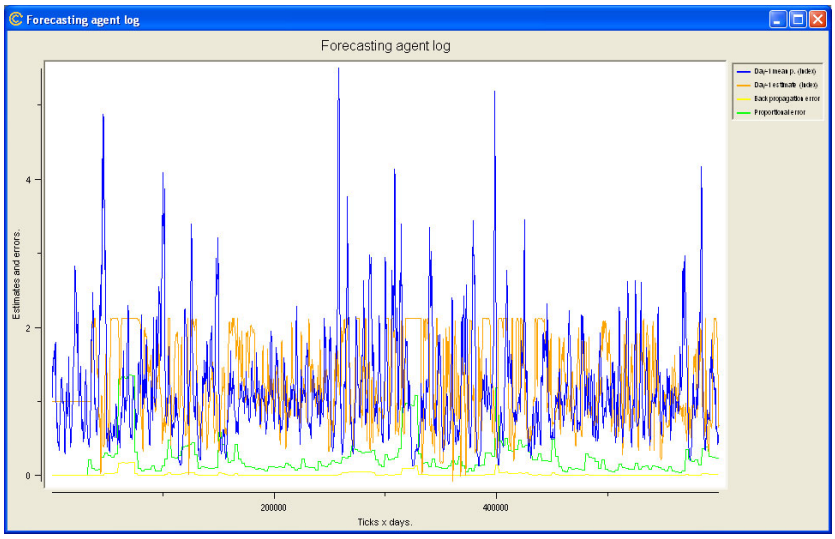


Figura 26 – Previsioni in un mercato con agenti sensibili agli eventi in percentuale del 5%

Raramente il prezzo eccede i *10* punti e solo in un caso si forma una bolla che sfiora gli *80* punti. La bolla è probabilmente generata dall'azione congiunta di diversi agenti sensibili ma la sua entità risulta notevolmente minore di quella verificata nell'esperimento *2c*. Confrontando il grafico del prezzo con quello delle previsioni risulta meno evidente la connessione tra i due. Nell'istante in cui il prezzo inizia a salire per generare l'unica bolla di rilievo all'interno di tutta la simulazione, non si riscontrano previsioni di particolare rilevanza. Questo fenomeno conferma l'ipotesi che la bolla sia stata generata dell'elevata percentuale di agenti sensibili introdotta nel modello.

L'effetto degli agenti previsori, in un mercato dotato di queste caratteristiche, non dipende dal numero di agenti sensibili ed è prevalentemente positivo. In entrambi i casi si assiste ad una stabilizzazione del mercato e solo in rari casi, alcune previsioni errate, possono stimolare la nascita di distorsioni rilevanti sul prezzo.

5.3.2.2 Esperimenti con agenti cognitivi (*Esperimenti 5*)

Il quinto gruppo di esperimenti introduce nel modello gli agenti cognitivi per valutare il loro grado di incidenza sul mercato. Gli esperimenti seguono le configurazioni sopra esposte.

1. Agenti sensibili agli eventi pari al 2% della popolazione

L'esperimento *5a* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *2%* ed un numero di agenti cognitivi pari a *4*. I risultati ottenuti sul prezzo sono i seguenti:

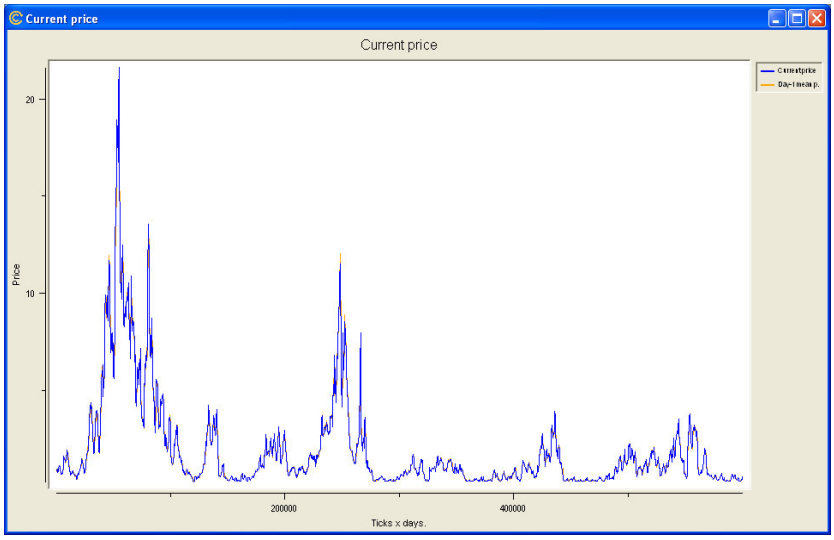


Figura 27 - Prezzo in presenza di agenti sensibili agli eventi pari al 2% con 4 agenti cognitivi

L'effetto che si osserva da questo esperimento era prevedibile. Si assiste ad un peggioramento rispetto al caso *2a* sia per quanto riguarda il numero di bolle che per quanto concerne la loro entità. L'entità assunta dalle bolle è leggermente maggiore ma risulta comunque decisamente contenuta e solo in un caso supera i *20* punti. L'effetto stabilizzante di una piccola percentuale di agenti sensibili agli eventi continua a permanere ed è riscontrabile nella modesta variabilità assunta dal prezzo nei tratti in cui non sono presenti bolle di rilievo; in questi tratti il prezzo raramente eccede i *5* punti. La causa della maggiore variabilità del mercato è probabilmente localizzata nell'introduzione stessa degli agenti cognitivi. Uno solo non è caratterizzato obiettivi esterni mentre gli altri tre sono ciascuno caratterizzati da obiettivi esterni di arricchimento. L'obbligo da parte degli agenti cognitivi di arricchirsi, contribuisce ad alterare la variabilità del mercato introducendo una maggiore volatilità nel prezzo.

2. Agenti sensibili agli eventi pari al 5% della popolazione

L'esperimento *5b* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *5%* ed un numero di agenti cognitivi pari a *4*. I risultati ottenuti sul prezzo sono i seguenti:

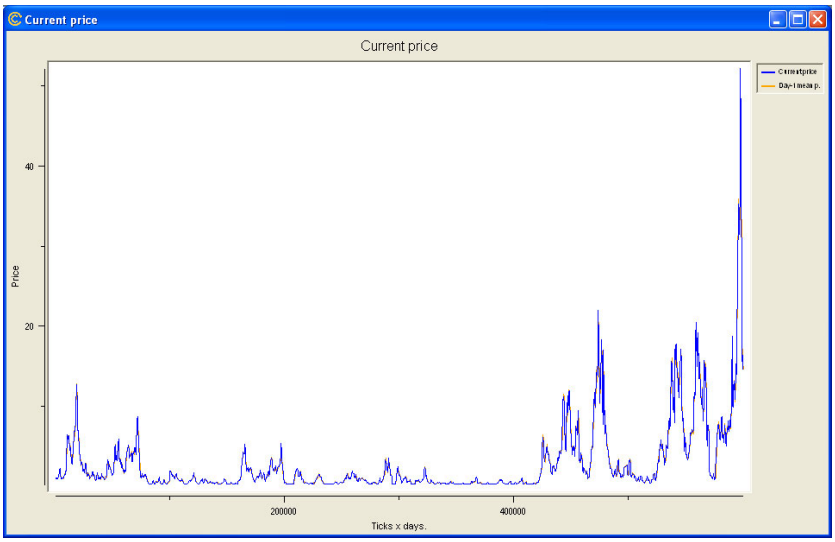


Figura 28 - Prezzo in presenza di agenti sensibili agli eventi pari al 5% con 4 agenti cognitivi

L'introduzione di agenti cognitivi è in grado di ridurre notevolmente la variabilità del prezzo, non solo per quanto riguarda il numero di bolle, ma anche per quanto riguarda la loro entità. Nonostante la bolla verificata nell'ultimo istante di tempo abbia superato i *50* punti, e nonostante la percentuale di agenti sensibili introdotti nel modello possa essere considerata critica, l'andamento del mercato risulta pressoché identico a quello sviluppato nel caso in cui il numero di agenti sensibili sia solo pari al *2%*. Solo in due casi le bolle superano i *20* punti e raramente il prezzo eccede i *5* punti. Anche in questo caso, la maggiore capacità cognitiva degli agenti, sembra essere in grado di contrastare l'effetto enfatizzante introdotto dagli agenti sensibili agli eventi.
L'effetto introdotto nel mercato da questa categoria di agenti è simile a quello introdotto dagli agenti previsori. Se la percentuale di agenti sensibili agli eventi è considerata modesta si assiste ad una leggera destabilizzazione del mercato, ma se la percentuale di agenti sensibili è ritenuta critica, l'introduzione di pochi agenti cognitivi sembra sufficiente per contrastare l'elevato effetto distorsivo introdotto dalle altre tipologie di agenti.

5.3.3 Fase III

La terza fase degli esperimenti riguarda il perseguimento del quarto ed ultimo obiettivo. Quali effetti derivano dalla modifica di alcuni parametri di rilievo per un mercato in cui siano presenti alcuni agenti sensibili agli eventi? Gli esperimenti condotti in questa fase della sperimentazione integrano lo studio dell'argomento e rappresentano una chiave fondamentale per una migliore comprensione degli esperimenti condotti finora. Riguardano tre aspetti:

1) **Variazione nel comportamento degli agenti sensibili agli eventi.**
Tramite questa variazione, gli agenti sensibili agli eventi sono tenuti ad operare in fase di preapertura ogni volta che in quel momento sia in corso un evento. In tal caso la scelta se agire in preapertura o in fase di contrattazione continua non sarà più lasciata a discrezione del singolo agente bensì obbligatoria. Se un evento sarà presente in fase di preapertura, tutti gli agenti appartenenti a questa categoria opereranno in quell'istante. Lo scopo di questo esperimento è rivolto all'individuazione di un'eventuale relazione tra bolle e fase di preapertura.

2) **Variazione della sensibilità degli agenti.**
Tramite questa modifica si cerca di capire quale influenza possa avere su tutto il sistema un aumento od una diminuzione della sensibilità degli agenti appartenenti alla categoria in esame. I possibili effetti sono prevedibili ma quello che interessa scoprire è l'esistenza di una proporzionalità reciproca tra le diverse grandezze del modello. Dimezzando la sensibilità, il prezzo aumenta o diminuisce e, nel caso diminuisca, diminuirà all'incirca della metà, più della metà o meno della metà? La stessa domanda può ovviamente essere formulata nel caso si esegua la modifica in senso opposto ossia si raddoppi la sensibilità.

3) **Variazione del numero di agenti presenti nel modello.**
Tramite questa variazione si aumenta la popolazione del modello mantenendo costante la ripartizione di base. Lo scopo di questo esperimento è rivolto allo studio dell'incidenza che la quantità di agenti ha nei confronti dell'intero sistema. Aumentando la popolazione del modello, i risultati cambiano sensibilmente oppure restano all'incirca inalterati? Nel caso si verifichino delle variazioni, saranno migliori o peggiori?

Gli esperimenti relativi alle tre varianti citate sono stati effettuati nelle seguenti condizioni:

Esperimento 6	*Esperimento 7*	*Esperimento 8*
Agenti Totali : *300*	Agenti Totali : *300*	Agenti Totali : *600*
Agenti Casuali : *265*	Agenti Casuali : *265*	Agenti Casuali : *530*
Agenti Limitatori : *20*	Agenti Limitatori : *20*	Agenti Limitatori : *40*
Agenti Eventi : *15*	Agenti Eventi : *15*	Agenti Eventi : *30*

5.3.3.1 Variazione nel comportamento degli agenti sensibili (Esperimento 6)

L'esperimento *6* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *5%* ed è stata eliminata la possibilità da parte degli agenti di agire discrezionalmente in fase di preapertura. I risultati ottenuti sul prezzo sono i seguenti:

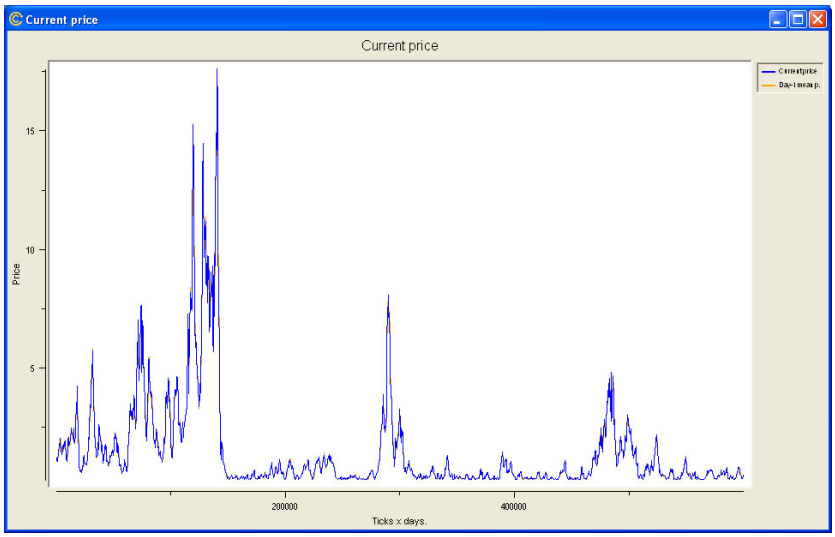


Figura 29 - Prezzo in presenza di agenti sensibili agli eventi pari al 5% con vincolo di preapertura

Il prezzo che si forma sul mercato mostra diverse caratteristiche. La bolla che nel caso base raggiungeva i *1500* punti continua a permanere ma la sua entità risulta estremamente ridotta ed il prezzo non supera ora i *18* punti. L'obbligo di azione in fase di preapertura da parte degli agenti sensibili agli eventi nel caso sussista un evento di entità non nulla, rende il loro comportamento molto più trasparente al mercato che in questo caso non è più in grado di subire gli effetti derivanti dall'azione degli stessi agenti durante la fase di contrattazione continua. Il motivo alla base del fenomeno è probabilmente localizzato nella modalità con cui avvengono i contratti durante le due fasi di contrattazione. Gli ordini che sono introdotti durante la fase di preapertura sono valutati in modo diverso rispetto agli ordini introdotti durante la fase di contrattazione continua ed il modo con cui sono eseguiti i contratti durante queste due fasi risulta determinante per la stabilità del mercato. Accertato che in simili condizioni il mercato reagisce in maniera molto più modesta alle reazioni degli agenti sensibili alle notizie, si può affermare che la soglia limite per la valutazione della stabilità sarà in questo caso indubbiamente più elevata. Se gli agenti sensibili alle notizie reagissero tutti in fase di preapertura quando sussiste una notizia, l'effetto che si genererebbe sul mercato sarebbe notevolmente ridotto ed il prezzo assumerebbe un andamento molto più regolare, simile a quello ottenuto con una minore percentuale di agenti sensibili.

5.3.3.2 Variazione della sensibilità degli agenti (Esperimento 7)

L'esperimento *7* ha utilizzato una percentuale di agenti sensibili agli eventi pari al *5%* modificandone la loro sensibilità. La sensibilità utilizzata in tutti gli altri esperimenti era fissata pari a *0.20* mentre la sensibilità utilizzata in questo esperimento è fissata pari a *0.15*. In termini matematici si può affermare che è stata ridotta del *25%* rispetto al suo valore originale. Lo scopo della simulazione riguarda l'individuazione di una proporzionalità esistente tra la sensibilità e l'andamento del mercato. I risultati ottenuti sul prezzo sono i seguenti:

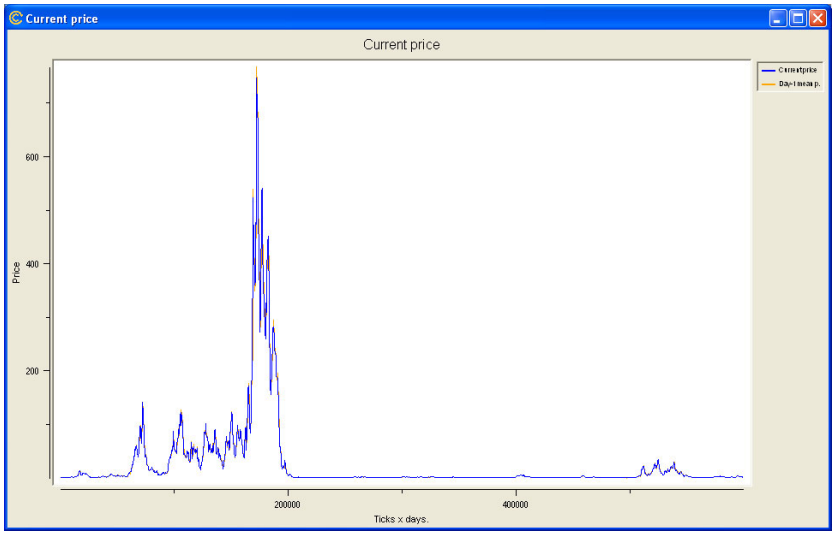


Figura 30 - Prezzo in presenza di agenti sensibili agli eventi pari al 5% con sensibilità 0.15

Da questo esperimento emergono aspetti che non erano previsti. La forma del grafico è identica al caso base, ma la bolla che si genera ha dimensioni decisamente più contenute. Come si può facilmente osservare dal grafico, il massimo valore raggiunto dalla bolla sfiora i *750* punti. Se lo si confronta con i dati ottenuti dal caso base, si può notare che la bolla si è ridotta del *50%*. L'esperimento in esame risponde perfettamente alla domanda oggetto di studio. La riduzione dell'effetto è proporzionale alla riduzione della sensibilità? Da quanto emerge dai dati sembrerebbe di no. Una riduzione nella sensibilità del *25%* genera una riduzione nell'entità della bolla sul prezzo pari al *50%*. In sintesi è possibile affermare che la sensibilità degli agenti è certamente un elemento determinante per i risultati che possono emergere in un mercato di questo tipo. Il fatto che ogni singolo agente possa avere una sensibilità agli eventi propria, rappresenta un ulteriore elemento di studio che caratterizza un sistema complesso come quello in esame.

5.3.3.3 Variazione del numero di agenti presenti nel modello (Esperimento 8)

L'esperimento *7* ha utilizzato le stesse percentuali utilizzate nell'esperimento 2c ma la popolazione del modello, intesa come numero di agenti, è stata aumentata. Gli agenti presenti ora nel modello sono *600*. Lo scopo della simulazione riguarda l'influenza che il numero di agenti ha sull'andamento del mercato. I risultati ottenuti sul prezzo sono i seguenti:

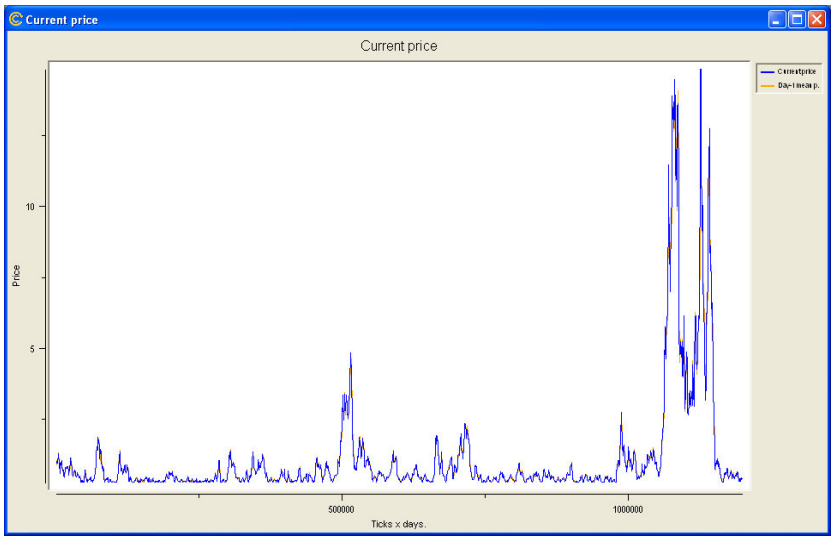


Figura 31 - Prezzo in presenza di agenti sensibili agli eventi pari al 5% con popolazione di 600

I risultati erano prevedibili. Una popolazione di *600* agenti riduce drasticamente la variabilità di un mercato indipendentemente dalla sua composizione di agenti. Le bolle raramente eccedono i *2.5* punti e solo in un caso, verificatosi al termine della simulazione, si è generata una bolla di entità comunque ridotta che non ha superato i *15* punti. La causa che ha condotto a questi risultati è probabilmente localizzata nel numero di agenti presenti nel modello. Si è detto che le percentuali delle singole categorie di agenti non sono state alterate ma un numero di agenti maggiore modifica probabilmente il processo casuale con cui i singoli agenti operano introducendo un maggior rumore di fondo. Data la maggiore dimensione del mercato, una percentuale del *5%* di agenti sensibili agli eventi sulla popolazione totale non è più in grado di destabilizzare il mercato anzi, sembrerebbe fornire quell'effetto stabilizzante che in precedenza era caratterizzato da una percentuale del *2%*. Il ragionamento è coerente se si riflette sul numero degli agenti introdotti: esattamente il doppio rispetto al caso precedente. I risultati emersi da questa simulazione permettono di comprendere un ulteriore fenomeno: la stabilità di un mercato non dipende solamente dal numero di agenti destabilizzanti presenti, ma anche dalla liquidità stessa del mercato rappresentata in questo caso dal numero di agenti operanti. Un mercato poco liquido

non sempre è in grado di contrastare l'effetto destabilizzante introdotto da alcuni agenti, ma un mercato decisamente più liquido, a parità di condizioni, riesce a contrastare decisamente meglio tutte le forze agenti su di esso introdotte dalle varie categorie di agenti.

5.4 Conclusioni

Gli aspetti più importanti che sono emersi da queste simulazioni possono essere sintetizzati in sei punti:

1) Una percentuale di agenti sensibili agli eventi pari al *2%* della popolazione totale introduce un effetto stabilizzante all'interno del mercato. La forza che deriva dall'operato di pochi agenti appartenenti alla categoria in esame risulta essere in ottimo equilibrio con le altre forze agenti sul mercato. La variabilità risulta notevolmente ridotta ma le bolle che si creano assumono entità maggiore rispetto al caso di riferimento. Considerando le caratteristiche attribuite agli agenti sensibili agli eventi, questo fenomeno era prevedibile ma non previsto.

2) Una percentuale di agenti sensibili pari al 5% della popolazione totale, a differenza di quanto accade nel caso precedente, risulta eccessiva e compromette notevolmente tanto la variabilità del mercato quanto l'entità delle bolle che in esso si possono generare. La forza che deriva dall'operato di un numero così elevato di agenti sensibili alle notizie eccede la somma delle altre forze che agiscono sul mercato alterando sostanzialmente l'andamento del prezzo. Questa condizione non si verifica però se gli agenti sensibili alle notizie operano obbligatoriamente in fase di preapertura quando sussistono degli eventi. In tal caso, anche una percentuale di agenti pari al 5% della popolazione totale, risulta insufficiente per destabilizzare significativamente il mercato e, nonostante un leggero aumento di volatilità, il prezzo risulta comunque più stabile.

3) L'entità assunta dalle bolle non è direttamente legata all'intensità degli eventi bensì alla durata che questi assumono durante la fase di simulazione. Un evento di notevole intensità ma di breve durata produce minori effetti di un evento meno rilevante ma di durata decisamente più lunga. Questa

caratteristica è in grado di spiegare perché i mercati finanziari subiscano maggiori flessioni quando le notizie hanno rilevanza modesta ma permangono per più tempo piuttosto che il caso in cui le notizie mostrano una elevata rilevanza ma permangono per meno tempo.

4) Il momento in cui gli agenti sensibili alle notizie operano è determinante per l'andamento del mercato. Se gli agenti agiscono prevalentemente in fase di preapertura, l'intensità del loro effetto è decisamente minore rispetto al caso in cui operano in fase di contrattazione continua. Il motivo è localizzato nella modalità con cui è formato il prezzo nelle due fasi di contrattazione. La formazione del prezzo in fase di preapertura è in grado di occultare l'effetto introdotto dagli agenti sensibili agli eventi aumentando notevolmente la stabilità del mercato.

5) La sensibilità degli agenti non è direttamente proporzionale all'entità dei risultati conseguibili. Una riduzione della sensibilità produce una riduzione più che proporzionale della variabilità del mercato e le bolle che si possono generare risultano di entità decisamente ridotta.

6) A parità di condizioni, un mercato decisamente più liquido, è in grado di contrastare meglio l'effetto destabilizzante introdotto da alcune categorie di agenti. Dai risultati emersi in fase di sperimentazione è lecito pensare che la stabilità di un mercato sia proporzionale al numero di agenti operanti: maggiore sarà la liquidità, minore sarà la sua variabilità. Questo fenomeno è riscontrabile anche nella realtà economica quotidiana.

Appendice

A. Listato del codice

A.1 ObserverSwarm.h

```
#import "ModelSwarm.h"
#import "Book.h"
#import "EventGenerator.h"
#import "ForecastingAgent.h"

//agentForge step 10b
#import "BPCTAgentA.h"
#import "BPCTAgentAInterface.h"
#import "BPCTAgentB.h"
#import "BPCTAgentBInterface.h"

#import <simtoolsgui/GUISwarm.h>
#import <analysis.h> // to use EZgraph


@interface ObserverSwarm: GUISwarm
{
       int displayFrequency;       // freq. of update in the
                                  // Observer widgets
       int showBookGraph, showVolumesGraph, showAgentWealthGraph,
       showForecastingAgentGraph, showEventsGraph,
       displayPreviousDayMean, savePriceData, saveForecastingData,
       saveBookData, saveVolumesData, saveAgentWealthData,  saveEventsData,

// BPCT Agent
       numberOfTheBPCTAgentA_ToBeObservedDirectly,
      saveBPCTAgentAData,
      numberOfTheBPCTAgentB_ToBeObservedDirectly,
      saveBPCTAgentBData,

      bPCTPatternNumberInVerificationSet;

  id displayActions;          // schedule data structures
  id displaySchedule;

  // agentForge step 10
        id <EZGraph> priceGraph, volumesGraph, bookGraph, eventsGraph,
      agentWealth, agentWealthGraph,
     forecastingAgentGraph, bPCTAgentA_Graph, bPCTAgentB_Graph;

  int stopAtDayNumber;

  ModelSwarm *modelSwarm;         // the Swarm containing our model
  Book * theBook;
  ForecastingAgent * forecastingAgent;
  EventGenerator * eventGenerator;


  //agentForge step 10d
  BPCTAgentA * agentA;
  BPCTAgentAInterface * agentAInterface;
  BPCTAgentB * agentB;
  BPCTAgentBInterface * agentBInterface;

}

+ createBegin: aZone;
- createEnd;
- buildObjects;
```

```
- buildActions;
- activateIn: swarmContext;

- checkToStop;

@end
```

A.2 ObserverSwarm.m

```
#import "ObserverSwarm.h"
#import <activity.h>
#import <simtoolsgui.h>

@implementation ObserverSwarm

+ createBegin: aZone
{
  ObserverSwarm *obj;
  id <ProbeMap> probeMap;

  // Superclass createBegin to allocate ourselves.

  obj = [super createBegin: aZone];

  // Fill in the relevant parameters.

  obj->displayFrequency          = 300;

  obj->stopAtDayNumber           = 2000; // to stop the program (if != 0)
  obj->displayPreviousDayMean    = 1;
  obj->savePriceData             = 0;
  obj->showBookGraph             = 1;
  obj->saveBookData              = 0;
  obj->showVolumesGraph          = 1;
  obj->saveVolumesData           = 0;
  obj->showAgentWealthGraph      = 1;
  obj->saveAgentWealthData       = 0;
  obj->showForecastingAgentGraph = 1;
  obj->saveForecastingData       = 0;
  obj->showEventsGraph             = 1;
  obj->saveEventsData            = 0;

  //agentForge step 11b
  obj->numberOfTheBPCTAgentA_ToBeObservedDirectly = 1;
  obj->saveBPCTAgentAData = 1;
  obj->numberOfTheBPCTAgentB_ToBeObservedDirectly = 1;
  obj->saveBPCTAgentBData = 1;

  // to build a customized probe map
  // without a probe map, the default is to show all variables and messages
  // here we choose to customize the appearance of the probe, to give a nicer
  // interface

  probeMap = [EmptyProbeMap createBegin: aZone];
  [probeMap setProbedClass: [self class]];
  probeMap = [probeMap createEnd];

  // add variables to be probed

  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "displayFrequency"           inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "stopAtDayNumber"            inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "displayPreviousDayMean"     inClass: [self class]]];
```

```
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "savePriceData"              inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "showBookGraph"              inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "saveBookData"               inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "showVolumesGraph"           inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "saveVolumesData"            inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "showAgentWealthGraph"       inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "saveAgentWealthData"        inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "showForecastingAgentGraph"  inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "saveForecastingData"        inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "showEventsGraph"            inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "saveEventsData"             inClass: [self class]]];


  //agentForge step 11c
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "numberOfTheBPCTAgentA_ToBeObservedDirectly"
                                         inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "saveBPCTAgentAData"         inClass: [self class]]];

  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "numberOfTheBPCTAgentB_ToBeObservedDirectly"
                                         inClass: [self class]]];
  [probeMap addProbe: [probeLibrary getProbeForVariable:
            "saveBPCTAgentBData"         inClass: [self class]]];

  // set our custom probeMap into the probeLibrary as the default probe for
  // the ObserverSwarm class

  [probeLibrary setProbeMap: probeMap For: [self class]];

  return obj;
}

- createEnd
{
  return [super createEnd];
}


- buildObjects
{
  [super buildObjects];

  modelSwarm = [ModelSwarm create: self];

  // to create probe objects on the model and the observer (self, here)
  // ARCHIVED to allow the "Save" button to operate

  CREATE_ARCHIVED_PROBE_DISPLAY (modelSwarm);
  CREATE_ARCHIVED_PROBE_DISPLAY (self);

  // we pause here to allow the parameters to be changed.

  [controlPanel setStateStopped];


  // The system will wait until the user hits "Start" or "Next"
  // on the control panel
```

```
[modelSwarm buildObjects];

// checking the consistence of the displayFrequency with the agentNumber:
// for display reasons it is necessary to update the graphic widgets
// after all agent actions (mainly the BPCT ones, that calculate their
// targets at the end of a day and their inputs at the beginning of the day)
// so the displays are updated at displayFrequency-1
// if displayFrequency=1, in the agentNumber steps of a day we have the
// new BPCT input and output updated at the first step and the BPCT target
// updated at the last
// if we adopt a displayFrequency multiple of angentNumber we have the
// apparent coincidence of these values (##)

if(displayFrequency !=1 && displayFrequency %
   [[modelSwarm getAgentList] getCount] != 0)
  {
    printf("displayFrequency must be 1 or multiple of agentNumber.\n");
    exit(0);
  }

theBook=[modelSwarm getBook];
forecastingAgent=[modelSwarm getForecastingAgent];
eventGenerator=[modelSwarm getEventGenerator];

// Observer display objects.

// The current price graph
priceGraph = [EZGraph createBegin: self];
SET_WINDOW_GEOMETRY_RECORD_NAME (priceGraph); // to allow "Save"

if(savePriceData==1)[priceGraph setFileOutput: (BOOL) 1];
                    // to send the data also to a file

[priceGraph setTitle: "Current price"];
[priceGraph setAxisLabelsX: "Ticks x days." Y: "Price"];
priceGraph = [priceGraph createEnd];
// that above is the old way of creating an EZgraph; now a unique
// methos exists

// pay attention: the lines below cannot be placed before createEnd
[priceGraph createSequence: "Current price" withFeedFrom: theBook
                                            andSelector: M(getPrice)];
if(displayPreviousDayMean==1)
[priceGraph createSequence: "Day-1 mean p." withFeedFrom: theBook
                                            andSelector: M(getMeanPrice)];

// The book graph
if (showBookGraph==1)
{
bookGraph = [EZGraph createBegin: self];
SET_WINDOW_GEOMETRY_RECORD_NAME (bookGraph); // to allow "Save"

if(saveBookData==1)[bookGraph setFileOutput: (BOOL) 1];
                   // to send the data also to a file

[bookGraph setTitle: "Book log"];
[bookGraph setAxisLabelsX: "Ticks x days." Y:"Sell and Buy Orders in Log."];
bookGraph = [bookGraph createEnd];
// that above is the old way of creating an EZgraph; now a unique
// methos exists

// pay attention: the lines below cannot be placed before createEnd

[bookGraph createSequence: "Sell orders" withFeedFrom: theBook
                                       andSelector: M(getSellOrderNumber)];
[bookGraph createSequence: "Buy orders"  withFeedFrom: theBook
                                       andSelector: M(getBuyOrderNumber)];
}
```

```
// The Volumes graph
if (showVolumesGraph==1)
{
volumesGraph = [EZGraph createBegin: self];
SET_WINDOW_GEOMETRY_RECORD_NAME (volumesGraph); // to allow "Save"

if(saveVolumesData==1)[volumesGraph setFileOutput: (BOOL) 1];
                  // to send the data also to a file

[volumesGraph setTitle: "Volumes log"];
[volumesGraph setAxisLabelsX: "Ticks x days." Y: "Volumes in Log."];
volumesGraph = [volumesGraph createEnd];
// that above is the old way of creating an EZgraph; now a unique
// methos exists

// pay attention: the lines below cannot be placed before createEnd

[volumesGraph createSequence: "Price Volumes" withFeedFrom: theBook
                                      andSelector: M(getPriceVolumes)];
[volumesGraph createSequence: "Qty Volumes"  withFeedFrom: theBook
                                      andSelector: M(getQuantityVolumes)];
}

// The Events graph
if (showEventsGraph==1)
{
eventsGraph = [EZGraph createBegin: self];
SET_WINDOW_GEOMETRY_RECORD_NAME (eventsGraph); // to allow "Save"

if(saveEventsData==1)[eventsGraph setFileOutput: (BOOL) 1];
                  // to send the data also to a file

[eventsGraph setTitle: "Events log"];
[eventsGraph setAxisLabelsX: "Ticks x days." Y: "Events intensity in Log."];
eventsGraph = [eventsGraph createEnd];
// that above is the old way of creating an EZgraph; now a unique
// methos exists

// pay attention: the lines below cannot be placed before createEnd

[eventsGraph createSequence: "Events" withFeedFrom: eventGenerator
                                      andSelector: M(getEventState)];
}


// The agent's wealth graph

if (showAgentWealthGraph==1)
{
agentWealthGraph = [EZGraph createBegin: self];
SET_WINDOW_GEOMETRY_RECORD_NAME (agentWealthGraph); // to allow "Save"

if(saveAgentWealthData==1)[agentWealthGraph setFileOutput: (BOOL) 1];
                   // to send the data also to a file

[agentWealthGraph setTitle: "Agent's wealth"];
[agentWealthGraph setAxisLabelsX: "Ticks x days."
                                Y: "Wealth."];
agentWealthGraph = [agentWealthGraph createEnd];
// that above is the old way of creating an EZgraph; now a unique
// methos exists

// pay attention: the lines below cannot be placed before createEnd

[agentWealthGraph createMinSequence: "MinWealth (all)"
                  withFeedFrom: [modelSwarm getAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (all)"
                  withFeedFrom: [modelSwarm getAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
```

```
[agentWealthGraph createMaxSequence: "MaxWealth (all)"
                  withFeedFrom: [modelSwarm getAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (r.)"
                  withFeedFrom: [modelSwarm getRandomAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (r.)"
                  withFeedFrom: [modelSwarm getRandomAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (r.)"
                  withFeedFrom: [modelSwarm getRandomAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (e.)"
                  withFeedFrom: [modelSwarm getEventAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (e.)"
                  withFeedFrom: [modelSwarm getEventAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (e.)"
                  withFeedFrom: [modelSwarm getEventAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (a.)"
                  withFeedFrom: [modelSwarm getAvatarAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (a.)"
                  withFeedFrom: [modelSwarm getAvatarAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (a.)"
                  withFeedFrom: [modelSwarm getAvatarAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (m.i.)"
                  withFeedFrom: [modelSwarm getMarketImitatingAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (m.i.)"
                  withFeedFrom: [modelSwarm getMarketImitatingAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (m.i.)"
                  withFeedFrom: [modelSwarm getMarketImitatingAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (l.i.)"
                  withFeedFrom: [modelSwarm getLocallyImitatingAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (l.i.)"
                  withFeedFrom: [modelSwarm getLocallyImitatingAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (l.i.)"
                  withFeedFrom: [modelSwarm getLocallyImitatingAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (s.l.)"
                  withFeedFrom: [modelSwarm getStopLossAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (s.l.)"
                  withFeedFrom: [modelSwarm getStopLossAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (s.l.)"
                  withFeedFrom: [modelSwarm getStopLossAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (ann)"
                  withFeedFrom: [modelSwarm getANNForecastAppAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (ann)"
                  withFeedFrom: [modelSwarm getANNForecastAppAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (ann)"
                  withFeedFrom: [modelSwarm getANNForecastAppAgentList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (bPCTA)"
                  withFeedFrom: [modelSwarm getBPCTAgentAList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (bPCTA)"
```

```
                  withFeedFrom: [modelSwarm getBPCTAgentAList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (bPCTA)"
                  withFeedFrom: [modelSwarm getBPCTAgentAList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMinSequence: "MinWealth (bPCTB)"
                  withFeedFrom: [modelSwarm getBPCTAgentBList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createAverageSequence: "MeanWealth (bPCTB)"
                  withFeedFrom: [modelSwarm getBPCTAgentBList]
                  andSelector: M(getWealthAtMeanDailyPrice)];
[agentWealthGraph createMaxSequence: "MaxWealth (bPCTB)"
                  withFeedFrom: [modelSwarm getBPCTAgentBList]
                  andSelector: M(getWealthAtMeanDailyPrice)];


//agentForge step 11d (before this point)
}
// The forecastingAgent graph

if (showForecastingAgentGraph==1)
{
forecastingAgentGraph = [EZGraph createBegin: self];
SET_WINDOW_GEOMETRY_RECORD_NAME (forecastingAgentGraph); // to allow "Save"

if(saveForecastingData==1)[forecastingAgentGraph setFileOutput: (BOOL) 1];
                                        // to send the data also to a file

[forecastingAgentGraph setTitle: "Forecasting agent log"];
[forecastingAgentGraph setAxisLabelsX:
                        "Ticks x days." Y: "Estimates and errors."];
forecastingAgentGraph = [forecastingAgentGraph createEnd];
// that above is the old way of creating an EZgraph; now a unique
// methos exists

// pay attention: the lines below cannot be placed before createEnd

[forecastingAgentGraph createSequence: "Day-1 mean p. (index)"
                  withFeedFrom: theBook
                       andSelector: M(getMeanPriceIndex)];

[forecastingAgentGraph createSequence: "Day-1 estimate (index)"
                  withFeedFrom: forecastingAgent
                       andSelector: M(getPastEstimateIndex)];

[forecastingAgentGraph createSequence: "Back propagation error"
          withFeedFrom: forecastingAgent
          andSelector: M(getBackPropagationErrorInTrainingSet)];
[forecastingAgentGraph createSequence: "Proportional error"
          withFeedFrom: forecastingAgent
          andSelector: M(getProportionalErrorInTrainingSet)];
}

// --- BPCT ---

// agent A

if([modelSwarm getBPCTAgentANumber] == 0)
                        numberOfTheBPCTAgentA_ToBeObservedDirectly=0;

// this may be 0 if we do not have any agent (above) or if we choose to
// have no display
if (numberOfTheBPCTAgentA_ToBeObservedDirectly > 0)
                               // the value comes from probe
{
// to identify the address of the chosen agent

[[modelSwarm getBPCTAgentAArrayIndex] setOffset:
                               numberOfTheBPCTAgentA_ToBeObservedDirectly-1];
agentA = [[modelSwarm getBPCTAgentAArrayIndex] get];
```

```
agentAInterface = [agentA getInterface];

 // agent A graph
 bPCTAgentA_Graph = [EZGraph createBegin: [self getZone]];
 SET_WINDOW_GEOMETRY_RECORD_NAME (bPCTAgentA_Graph); // to allow "Save"
 [bPCTAgentA_Graph setTitle: "BPCTAgent A data"];
 [bPCTAgentA_Graph setAxisLabelsX:
  "Ticks x days" Y: "Value"];
 bPCTAgentA_Graph = [bPCTAgentA_Graph createEnd];

 if(saveBPCTAgentAData==1)[bPCTAgentA_Graph setFileOutput: (BOOL) 1];
                                // to send the data also to a file

 [bPCTAgentA_Graph createSequence: "meanPrice1_A"
                                withFeedFrom: agentAInterface
                                andSelector: M(getMeanPrice1)];

 [bPCTAgentA_Graph createSequence: "liquidityQuantity_out_A"
                                withFeedFrom: agentAInterface
                                andSelector: M(getLiquidityQuantity_out)];
 [bPCTAgentA_Graph createSequence: "liquidityQuantity_target_A"
                                withFeedFrom: agentAInterface
                                andSelector: M(getLiquidityQuantity_target)];
 [bPCTAgentA_Graph createSequence: "shareQuantity_out_A"
                                withFeedFrom: agentAInterface
                                andSelector: M(getShareQuantity_out)];
 [bPCTAgentA_Graph createSequence: "shareQuantity_target_A"
                                withFeedFrom: agentAInterface
                                andSelector: M(getShareQuantity_target)];
 [bPCTAgentA_Graph createSequence: "buySell_out_A"
                                withFeedFrom: agentAInterface
                                andSelector: M(getBuySell_out)];
 [bPCTAgentA_Graph createSequence: "buySell_target_A"
                                withFeedFrom: agentAInterface
                                andSelector: M(getBuySell_target)];

}

// agent B

if([modelSwarm getBPCTAgentBNumber] == 0)
                        numberOfTheBPCTAgentB_ToBeObservedDirectly=0;

// this may be 0 if we do not have any agent (above) or if we choose to
// have no display
if (numberOfTheBPCTAgentB_ToBeObservedDirectly > 0)
                                // the value comes from probe
{
// to identify the address of the chosen agent

[[modelSwarm getBPCTAgentBArrayIndex] setOffset:
                               numberOfTheBPCTAgentB_ToBeObservedDirectly-1];
agentB = [[modelSwarm getBPCTAgentBArrayIndex] get];
agentBInterface = [agentB getInterface];

 // agent B graph
 bPCTAgentB_Graph = [EZGraph createBegin: [self getZone]];
 SET_WINDOW_GEOMETRY_RECORD_NAME (bPCTAgentB_Graph); // to allow "Save"
 [bPCTAgentB_Graph setTitle: "BPCTAgent B data"];
 [bPCTAgentB_Graph setAxisLabelsX:
  "Ticks x days" Y: "Value"];
 bPCTAgentB_Graph = [bPCTAgentB_Graph createEnd];

 if(saveBPCTAgentBData==1)[bPCTAgentB_Graph setFileOutput: (BOOL) 1];
                                // to send the data also to a file

 [bPCTAgentB_Graph createSequence: "meanPrice1_B"
                                withFeedFrom: agentBInterface
                                andSelector: M(getMeanPrice1)];
```

```
    [bPCTAgentB_Graph createSequence: "liquidityQuantity_out_B"
                                withFeedFrom: agentBInterface
                                andSelector: M(getLiquidityQuantity_out)];
    [bPCTAgentB_Graph createSequence: "liquidityQuantity_target_B"
                                withFeedFrom: agentBInterface
                                andSelector: M(getLiquidityQuantity_target)];
    [bPCTAgentB_Graph createSequence: "shareQuantity_out_B"
                                withFeedFrom: agentBInterface
                                andSelector: M(getShareQuantity_out)];
    [bPCTAgentB_Graph createSequence: "shareQuantity_target_B"
                                withFeedFrom: agentBInterface
                                andSelector: M(getShareQuantity_target)];
    [bPCTAgentB_Graph createSequence: "closingPriceWealth_out_B"
                                withFeedFrom: agentBInterface
                                andSelector: M(getClosingPriceWealth_out)];
    [bPCTAgentB_Graph createSequence: "closingPriceWealth_target_B"
                                withFeedFrom: agentBInterface
                                andSelector:
M(getClosingPriceWealth_target)];
    [bPCTAgentB_Graph createSequence: "forecastedPriceWealth_out_B"
                                withFeedFrom: agentBInterface
                                andSelector:
M(getForecastedPriceWealth_out)];
    [bPCTAgentB_Graph createSequence: "forecastedPriceWealth_target_B"
                                withFeedFrom: agentBInterface
                                andSelector:
M(getForecastedPriceWealth_target)];
   [bPCTAgentB_Graph createSequence: "buySell_out_B"
                                withFeedFrom: agentBInterface
                                andSelector: M(getBuySell_out)];
    [bPCTAgentB_Graph createSequence: "buySell_target_B"
                                withFeedFrom: agentBInterface
                                andSelector: M(getBuySell_target)];

  }


  // agentForge step 11 (before this point, add ...)

  // to check (pressing only once the next or the start button) the initial
  //settings, uncomment the following line
  //  [controlPanel setStateStopped];

  return self;
}

- buildActions
{
  [super buildActions];

  // model schedule.

  [modelSwarm buildActions];

  //agentForge step 11e
  bPCTPatternNumberInVerificationSet=0;
  if(agentA != nil)bPCTPatternNumberInVerificationSet=
                              [agentA getBPCTPatternNumberInVerificationSet];
  if(agentB != nil)bPCTPatternNumberInVerificationSet=
                              [agentB getBPCTPatternNumberInVerificationSet];
  // this variable is used below for a consistence check, as we need
  // here to have a verification set on length 1 (the same valued is used also
  // by the other BPCT agent; we have to repeat the assign operation in case
  // of missing categories of agents)

  // ActionGroup for Observer display

  displayActions = [ActionGroup create: self];

  // to update probes
```

```
  [displayActions createActionTo: probeDisplayManager message: M(update)];
  [displayActions createActionTo: actionCache message: M(doTkEvents)];

  [displayActions createActionTo: priceGraph message: M(step)];
  if (showBookGraph==1)
  [displayActions createActionTo: bookGraph  message: M(step)];
  if (showVolumesGraph==1)
  [displayActions createActionTo: volumesGraph  message: M(step)];
  if (showAgentWealthGraph==1)
  [displayActions createActionTo: agentWealthGraph  message: M(step)];
  if (showEventsGraph==1)
  [displayActions createActionTo: eventsGraph message: M(step)];
  if (showForecastingAgentGraph==1)
  [displayActions createActionTo: forecastingAgentGraph  message: M(step)];

  [displayActions createActionTo: self message: M(checkToStop)];

  // Display schedule. Note that the repeat interval is set by our
  // own Swarm data structure. The display is frequently the slowest part of a
  // simulation, when it is redraw less frequently things are faster

  displaySchedule = [Schedule createBegin: self];
  // we can use here a display frequency lower (e.g.1) than the number of
//steps
  // in the model (step number = agentNumber) to display the prices of each
  // step-tick
  [displaySchedule setRepeatInterval: displayFrequency];
  displaySchedule = [displaySchedule createEnd];

  [displaySchedule at: displayFrequency-1 createAction: displayActions];
                                              // see ## comment above

  //agentForge step 11f
  // BPCT

  if (numberOfTheBPCTAgentA_ToBeObservedDirectly > 0)
  // the following if condition is related to this CT use of the program, to
  // avoid nonsens
   if (bPCTPatternNumberInVerificationSet == -1)
     [displaySchedule at: displayFrequency-1  // see ## comment above
                        createActionTo: bPCTAgentA_Graph message: M(step)];

  if (numberOfTheBPCTAgentB_ToBeObservedDirectly > 0)
  // the following if condition is related to this CT use of the program, to
  // avoid nonsens
   if (bPCTPatternNumberInVerificationSet == -1)
     [displaySchedule at: displayFrequency-1  // see ## comment above
                        createActionTo: bPCTAgentB_Graph message: M(step)];

  return self;
}

- activateIn: swarmContext
{
// activateIn: - to activate the schedules to make them ready to run.

  [super activateIn: swarmContext];

  [modelSwarm activateIn: self];

  [displaySchedule activateIn: self];

  return [self getSwarmActivity];
}

  // to check for the stopping conditions
- checkToStop
{

  // if stopAtDayNumber is left to 0, the program will never stop
```

```
  // this stop occurs after the first tick of the time in modelSwarm,
  // but this fact does not affect the results in the observerSwarm
  // being the stop performed before the graph update

  if (stopAtDayNumber !=0 &&
      stopAtDayNumber <= [modelSwarm getCurrentDay])
  {
      printf("Stopping at day number %4d\n",
                                [modelSwarm getCurrentDay]);

      // we can restart by pressing "Start" or "Next", but the simulation
      // will run for displayFrequency ticks and then it will stop again

      [controlPanel setStateStopped];
  }

  return self;
}

@end
```

A.3 ModelSwarm.h

```
#import <objectbase/Swarm.h>
#import <simtools.h>      // necessary to invoke ObjectLoader
#import <objectbase.h>    // needed by <ProbeMap> in ModelSwarm.m
#import <activity.h>
#import <collections.h>

  // agentForge step 2

#import "AvatarAgent.h"

#import "BasicSumAgent.h"
#import "RandomAgent.h"
#import "MarketImitatingAgent.h"
#import "LocallyImitatingAgent.h"
#import "StopLossAgent.h"
#import "EventAgent.h"
#import "ANNForecastAppAgent.h"
#import "ForecastingAgent.h"
#import "CurrentAgent.h"
#import "BPCTAgentA.h"
#import "BPCTAgentAInterface.h" // related to the agents used in our
simulation
#import "BPCTAgentB.h"
#import "BPCTAgentBInterface.h" // related to the agents used in our
simulation
#import "BPCTPriceRuleMaster.h"

#import "Book.h"
#import "EventGenerator.h"
#import "EventRuleMaster.h"
#import "RandomRuleMaster.h"
#import "StopLossRuleMaster.h"
#import "SimpleANNRuleMaster.h"
#import "SimpleANNRuleMaker.h"
#import "MatrixMult.h"
#import "VectorTransFunc.h"
#import "TransFunc.h"
#import "Quota.h"
#import <random.h> // to generate ad hoc private distributions to be used
                   // in SimpleANN and BPCT, to avoid interferences with
random
                   // sequences used in determining agent behavior
#import "BPCTRuleMaster.h"
#import "BPCTRuleMaker.h"
#import "BPCTDataWarehouse.h"
```

```
@interface ModelSwarm: Swarm
{
  int bookNumber, dayNumber, maxOrderQuantity, priceVolumesHistoryLength,
     quantityVolumesHistoryLength, meanPriceHistoryLength,
      localHistoryLength, stopLossInterval, checkingIfShortOrLong, printing,
delay,
      dataWindowLength, nAheadForecasting, forecastingTrainingSetLength,
      epochNumberInEachForecastingTrainingCycle, learningProcessEveryNDays,
      cleanForecastingANNEveryMgtemNDays, eventType,
eventAgentChoiceToActBeforeOpening;
  float asymmetricBuySellProb, agentProbToActBeforeOpening,
        minCorrectingCoeff, maxCorrectingCoeff,
        asymmetricRange, floorP, agentProbToActBelowFloorP,
        maxLossRate, eventChangeProbability, eventSensibility,
        aNNInactivityRange, aNNForecastAppAgentActDailyProb;

  // agentForge step 3
  int   agentNumber, randomAgentNumber, marketImitatingAgentNumber,
avatarAgentNumber,
    locallyImitatingAgentNumber, stopLossAgentNumber, eventAgentNumber,
    aNNForecastAppAgentNumber, bPCTAgentANumber,
    bPCTAgentAEO_EP_0_Number, bPCTAgentAEO_EP_1_Number,
    bPCTAgentAEO_EP_2_Number, bPCTAgentAEO_EP_3_Number,
                              bPCTAgentBNumber,
    bPCTAgentBEO_EP_0_Number, bPCTAgentBEO_EP_1_Number,
    bPCTAgentBEO_EP_2_Number, bPCTAgentBEO_EP_3_Number;
  id <List> agentList, randomAgentList, marketImitatingAgentList,
            locallyImitatingAgentList, stopLossAgentList,
aNNForecastAppAgentList,
            bPCTAgentAList, bPCTAgentBList, avatarAgentList, eventAgentList;

  id <Array> bPCTAgentAArray;
  id <Index> bPCTAgentAArrayIndex;

  id <Array> bPCTAgentBArray;
  id <Index> bPCTAgentBArrayIndex;

  id <Array> agentArray;
  id <Index> agentArrayIndex;
  id <ListShuffler> listShuffler;

  id <ActionGroup> modelActions1, modelActions2, modelActions3;
  id <Schedule> modelSchedule;

  // agentForge step 3r

  // see above about private distributions
  id <SimpleRandomGenerator> myGenerator;
  id <UniformDoubleDist> myUniformDblRand;
  id <UniformIntegerDist> myUniformIntRand;
  id <NormalDist> myNormalDistRand;
  id <SimpleRandomGenerator> myGenerator2;
  id <UniformDoubleDist> myUniformDblRand2;
  id <UniformIntegerDist> myUniformIntRand2;
  id <SimpleRandomGenerator> myGenerator3;
  id <UniformDoubleDist> myUniformDblRand3;
  id <UniformIntegerDist> myUniformIntRand3;

  Book * theBook;
  EventGenerator * eventGenerator;
  RandomRuleMaster * randomRuleMaster;
  StopLossRuleMaster * stopLossRuleMaster;
  EventRuleMaster * eventRuleMaster;

  CurrentAgent * theCurrentAgent;
  ForecastingAgent * forecastingAgent;

  SimpleANNRuleMaster * simpleANNRuleMaster;
  SimpleANNRuleMaker * simpleANNRuleMaker;
```

```
  MatrixMult * matrixMult;
  VectorTransFunc * vectorTransFunc;
  TransFunc * transFunc;
  Quota * quota;


  // BPCT

  // agentForge step 3b
  int epochNumberInEachBPCTTrainingCycle;
  float agentAEO_EPDelta, agentBEO_EPDelta;

  // we are creating independent RuleMaster/Maker for each type of
  // BPCT agent, to have separated random distributions also in weight
  // generation
  BPCTRuleMaster * bPCTRuleMasterA, * bPCTRuleMasterB;
  BPCTRuleMaker * bPCTRuleMakerA, * bPCTRuleMakerB;
  BPCTPriceRuleMaster * bPCTPriceRuleMasterA, * bPCTPriceRuleMasterB;

  // agentForge step 3c
  int bPCTAgentAInputNodeNumber, bPCTAgentAHiddenNodeNumber,
      bPCTAgentAOutputNodeNumber,
      bPCTAgentBInputNodeNumber, bPCTAgentBHiddenNodeNumber,
      bPCTAgentBOutputNodeNumber,

      bPCTPatternNumberInVerificationSet,
      bPCTPatternNumberInTrainingSet, bPCTAgentsAreDisplayingData,
      usingRandomOrderInBPCTLearning,
      longTermLearningInBPCT_OnlyWithCompleteTrainingSet,
      useOutputsAsTargetsInBPCT_RelearningScheme;
  float bPCTWeightRange, bPCTEps, bPCTAlpha;

}


+ createBegin: aZone;
- createEnd;
- buildObjects;
- buildActions;
- activateIn: swarmContext;

- increaseCurrentDayNumber;
- (int) getCurrentDay;
- getAgentArrayIndex;
- getBPCTAgentAArrayIndex;
- getBPCTAgentBArrayIndex;
- getBook;
- getEventGenerator;

  // agentForge step 4
- (int) getBPCTAgentANumber;
- getBPCTAgentAArrayIndex;
- (int) getBPCTAgentBNumber;
- getBPCTAgentBArrayIndex;
- getAgentList;

- getAvatarAgentList;

- getRandomAgentList;
- getMarketImitatingAgentList;
- getLocallyImitatingAgentList;
- getStopLossAgentList;
- getEventAgentList;
- getANNForecastAppAgentList;
- getBPCTAgentAList;
- getBPCTAgentBList;
- getForecastingAgent;
- openProbeTo: (int) ag;

@end
```

A.4 ModelSwarm.m

```
#import "ModelSwarm.h"

@implementation ModelSwarm

+ createBegin: aZone
{
  ModelSwarm *obj;
  id <ProbeMap> probeMap;

  // in createBegin, we set up the simulation parameters

  // first, call our superclass createBegin - the return value is the
  // allocated Swarm object.

  obj = [super createBegin: aZone];

  // now fill in simulation parameters with default values.

  // agentForge step 5
  obj-> bookNumber                          =  1;
  obj-> randomAgentNumber                   =  300;
  obj-> marketImitatingAgentNumber          =  0;
  obj-> locallyImitatingAgentNumber         =  0;
  obj-> stopLossAgentNumber                 =  0;
  obj-> eventAgentNumber                    =  0;
  obj-> avatarAgentNumber                   =  0;
  obj-> aNNForecastAppAgentNumber           =  0;
  obj-> bPCTAgentAEO_EP_0_Number            =  0;
  obj-> bPCTAgentAEO_EP_1_Number            =  0;
  obj-> bPCTAgentAEO_EP_2_Number            =  0;
  obj-> bPCTAgentAEO_EP_3_Number            =  0;
  obj-> bPCTAgentBEO_EP_0_Number            =  0;
  obj-> bPCTAgentBEO_EP_1_Number            =  0;
  obj-> bPCTAgentBEO_EP_2_Number            =  0;
  obj-> bPCTAgentBEO_EP_3_Number            =  0;

  obj->bPCTAgentANumber = obj->bPCTAgentAEO_EP_0_Number + obj-
>bPCTAgentAEO_EP_1_Number
                        + obj->bPCTAgentAEO_EP_2_Number + obj-
>bPCTAgentAEO_EP_3_Number;
  obj->bPCTAgentBNumber = obj->bPCTAgentBEO_EP_0_Number + obj-
>bPCTAgentBEO_EP_1_Number
                        + obj->bPCTAgentBEO_EP_2_Number + obj-
>bPCTAgentBEO_EP_3_Number;

  obj-> agentNumber = obj-> randomAgentNumber + obj->
marketImitatingAgentNumber +
                      obj-> locallyImitatingAgentNumber + obj->
stopLossAgentNumber +
                      obj-> aNNForecastAppAgentNumber + obj-> bPCTAgentANumber
+
                      obj-> bPCTAgentBNumber + obj-> eventAgentNumber + obj->
avatarAgentNumber;

  // repeat this sum below, in BuildObjects method

  obj-> dayNumber                           =  0;
  obj-> asymmetricBuySellProb               =  0.9;
  obj-> minCorrectingCoeff                  =  0.9;
  obj-> maxCorrectingCoeff                  =  1.1;
  obj-> asymmetricRange                     =  0.0;
  obj-> agentProbToActBeforeOpening         =  0.05;
  obj-> floorP                              =  0.3;
  obj-> agentProbToActBelowFloorP           =  0.5;
  obj-> eventType                           =  1;
  obj-> eventChangeProbability              =  0.003;
  obj-> eventSensibility                    =  0.20;
```

```
  obj-> eventAgentChoiceToActBeforeOpening =  1;
  obj-> maxOrderQuantity                   =  1; // max order number per agent

  // agentForge step 6
  // used by imitating agents
  obj-> meanPriceHistoryLength             =  200;
  obj-> priceVolumesHistoryLength          =  200;
  obj-> quantityVolumesHistoryLength       =  200;
  obj-> localHistoryLength                 =  20;
  // used by stop loss agent
  obj-> maxLossRate                        =  0.10;
  obj-> stopLossInterval                   =  2;
  obj-> checkingIfShortOrLong              =  1;
  // used by forecasting agent
  obj-> dataWindowLength                   =  30;
  obj-> nAheadForecasting                  =  10;
  // used by forecasting agent
  obj-> forecastingTrainingSetLength       =  100;
  obj->
 epochNumberInEachForecastingTrainingCycle =  100;
  obj-> learningProcessEveryNDays          =  10;
  obj-> cleanForecastingANNEveryMgtemNDays =  50;
  // used by agents applying ANN forecast
  obj-> aNNInactivityRange                 =  0.02;
  obj-> aNNForecastAppAgentActDailyProb    =  0.1;
  // used by agent of BPCT type
  obj->epochNumberInEachBPCTTrainingCycle  =  100;
  obj->agentAEO_EPDelta                    =  0.1;
  obj->agentBEO_EPDelta                    =  10;

  obj-> printing                           =  0; // if 1 many objects print
                                                 // data on the terminal
                                                 // window; if 2 only
                                                 // forecastingAgent prints;
                                                 // forecastinAgent uses also
3;
                                                 // 4 is used in
BasicSumAgent;
                                                 // 5 in ANNForecastAppAgent;

  obj-> delay                              =  0; // if delay=1 the random
agent execute a for cicle to delay the execution of the simulation

  // build a customized probe map. Without a probe map, the default
  // is to show all variables and messages. Here we choose to
  // customize the format of the probe to give a nicer interface.

  probeMap = [EmptyProbeMap createBegin: aZone];
  [probeMap setProbedClass: [self class]];
  probeMap = [probeMap createEnd];

  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "randomAgentNumber"           inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "avatarAgentNumber"           inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "marketImitatingAgentNumber"  inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "locallyImitatingAgentNumber" inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "stopLossAgentNumber"         inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "eventAgentNumber"            inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "aNNForecastAppAgentNumber"   inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "bPCTAgentAEO_EP_0_Number"    inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "bPCTAgentAEO_EP_1_Number"    inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
```

```
    getProbeForVariable: "bPCTAgentAEO_EP_2_Number"    inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "bPCTAgentAEO_EP_3_Number"    inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "bPCTAgentBEO_EP_0_Number"    inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "bPCTAgentBEO_EP_1_Number"    inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "bPCTAgentBEO_EP_2_Number"    inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "bPCTAgentBEO_EP_3_Number"    inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "agentNumber"                 inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "asymmetricBuySellProb"       inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "minCorrectingCoeff"          inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "maxCorrectingCoeff"          inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "asymmetricRange"             inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "agentProbToActBeforeOpening" inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "floorP"                      inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "agentProbToActBelowFloorP"   inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "eventType"                   inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "eventChangeProbability"      inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "eventSensibility"            inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "eventAgentChoiceToActBeforeOpening"
                                                        inClass: [self
class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "maxOrderQuantity"            inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "meanPriceHistoryLength"      inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "priceVolumesHistoryLength"   inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "quantityVolumesHistoryLength"
                                                       inClass: [self
class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "localHistoryLength"          inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "maxLossRate"                 inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "stopLossInterval"            inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "checkingIfShortOrLong"       inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "dataWindowLength"            inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "nAheadForecasting"           inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable:"forecastingTrainingSetLength" inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable:"epochNumberInEachForecastingTrainingCycle"
                                                       inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "learningProcessEveryNDays"   inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
    getProbeForVariable: "cleanForecastingANNEveryMgtemNDays"
                                                       inClass: [self class]]];
   [probeMap addProbe: [probeLibrary
```

```
   getProbeForVariable: "aNNInactivityRange"          inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "aNNForecastAppAgentActDailyProb"
                                                      inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "epochNumberInEachBPCTTrainingCycle"
                                                      inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "agentAEO_EPDelta"            inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "agentBEO_EPDelta"            inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "printing"                    inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForVariable: "delay"                       inClass: [self class]]];
  [probeMap addProbe: [probeLibrary
   getProbeForMessage:  "openProbeTo:"                inClass: [self class]]];


  // Now install our custom probeMap into the probeLibrary.

  [probeLibrary setProbeMap: probeMap For: [self class]];

  return obj;
}

- createEnd
{
  return [super createEnd];
}


- buildObjects
{
  int i;
  float otherAgents;

  // agentForge step 7
  RandomAgent * anAgent1;
  MarketImitatingAgent * anAgent2;
  LocallyImitatingAgent * anAgent3;
  StopLossAgent * anAgent4;
  ANNForecastAppAgent * anAgent5;
  BPCTAgentA * anAgent6;
  BPCTAgentB * anAgent7;
  AvatarAgent * anAgent8;
  EventAgent * anAgent9;

  BPCTAgentAInterface * anInterface6;
  BPCTAgentBInterface * anInterface7;

  BPCTDataWarehouse * aDataWarehouse;

  // BPCT specific
  char * bPCTMinmaxFileNameA      ="minmaxA.data",  // mandatory, one for each
                                                    // BPCT agent type
       * bPCTMinmaxFileNameB      ="minmaxB.data",

       * bPCTInitValuesFileNameA  ="initA.val",     // this file is used only
                                                    // with internal data
                                                    // generation (CT scheme)
                                                    // but anyway the
existence
                                                    // of the file is not
                                                    // mandatory
       * bPCTInitValuesFileNameB  ="initB.val",
       * unusedFile              ="";
  // BPCT end

  // creating tools to generate ad hoc distributions to be used
```

```
  // in SimpleANN and BPCT, to avoid interferences with random
  // sequences used in determining agent behavior

  // agentForge step 7r

  unsigned int mySeed, mySeed2, mySeed3;
  // used by forecastingAgent
  mySeed = 223776;
  myGenerator = [MT19937gen create: self setStateFromSeed: mySeed];
              // MT19937gen is the generator internally used for default call
              // to Dbl and Int uniform distributions
  myUniformDblRand = [UniformDoubleDist create: self
                                              setGenerator: myGenerator];
  myUniformIntRand = [UniformIntegerDist create: self
                                              setGenerator: myGenerator];
  myNormalDistRand = [NormalDist create: self
                                                 setGenerator: myGenerator];



  // used by BPCTAgentA
  mySeed2 = 112233;
  myGenerator2 = [MT19937gen create: self setStateFromSeed: mySeed2];
              // MT19937gen is the generator internally used for default call
              // to Dbl and Int uniform distributions
  myUniformDblRand2 = [UniformDoubleDist create: self
                                              setGenerator: myGenerator2];
  myUniformIntRand2 = [UniformIntegerDist create: self
                                              setGenerator: myGenerator2];

  // used by BPCTAgentB
  mySeed3 = 333112;
  myGenerator3 = [MT19937gen create: self setStateFromSeed: mySeed3];
              // MT19937gen is the generator internally used for default call
              // to Dbl and Int uniform distributions
  myUniformDblRand3 = [UniformDoubleDist create: self
                                              setGenerator: myGenerator3];
  myUniformIntRand3 = [UniformIntegerDist create: self
                                              setGenerator: myGenerator3];

  // agentForge step 5b
  // this is an operating second definition of the followin sums (see above in
  // obj-> agentNumber assignement

  bPCTAgentANumber = bPCTAgentAEO_EP_0_Number + bPCTAgentAEO_EP_1_Number
                   + bPCTAgentAEO_EP_2_Number + bPCTAgentAEO_EP_3_Number;

  bPCTAgentBNumber = bPCTAgentBEO_EP_0_Number + bPCTAgentBEO_EP_1_Number
                   + bPCTAgentBEO_EP_2_Number + bPCTAgentBEO_EP_3_Number;

  agentNumber = randomAgentNumber + marketImitatingAgentNumber +
                locallyImitatingAgentNumber + stopLossAgentNumber +
eventAgentNumber +
                aNNForecastAppAgentNumber+bPCTAgentANumber+bPCTAgentBNumber+
                avatarAgentNumber;

  // if dataWindowLength+nAheadForecasting > meanPriceHistoryLength
  // a matrix error arises
  if(dataWindowLength+2*nAheadForecasting>meanPriceHistoryLength)
  {
    printf("The sum of dataWindowLength plus 2*nAheadForecasting is\n"
           "greater than meanPriceHistoryLength; this is a nonsense.\n"
           "See the comment 'NB about lagged values and return indexes' in\n"
           "ForecastingAgent.m\n");
    exit(0);
  }

  // randomRuleMaster
  randomRuleMaster=[RandomRuleMaster createBegin: self];
  [randomRuleMaster setAgentProbToActBeforeOpening:
```

```
                                    agentProbToActBeforeOpening];
  [randomRuleMaster setMinCorrectingCoeff: minCorrectingCoeff];
  [randomRuleMaster setMaxCorrectingCoeff: maxCorrectingCoeff];
  [randomRuleMaster setAsymmetricRange: asymmetricRange];
  [randomRuleMaster setFloorP: floorP
             andAgentProbToActBelowFloorP: agentProbToActBelowFloorP];
  randomRuleMaster=[randomRuleMaster createEnd];

  // eventRuleMaster
  eventRuleMaster=[EventRuleMaster createBegin: self];
  [eventRuleMaster setAgentProbToActBeforeOpening:
                                    agentProbToActBeforeOpening];
  if (eventAgentChoiceToActBeforeOpening!=0 &&
eventAgentChoiceToActBeforeOpening!=1)
                                      {printf ("The
eventAgentChoiceToActBeforeOpening must be 1 or 0. \n");
                                    eventAgentChoiceToActBeforeOpening=1;
                                    }
  [eventRuleMaster setEventAgentChoiceToActBeforeOpening:
                                      eventAgentChoiceToActBeforeOpening];
  [eventRuleMaster setMinCorrectingCoeff: minCorrectingCoeff];
  [eventRuleMaster setMaxCorrectingCoeff: maxCorrectingCoeff];
  [eventRuleMaster setAsymmetricRange: asymmetricRange];
  if (eventSensibility<0){printf ("The eventSensibility must be greater than
zero. \n");
                                    eventSensibility=0.20;
                                    }
  [eventRuleMaster setEventSensibility: eventSensibility];
  eventRuleMaster=[eventRuleMaster createEnd];

  // stopLossRuleMaster
  stopLossRuleMaster=[StopLossRuleMaster createBegin: self];
  [stopLossRuleMaster setAgentProbToActBeforeOpening:
                                    agentProbToActBeforeOpening];
  [stopLossRuleMaster setMinCorrectingCoeff: minCorrectingCoeff];
  [stopLossRuleMaster setMaxCorrectingCoeff: maxCorrectingCoeff];
  [stopLossRuleMaster setAsymmetricRange: asymmetricRange];
  [stopLossRuleMaster setFloorP: floorP
             andAgentProbToActBelowFloorP: agentProbToActBelowFloorP];
  stopLossRuleMaster=[stopLossRuleMaster createEnd];

  listShuffler=[ListShuffler createBegin: self];
  listShuffler=[listShuffler createEnd];

  // agentForge step 5bb
  agentList=[List create: self];
  randomAgentList=[List create: self];
  avatarAgentList=[List create: self];
  marketImitatingAgentList=[List create: self];
  locallyImitatingAgentList=[List create: self];
  eventAgentList=[List create: self];
  stopLossAgentList=[List create: self];
  aNNForecastAppAgentList=[List create: self];
  bPCTAgentAList=[List create: self];
  bPCTAgentBList=[List create: self];

  agentArray = [Array create: self];
  if (agentNumber==0) {printf("Nonsense: agentNumber cannot be 0");exit(0);}
  [agentArray setCount: agentNumber];
  agentArrayIndex = [agentArray begin: self];

  // agentForge step 5bbb

  if (bPCTAgentANumber!=0)
  {
   bPCTAgentAArray = [Array create: self];
   [bPCTAgentAArray setCount: bPCTAgentANumber];
   bPCTAgentAArrayIndex = [bPCTAgentAArray begin: self];
  }
```

```
  if (bPCTAgentBNumber!=0)
  {
   bPCTAgentBArray = [Array create: self];
   [bPCTAgentBArray setCount: bPCTAgentBNumber];
   bPCTAgentBArrayIndex = [bPCTAgentBArray begin: self];
  }
  // theBook
  theBook = [Book createBegin: self];
  [theBook setAgentArrayIndex: agentArrayIndex];
  [theBook setNAheadForecasting: nAheadForecasting];
  [theBook setAgentNumber: agentNumber];            // to build the matrixes
  [theBook setMaxOrderQuantity: maxOrderQuantity];  // "   "
    if (meanPriceHistoryLength<2){printf("The length of the history of mean "
                                         "prices cannot be <2 (internally "
                                         "set to 2).\n");
                                  meanPriceHistoryLength=2;
                                  }
  [theBook setMeanPriceHistoryLength: meanPriceHistoryLength];
    if (priceVolumesHistoryLength<2){printf("The length of the history of
Volumes "
                                         "cannot be <2 (internally "
                                         "set to 2).\n");
                                  priceVolumesHistoryLength=2;
                                  }
  [theBook setPriceVolumesHistoryLength: priceVolumesHistoryLength];
    if (quantityVolumesHistoryLength<2){printf("The length of the history of
Volumes "
                                         "cannot be <2 (internally "
                                         "set to 2).\n");
                                  quantityVolumesHistoryLength=2;
                                  }
  [theBook setQuantityVolumesHistoryLength: quantityVolumesHistoryLength];
    if (localHistoryLength<1){printf    ("The length of the local history"
                                         "cannot be <1       (internally "
                                         "set to 1).\n");
                                  localHistoryLength=1;
                                  }
  [theBook setLocalHistoryLength: localHistoryLength];
  [theBook setPrinting: printing];
  theBook = [theBook createEnd];

  //eventGenerator
  eventGenerator = [EventGenerator createBegin: self];
    if (eventType!=0 && eventType!=1){printf ("The eventType must value 0 or
1. \n");
                                  eventType=0;
                                  }
  [eventGenerator setEventType: eventType];
  [eventGenerator setStart: 0];
  [eventGenerator setMyNormalDistRand: myNormalDistRand];
    if (eventChangeProbability>1 || eventChangeProbability<0){printf ("The
eventChangeProbability must be"

       "set between 0.00 and 1.00. \n");
                                  eventChangeProbability=0.15;
                                  }
  [eventGenerator setEventChangeProbability: eventChangeProbability];
  eventGenerator = [eventGenerator createEnd];

  // a few checks
    if (asymmetricBuySellProb < 0.5){printf ("The asymmetricBuySellProb "
                                         "cannot be < 0.5 (internally "
                                         "set to 0.5).\n");
                                  asymmetricBuySellProb=0.5;
                                  }

    if (stopLossInterval > meanPriceHistoryLength)
                                  {printf ("stopLossInterval "
                                         "cannot be > meanPriceHistoryLength"
                                         "\n(internally "
```

```
                                        "set to meanPriceHistoryLength).\n");
                                    stopLossInterval=meanPriceHistoryLength;
                                    }

  // randomAgent
  for (i=1;i<=randomAgentNumber;i++)
  {
  anAgent1=[RandomAgent createBegin: self];
  [anAgent1 setNumber: i];
  [anAgent1 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent1 setBook: theBook];
  [anAgent1 setRuleMaster: randomRuleMaster];
  [anAgent1 setPrinting: printing];
  [anAgent1 setDelay: delay];
  anAgent1=[anAgent1 createEnd];

  [agentList addLast: anAgent1];
  [randomAgentList addLast: anAgent1];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent1];
  }

  // marketImitatingAgent
  for (i=randomAgentNumber+1;i<=randomAgentNumber +
marketImitatingAgentNumber;i++)
  {
  anAgent2=[MarketImitatingAgent createBegin: self];
  [anAgent2 setNumber: i];
  [anAgent2 setAsymmetricBuySellProb: asymmetricBuySellProb];
  [anAgent2 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent2 setBook: theBook];
  [anAgent2 setRuleMaster: randomRuleMaster];
  [anAgent2 setPrinting: printing];
  anAgent2=[anAgent2 createEnd];

  [agentList addLast: anAgent2];
  [marketImitatingAgentList addLast: anAgent2];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent2];
  }

  // locallyImitatingAgent
  for (i=randomAgentNumber+marketImitatingAgentNumber+1;

i<=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumber;i+
+)
  {
  anAgent3=[LocallyImitatingAgent createBegin: self];
  [anAgent3 setNumber: i];
  [anAgent3 setAsymmetricBuySellProb: asymmetricBuySellProb];
  [anAgent3 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent3 setBook: theBook];
  [anAgent3 setRuleMaster: randomRuleMaster];
  [anAgent3 setPrinting: printing];
  anAgent3=[anAgent3 createEnd];

  [agentList addLast: anAgent3];
  [locallyImitatingAgentList addLast: anAgent3];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent3];
  }

  // stopLossAgent
  for
(i=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumber+1;

i<=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumber+
```

```
        stopLossAgentNumber;i++)
  {
  anAgent4=[StopLossAgent createBegin: self];
  [anAgent4 setNumber: i];
  [anAgent4 setAsymmetricBuySellProb: asymmetricBuySellProb];
  [anAgent4 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent4 setStopLossInterval: stopLossInterval];
  [anAgent4 setMaxLossRate: maxLossRate
            andCheckingIfShortOrLong: checkingIfShortOrLong];
  [anAgent4 setBook: theBook];
  [anAgent4 setRuleMaster: stopLossRuleMaster];
  [anAgent4 setPrinting: printing];
  anAgent4=[anAgent4 createEnd];

  [agentList addLast: anAgent4];
  [stopLossAgentList addLast: anAgent4];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent4];
  }

  // we must create here a lot of objects before anAgent5, because we have to
  // pass it the address of ForecastingAgent, needing the other objects
created
  // immediately here

  // we create an instance of MatrixMult, VectorTransFunc
  // and of RuleMaster-RuleMaker

  matrixMult = [MatrixMult createBegin: self];
  matrixMult = [matrixMult createEnd];

  transFunc = [TransFunc createBegin: self];
  transFunc = [transFunc createEnd];

  vectorTransFunc = [VectorTransFunc createBegin: self];
  vectorTransFunc = [vectorTransFunc setTransFunc: transFunc];
  vectorTransFunc = [vectorTransFunc createEnd];

  // creating the simpleANNRuleMaker to evolve the ANN owned by the
  // forecastingAgent
  simpleANNRuleMaker=[SimpleANNRuleMaker createBegin: self];
  [simpleANNRuleMaker setMyUniformIntRand: myUniformIntRand];
  [simpleANNRuleMaker setMatrixMult: matrixMult];
  [simpleANNRuleMaker setVectorTransFunc: vectorTransFunc];
  simpleANNRuleMaker=[simpleANNRuleMaker createEnd];

  // creating the simpleANNRuleMaster to apply the ANN owned by the
  // forecastingAgent
  simpleANNRuleMaster=[SimpleANNRuleMaster createBegin: self];
  [simpleANNRuleMaster setSimpleANNRuleMaker: simpleANNRuleMaker];
  [simpleANNRuleMaster setMatrixMult: matrixMult];
  [simpleANNRuleMaster setVectorTransFunc: vectorTransFunc];
  simpleANNRuleMaster=[simpleANNRuleMaster createEnd];

  // creating the forecastingAgent
    if (cleanForecastingANNEveryMgtemNDays<learningProcessEveryNDays ||
        cleanForecastingANNEveryMgtemNDays%learningProcessEveryNDays !=0)
                {printf("cleanForecastingANNEveryMgtemNDays must be\n"
                        "must be greater than or equal and multiple of\n"
                        "learningProcessEveryNDays.\n");
           exit(0);
                                }
  forecastingAgent=[ForecastingAgent createBegin: self];
  [forecastingAgent setBook: theBook];
  [forecastingAgent setDataWindowLength: dataWindowLength
                andNAheadForecasting: nAheadForecasting
                andForecastingTrainingSetLength: forecastingTrainingSetLength
                andEpochNumberInEachForecastingTrainingCycle:
                   epochNumberInEachForecastingTrainingCycle
```

```
                andLearningProcessEveryNDays: learningProcessEveryNDays
                andCleanForecastingANNEveryMgtemNDays:
                                    cleanForecastingANNEveryMgtemNDays
                andForecastHistoryLength: meanPriceHistoryLength];
  [forecastingAgent setModelSwarmAddress: self];
  [forecastingAgent setSimpleANNRuleMasterAddress: simpleANNRuleMaster];
  [forecastingAgent setMyUniformDblRand: myUniformDblRand]; // ad hoc distr.
  [forecastingAgent setPrinting: printing];
  forecastingAgent=[forecastingAgent createEnd];

  quota = [Quota createBegin: self];
  [quota setForecastingAgent: forecastingAgent];
  [quota setBook: theBook];
  [quota setCleanForecastingANNEveryMgtemNDays:
                               cleanForecastingANNEveryMgtemNDays];
  quota = [quota createEnd];


  // aNNForecastAppAgent
  for
(i=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumber+
         stopLossAgentNumber+1;

i<=randomAgentNumber+marketImitatingAgentNumber+locallyImitatingAgentNumber+
          stopLossAgentNumber+aNNForecastAppAgentNumber;i++)
  {
  anAgent5=[ANNForecastAppAgent createBegin: self];
  [anAgent5 setNumber: i];
  [anAgent5 setAsymmetricBuySellProb: asymmetricBuySellProb];
  [anAgent5 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent5 setBook: theBook];
  [anAgent5 setANNInactivityRange: aNNInactivityRange
            andANNForecastAppAgentActDailyProb:
aNNForecastAppAgentActDailyProb];
  [anAgent5 setMyUniformDblRand: myUniformDblRand]; // ad hoc distr.
  [anAgent5 setRuleMaster: randomRuleMaster];
  [anAgent5 setForecastingAgent: forecastingAgent];
  [anAgent5 setPrinting: printing];
  anAgent5=[anAgent5 createEnd];

  [agentList addLast: anAgent5];
  [aNNForecastAppAgentList addLast: anAgent5];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent5];
  }

  // BPCT

  // agentForge step 7b
      bPCTAgentAInputNodeNumber  = 7;
      bPCTAgentAHiddenNodeNumber = 5;
      bPCTAgentAOutputNodeNumber = 3;

      bPCTAgentBInputNodeNumber  = 8;
      bPCTAgentBHiddenNodeNumber = 6;
      bPCTAgentBOutputNodeNumber = 5;

      bPCTPatternNumberInVerificationSet = -1;
      bPCTPatternNumberInTrainingSet     = -10;
      bPCTAgentsAreDisplayingData        =  0;
      usingRandomOrderInBPCTLearning     =  1;
      longTermLearningInBPCT_OnlyWithCompleteTrainingSet = 1;
      useOutputsAsTargetsInBPCT_RelearningScheme = 0;
      bPCTWeightRange = 0.3;
      bPCTEps          = 0.6;
      bPCTAlpha        = 0.9;

  [ObjectLoader load: self fromFileNamed: "bp.setup"];
```

```
  otherAgents=randomAgentNumber+marketImitatingAgentNumber+
              locallyImitatingAgentNumber+
              stopLossAgentNumber+aNNForecastAppAgentNumber;

  // bPCTAgent A
  // the various rule Master/Maker are private, to keep independent the random
  // distributions
  bPCTRuleMakerA = [BPCTRuleMaker createBegin: self];
  [bPCTRuleMakerA setMyUniformIntRand: myUniformIntRand2]; // ad hoc distr.
  [bPCTRuleMakerA setMatrixMult: matrixMult];
  [bPCTRuleMakerA setVectorTransFunc: vectorTransFunc];
  bPCTRuleMakerA = [bPCTRuleMakerA createEnd];

  bPCTRuleMasterA = [BPCTRuleMaster createBegin: self];
  [bPCTRuleMasterA setRuleMaker: bPCTRuleMakerA];
  [bPCTRuleMasterA setMatrixMult: matrixMult];
  [bPCTRuleMasterA setVectorTransFunc: vectorTransFunc];
  bPCTRuleMasterA = [bPCTRuleMasterA createEnd];

  bPCTPriceRuleMasterA = [BPCTPriceRuleMaster createBegin: self];
  [bPCTPriceRuleMasterA setMyUniformDblRand: myUniformDblRand2]; // ad hoc
distr.
  [bPCTPriceRuleMasterA setAgentProbToActBeforeOpening:
                                  agentProbToActBeforeOpening];
  [bPCTPriceRuleMasterA setMinCorrectingCoeff: minCorrectingCoeff];
  [bPCTPriceRuleMasterA setMaxCorrectingCoeff: maxCorrectingCoeff];
  [bPCTPriceRuleMasterA setAsymmetricRange: asymmetricRange];
  bPCTPriceRuleMasterA = [bPCTPriceRuleMasterA createEnd];

  for (i=otherAgents + 1;
       i<=otherAgents + bPCTAgentANumber;i++)
  {
  // first we create the datawarehouse where BPCTagent technical data are
stored
  aDataWarehouse= [BPCTDataWarehouse createBegin: self];
  [aDataWarehouse setMinmaxRowToBeModifiedFromInt: 9
                          usingGenericIntVariableAddress: &maxOrderQuantity];
  [aDataWarehouse setVerificationFileName: unusedFile // never used here
                  andTrainingFileName:     unusedFile // never used here
                  andMinmaxName: bPCTMinmaxFileNameA
                  andInitValuesFileName: bPCTInitValuesFileNameA];
  [aDataWarehouse setInputNodeNumber:  bPCTAgentAInputNodeNumber
                  andHiddenNodeNumber: bPCTAgentAHiddenNodeNumber
                  andOutputNodeNumber: bPCTAgentAOutputNodeNumber
            andPatternNumberInVerificationSet:
bPCTPatternNumberInVerificationSet
            andPatternNumberInTrainingSet:     bPCTPatternNumberInTrainingSet
            andEpochNumberInEachTrainingCycle:
                                     epochNumberInEachBPCTTrainingCycle];
  [aDataWarehouse setBackPropagationParametersWeightRange: bPCTWeightRange
                  eps: bPCTEps alpha: bPCTAlpha
                  andWithOrderInLearning: usingRandomOrderInBPCTLearning
                  andLongTermLearningInCT:
                  longTermLearningInBPCT_OnlyWithCompleteTrainingSet
                  andUseOutputsAsTargetsInCT:
                  useOutputsAsTargetsInBPCT_RelearningScheme];
  [aDataWarehouse setMyUniformDblRand: myUniformDblRand2]; // ad hoc dist.

  aDataWarehouse=[aDataWarehouse createEnd];

  // then we create an interface for our agent, to simplifly its links
  // with the observer, if any, but mainly as a help in CT building

  anInterface6 = [BPCTAgentAInterface createBegin: self];
  if (i<=otherAgents+bPCTAgentAEO_EP_0_Number+bPCTAgentAEO_EP_1_Number
                   +bPCTAgentAEO_EP_2_Number+bPCTAgentAEO_EP_3_Number)
    [anInterface6 setUseEO_EP: 3]; // EO_EP 3
  if (i<=otherAgents+bPCTAgentAEO_EP_0_Number+bPCTAgentAEO_EP_1_Number
                   +bPCTAgentAEO_EP_2_Number)
    [anInterface6 setUseEO_EP: 2]; // EO_EP 2
```

```
  if (i<=otherAgents+bPCTAgentAEO_EP_0_Number+bPCTAgentAEO_EP_1_Number)
    [anInterface6 setUseEO_EP: 1]; // EO_EP 1
  if (i<=otherAgents+bPCTAgentAEO_EP_0_Number)
    [anInterface6 setUseEO_EP: 0]; // no EO_EP use
  [anInterface6 setEO_EPDelta: agentAEO_EPDelta];
  [anInterface6 setMyUniformIntRand: myUniformIntRand2]; // ad hoc distr.
  [anInterface6 setAgentNumber: i];
  [anInterface6 setDataWarehouse: aDataWarehouse];
  [anInterface6 setBook: theBook];
  anInterface6 = [anInterface6 createEnd];
  [anInterface6 initialize]; // NB. after createEnd

  anAgent6 = [BPCTAgentA createBegin: self];
  [anAgent6 setNumber: i andSetReadWeightsFromFile: 0]; // it never reads
weights
                                                        // from a file
  [anAgent6 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent6 setDataWarehouse: aDataWarehouse];
  [anAgent6 setInterface: anInterface6];         // double declaration for the
parent
  [anAgent6 setSpecificInterface: anInterface6]; // and for the inheriting
class
  [anAgent6 setRuleMaster: bPCTRuleMasterA];
  [anAgent6 setPriceRuleMaster: bPCTPriceRuleMasterA];
  [anAgent6 setDisplayDataWhileRunning: bPCTAgentsAreDisplayingData];
  [anAgent6 setBook: theBook]; // used by accounting method act2
  [anAgent6 setPrinting: printing];

  anAgent6 = [anAgent6 createEnd];


  [agentList addLast: anAgent6];
  [bPCTAgentAList addLast: anAgent6];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent6];

  [bPCTAgentAArrayIndex next];
  [bPCTAgentAArrayIndex put: anAgent6];

  }

  otherAgents+=bPCTAgentANumber;

  // bPCTAgent B
  // the various rule Master/Maker are private, to keep independent the random
  // distributions
  bPCTRuleMakerB = [BPCTRuleMaker createBegin: self];
  [bPCTRuleMakerB setMyUniformIntRand: myUniformIntRand3]; // ad hoc distr.
  [bPCTRuleMakerB setMatrixMult: matrixMult];
  [bPCTRuleMakerB setVectorTransFunc: vectorTransFunc];
  bPCTRuleMakerB = [bPCTRuleMakerB createEnd];

  bPCTRuleMasterB = [BPCTRuleMaster createBegin: self];
  [bPCTRuleMasterB setRuleMaker: bPCTRuleMakerB];
  [bPCTRuleMasterB setMatrixMult: matrixMult];
  [bPCTRuleMasterB setVectorTransFunc: vectorTransFunc];
  bPCTRuleMasterB = [bPCTRuleMasterB createEnd];

  bPCTPriceRuleMasterB = [BPCTPriceRuleMaster createBegin: self];
  [bPCTPriceRuleMasterB setMyUniformDblRand: myUniformDblRand3]; // ad hoc
distr.
  [bPCTPriceRuleMasterB setAgentProbToActBeforeOpening:
                                agentProbToActBeforeOpening];
  [bPCTPriceRuleMasterB setMinCorrectingCoeff: minCorrectingCoeff];
  [bPCTPriceRuleMasterB setMaxCorrectingCoeff: maxCorrectingCoeff];
  [bPCTPriceRuleMasterB setAsymmetricRange: asymmetricRange];
  bPCTPriceRuleMasterB = [bPCTPriceRuleMasterB createEnd];

  for (i=otherAgents + 1;
```

```
       i<=otherAgents + bPCTAgentBNumber;i++)
  {
  // first we create the datawarehouse where BPCTagent technical data are
stored
  aDataWarehouse= [BPCTDataWarehouse createBegin: self];
  [aDataWarehouse setMinmaxRowToBeModifiedFromInt: 12
                          usingGenericIntVariableAddress: &maxOrderQuantity];
  [aDataWarehouse setVerificationFileName: unusedFile // never used here
                  andTrainingFileName:     unusedFile // never used here
                  andMinmaxName: bPCTMinmaxFileNameB
                  andInitValuesFileName: bPCTInitValuesFileNameB];
  [aDataWarehouse setInputNodeNumber:  bPCTAgentBInputNodeNumber
                  andHiddenNodeNumber: bPCTAgentBHiddenNodeNumber
                  andOutputNodeNumber: bPCTAgentBOutputNodeNumber
           andPatternNumberInVerificationSet:
bPCTPatternNumberInVerificationSet
           andPatternNumberInTrainingSet:     bPCTPatternNumberInTrainingSet
           andEpochNumberInEachTrainingCycle:
                                      epochNumberInEachBPCTTrainingCycle];
  [aDataWarehouse setBackPropagationParametersWeightRange: bPCTWeightRange
                  eps: bPCTEps alpha: bPCTAlpha
                  andWithOrderInLearning: usingRandomOrderInBPCTLearning
                  andLongTermLearningInCT:
                  longTermLearningInBPCT_OnlyWithCompleteTrainingSet
                  andUseOutputsAsTargetsInCT:
                  useOutputsAsTargetsInBPCT_RelearningScheme];
  [aDataWarehouse setMyUniformDblRand: myUniformDblRand3]; // ad hoc dist.

  aDataWarehouse=[aDataWarehouse createEnd];

  // then we create an interface for our agent, to simplifly its links
  // with the observer, if any, but mainly as a help in CT building

  anInterface7 = [BPCTAgentBInterface createBegin: self];
  if (i<=otherAgents+bPCTAgentBEO_EP_0_Number+bPCTAgentBEO_EP_1_Number
                   +bPCTAgentBEO_EP_2_Number+bPCTAgentBEO_EP_3_Number)
    [anInterface7 setUseEO_EP: 3]; // EO_EP 3
  if (i<=otherAgents+bPCTAgentBEO_EP_0_Number+bPCTAgentBEO_EP_1_Number
                   +bPCTAgentBEO_EP_2_Number)
    [anInterface7 setUseEO_EP: 2]; // EO_EP 2
  if (i<=otherAgents+bPCTAgentBEO_EP_0_Number+bPCTAgentBEO_EP_1_Number)
    [anInterface7 setUseEO_EP: 1]; // EO_EP 1
  if (i<=otherAgents+bPCTAgentBEO_EP_0_Number)
    [anInterface7 setUseEO_EP: 0]; // no EO_EP use
  [anInterface7 setEO_EPDelta: agentBEO_EPDelta];
  [anInterface7 setMyUniformIntRand: myUniformIntRand3]; // ad hoc distr.
  [anInterface7 setAgentNumber: i];
  [anInterface7 setDataWarehouse: aDataWarehouse];
  [anInterface7 setBook: theBook];
  [anInterface7 setForecastingAgent: forecastingAgent];
  anInterface7 = [anInterface7 createEnd];
  [anInterface7 initialize]; // NB. after createEnd

  anAgent7 = [BPCTAgentB createBegin: self];
  [anAgent7 setNumber: i andSetReadWeightsFromFile: 0]; // it never reads
weights
                                                        // from a file
  [anAgent7 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent7 setDataWarehouse: aDataWarehouse];
  [anAgent7 setInterface: anInterface7];         // double declaration for the
parent
  [anAgent7 setSpecificInterface: anInterface7]; // and for the inheriting
class
  [anAgent7 setRuleMaster: bPCTRuleMasterB];
  [anAgent7 setPriceRuleMaster: bPCTPriceRuleMasterB];
  [anAgent7 setDisplayDataWhileRunning: bPCTAgentsAreDisplayingData];
  [anAgent7 setBook: theBook]; // used by accounting method act2
  [anAgent7 setPrinting: printing];

  anAgent7 = [anAgent7 createEnd];
```

```
  [agentList addLast: anAgent7];
  [bPCTAgentBList addLast: anAgent7];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent7];

  [bPCTAgentBArrayIndex next];
  [bPCTAgentBArrayIndex put: anAgent7];

  }

  // agentForge step 8
  // AvatarAgent
  otherAgents = otherAgents + bPCTAgentBNumber;

  for (i=otherAgents + 1;
     i<=otherAgents + avatarAgentNumber;i++)
  {

  anAgent8 = [AvatarAgent createBegin: self];
  [anAgent8 setNumber: i];
  [anAgent8 setBook: theBook]; // used by accounting method act2
  [anAgent8 setPrinting: printing];
  [anAgent8 setMaxOrderQuantity: maxOrderQuantity];
  anAgent8=[anAgent8 createEnd];

  [agentList addLast: anAgent8];
  [avatarAgentList addLast: anAgent8];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent8];

  }

 // eventAgent
  otherAgents = otherAgents + avatarAgentNumber;
  for (i=otherAgents+1;i<=otherAgents+eventAgentNumber;i++)
  {
  anAgent9=[EventAgent createBegin: self];
  [anAgent9 setNumber: i];
  [anAgent9 setMaxOrderQuantity: maxOrderQuantity];
  [anAgent9 setBook: theBook];
  [anAgent9 setEventGenerator: eventGenerator];
  [anAgent9 setRuleMaster: eventRuleMaster];
  [anAgent9 setPrinting: printing];
  anAgent9=[anAgent9 createEnd];

  [agentList addLast: anAgent9];
  [eventAgentList addLast: anAgent9];

  [agentArrayIndex next];
  [agentArrayIndex put: anAgent9];
  }

  // current agent
  // see the comment in CurrentAgent.m to understand this trick
  theCurrentAgent=[CurrentAgent createBegin: self];
  [theCurrentAgent setAgentList: agentList];
  theCurrentAgent=[theCurrentAgent createEnd];

  return self;
}

- buildActions
{
  int i;
  // we create the list of simulation actions
```

```
  modelActions1 = [ActionGroup create: self];
  [modelActions1 createActionTo: self message: M(increaseCurrentDayNumber)];
  [modelActions1 createActionTo: theBook message: M(setClean)];
  [modelActions1 createActionTo: forecastingAgent message: M(step)];
  [modelActions1 createActionTo: quota message: M(step)];
  // this shuffling action is not relevant if agent are equal instances of
  // the same class, but it is useful to avoid biased situations when agents
  // are not homegeneous
  [modelActions1 createActionTo: listShuffler
                             message: M(shuffleWholeList:) : agentList];
  // acting before opening
  [modelActions1 createActionTo: eventGenerator message: M(setEventState)];
  [modelActions1 createActionForEach: agentList message: M(act0)];

  // acting in the market
  modelActions2 = [ActionGroup create: self];
  [modelActions2 createActionTo: eventGenerator message: M(setEventState)];
  [modelActions2 createActionTo: theCurrentAgent message: M(act1)];
  if (avatarAgentNumber>0)[modelActions2 createActionForEach: avatarAgentList
message: M(act1)];


  // accounting ...
  modelActions3 = [ActionGroup create: self];
  [modelActions3 createActionTo: theBook message: M(setMeanPrice)];
  [modelActions3 createActionForEach: agentList message: M(act2)];

  // then we create a schedule that executes the modelActions.

  modelSchedule = [Schedule createBegin: self];
  // we use here agentNumber steps in each cycle, while the observer uses
  // a low display frequency (e.g. 1) to show the price of each step-tick
  // we can also use a high d.f. (e.g.1000 with 100 agents) to run a faster
  // simulation
  [modelSchedule setRepeatInterval: agentNumber];
  modelSchedule = [modelSchedule createEnd];

  [modelSchedule at: 0 createAction: modelActions1];
  for (i=0;i<agentNumber;i++){
       [modelSchedule at: i createAction: modelActions2];
  }
  [modelSchedule at: agentNumber-1 createAction: modelActions3];


  return self;
}

- activateIn: swarmContext
{
  // here, we activate the swarm in the context passed in
  // then we activate our schedule in ourselves

  [super activateIn: swarmContext];

  [modelSchedule activateIn: self];


  return [self getSwarmActivity];
}

- increaseCurrentDayNumber
{
  dayNumber++;
  if(printing==1)printf("Day number #%5d\n",dayNumber);
  //New price
                     FILE * pFile;
                     pFile = fopen ("Day.dat","w");
                     if (pFile == NULL) {
                       perror("cannot open output file");
                    exit(1);
```

```
                }
                        fprintf(pFile,"%5d\n", dayNumber);
                fclose(pFile);
  //New end
  return self;
}

- (int) getCurrentDay
{
  return dayNumber;
}

  // agentForge step 9
- (int) getBPCTAgentANumber{return bPCTAgentANumber;}
- (int) getBPCTAgentBNumber{return bPCTAgentBNumber;}
- getAgentList{return agentList;}

- getAvatarAgentList{return avatarAgentList;}

- getRandomAgentList{return randomAgentList;}
- getMarketImitatingAgentList{return marketImitatingAgentList;}
- getLocallyImitatingAgentList{return locallyImitatingAgentList;}
- getStopLossAgentList{return stopLossAgentList;}
- getEventAgentList{return eventAgentList;}
- getANNForecastAppAgentList{return aNNForecastAppAgentList;}
- getBPCTAgentAList{return bPCTAgentAList;}
- getBPCTAgentBList{return bPCTAgentBList;}

- getAgentArrayIndex{return agentArrayIndex;}
- getBPCTAgentAArrayIndex{return bPCTAgentAArrayIndex;}
- getBPCTAgentBArrayIndex{return bPCTAgentBArrayIndex;}


- getForecastingAgent
{
  return forecastingAgent;
}

- getBook
{
  return theBook;
}

- getEventGenerator
{
  return eventGenerator;
}

- openProbeTo: (int) n
{
  BasicSumAgent * anAgent;

  if (n<1 || n>agentNumber) return self;

  [agentArrayIndex setOffset: n-1];
  anAgent=[agentArrayIndex get];
  [anAgent getProbe];

  return anAgent;
}


@end
```

A.5 EventGenerator.h

```
#import <objectbase/SwarmObject.h>
#import <random.h>
```

```
#import <simtools.h>      // necessary to invoke ObjectLoader -> Serve?
#import <simtoolsgui/GUISwarm.h> // to use CREATE_ARCHIVED_PROBE_DISPLAY
#import <simtoolsgui.h>          // ""

@interface EventGenerator: SwarmObject
{

  int eventType, changeState, changeEvent;

  float eventChangeProbability, eventState;

  id <NormalDist> myNormalDistRand;

}

- createEnd;

// set the event type, the probability that an event can change its state
(active or inactive) and its sensibility
- setStart: (int) s;
- setMyNormalDistRand: u;
- setEventType: (int) et;
- setEventChangeProbability: (float) ecp;

// this method is used to put in working the event generator: it returns the
state and hence
// the intensity of the event
- setEventState;
- (float) getEventState;

@end
```

A.6 EventGenerator.m

```
#import "EventGenerator.h"
#import "ModelSwarm.h"

@implementation EventGenerator

-setStart: (int) s
{
  eventState = s;
  return self;
}

-setMyNormalDistRand: u
{
  myNormalDistRand = u;
  return self;
}

-setEventChangeProbability: (float) ecp
{
  eventChangeProbability = ecp;
  return self;
}

- setEventType: (int) et
{
  eventType = et;
  return self;
}

- (float) getEventState
{
  return eventState;
}
```

```
- setEventState
{
  //It evaluates the possibility to change state and event
  changeState = 0;
  if ((float) [uniformDblRand getDoubleWithMin: 0.0 withMax: 1.0] <
eventChangeProbability)
      changeState = 1;

  changeEvent = 0;
  if ((float) [uniformDblRand getDoubleWithMin: -1 withMax: +1] > 0)
      changeEvent = 1;

  //Analyze all case
  if (changeState==1)
  {
       if (eventState!=0)
       {
              if (changeEvent==1)
              {
                  if (eventType==1)
                  {
                    //evaluate the intensity of the variable new event
                    eventState = [myNormalDistRand getSampleWithMean: 0.0
withVariance: 0.15];
                  }
                  else
                  {
                    //evaluate the intensity of the fixed new event
                    eventState = [uniformIntRand getIntegerWithMin: -1 withMax:
+1];
                  }
              }
              else
              {
                     eventState=0;
              }
       }
       else
       {
           if (eventType==1)
           {
             //evaluate the intensity of the variable new event
             eventState = [myNormalDistRand getSampleWithMean: 0.0
withVariance: 0.15];
           }
           else
           {
             //evaluate the intensity of the fixed new event
             eventState = [uniformIntRand getIntegerWithMin: -1 withMax: +1];
           }

       }
  }

  return self;
}

- createEnd
{
   [super createEnd];
   return self;
}

@end
```

A.7 EventRuleMaster.h

```
// This is the EventRuleMaster, guiding the behavior of EventAgent

#import "BasicSumRuleMaster.h"

@interface EventRuleMaster: BasicSumRuleMaster
{

float price, eventCoefficient;
int choice;

}

- (float) getPriceBeforeOpening: (float) lp
                                                withEventState: (float) es
                                                withBuySellP: (float) p;

- (float) getPrice: (float) lp
                                                withEventState: (float) es
                                                withBuySellP: (float) p;

@end
```

A.8 EventRuleMaster.m

```
#import "EventRuleMaster.h"

@implementation EventRuleMaster


- (float) getPriceBeforeOpening: (float) lastPrice withEventState: (float)
eventState withBuySellP: (float) p
{
  //Choice to act before opening
  if (eventAgentChoiceToActBeforeOpening == 1)
  {
       if ((float) eventState>1 || (float) eventState<-1)
       {
             eventCoefficient=1;
       }
       if ((float) eventState<=1 && (float) eventState>=0)
       {
             eventCoefficient=eventState;
       }
       if ((float) eventState>=(-1) && (float) eventState<0)
       {
             eventCoefficient=-eventState;
       }

       if (eventCoefficient < (float) [uniformDblRand getDoubleWithMin: 0.0
withMax: 1.0])
       return 0.0;
       else return [self getPrice: lastPrice withEventState: eventState
withBuySellP: p];
  }
  //Due to act before opening if eventState different from zero
  else
  {
       if (eventState == 0)
       return 0.0;
       else return [self getPrice: lastPrice withEventState: eventState
withBuySellP: p];
  }
}
```

```
- (float) getPrice: (float) lastPrice withEventState: (float) eventState
withBuySellP: (float) p
{
  //positive event -> buyer
  if ((float) eventState>0)
  {
     price = lastPrice * (float)
              [uniformDblRand getDoubleWithMin: 1
                                  withMax: 1 + (eventState*eventSensibility)];

     choice=1;
  }

  //negative event -> seller
  if ((float) eventState<0)
  {
     price = lastPrice * (float)
             [uniformDblRand getDoubleWithMin: 1 +
(eventState*eventSensibility)
                                       withMax: 1];

     choice=-1;
  }

  //no event active -> random
  if ((float) eventState==0)
  {
    price = lastPrice * (float)
        [uniformDblRand getDoubleWithMin: minCorrectingCoeff + asymmetricRange
                                  withMax: maxCorrectingCoeff +
asymmetricRange];

    choice=1; // buy with probability p
    if ((float) [uniformDblRand getDoubleWithMin: 0.0 withMax: 1.0] > p)
    choice=-1;// sell
  }

  return choice*price;

}

@end
```

A.9 EventAgent.h

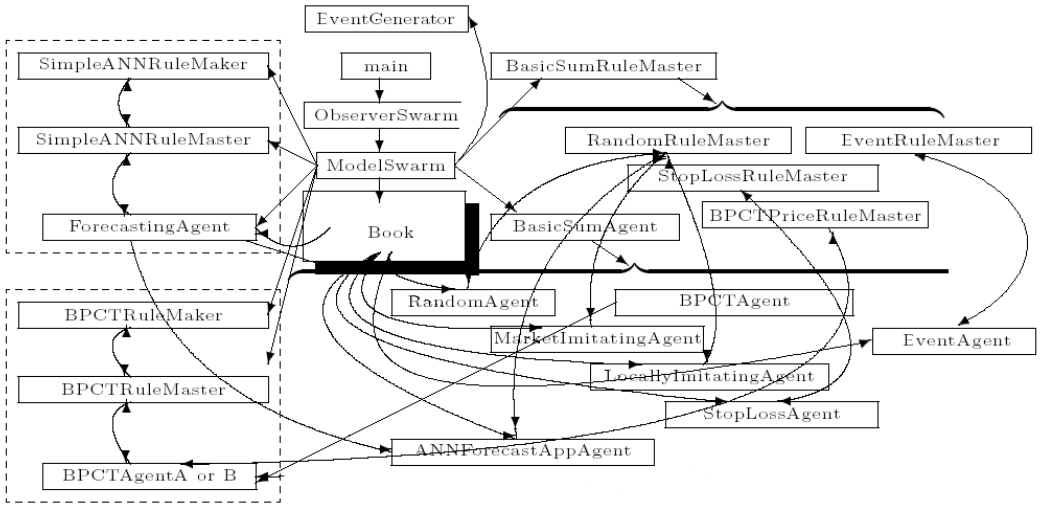
```
// EventAgent inherits from BasicSumAgent and operates when an event is
// generated by Event generator; if the event is good, he goes on the
// market for buying at a high price else, if the event is bad, he goes
// on the market for selling at a lower price.

#import "BasicSumAgent.h"
#import "EventRuleMaster.h"

@interface EventAgent: BasicSumAgent
{
  EventRuleMaster * ruleMaster;
}

- setRuleMaster: r;

- act0; // acting - placing buy or sell orders - before opening
- act1;  // placing buy or sell orders

@end
```

A.10 EventAgent.m

```
#import "EventAgent.h"

@implementation EventAgent

  // our ruleMaster
- setRuleMaster: r
{
  ruleMaster=r;
  return self;
}

  // Acting before opening
- act0
{
   float lastPrice, lastEvent;
   int i;

     // clearing executedPrices
        executedPriceCount=0;
        for (i=0;i<maxOrderQuantity;i++) [executedPrices R:i C:0 setFrom:0];

   actingBeforeOpening=0;

   buySellSwitch=0.5; // the standard random choice situation

   lastPrice=[theBook getPrice];
   lastEvent=[eventGenerator getEventState];


   // Asking the ruleMaster for a position (if eventState is > 0 we are
   // buyers; if it is <0 we are sellers; if it is =0 we act like randomAgent
   price=[ruleMaster getPriceBeforeOpening: lastPrice
                    withEventState: lastEvent
                      withBuySellP: buySellSwitch];

   if (price!=0) actingBeforeOpening=1;

   // how many orders?
   if (maxOrderQuantity==1) iMax=1;
    else iMax=[uniformIntRand getIntegerWithMin: 1 withMax: maxOrderQuantity];
   for (i=1;i<=iMax;i++)
   //if price==0 theBook is not recording it
   [theBook setOrderBeforeOpeningFromAgent: number atPrice: price];

   return self;
}

  // acting in the market
- act1
{
   float lastPrice, lastEvent;
   int i;

   buySellSwitch=0.5; // the standard random choice situation

   // the agent has operated before opening
   if (actingBeforeOpening==1) return self;

   lastPrice=[theBook getPrice];
   lastEvent=[eventGenerator getEventState];

   price=[ruleMaster getPrice: lastPrice
                    withEventState: lastEvent
                     withBuySellP: buySellSwitch];


   // how many orders?
```

```
    if (maxOrderQuantity==1) iMax=1;
     else iMax=[uniformIntRand getIntegerWithMin: 1 withMax: maxOrderQuantity];
    for (i=1;i<=iMax;i++)
    [theBook setOrderFromAgent: number atPrice: price];

    return self;
}

@end
```

B. Schema di SUM



Riferimenti Bibliografici

Pia, A. (1997), *Il Mercato Azionario Italiano*, Torino, Giappichelli Editore

Bodie, Kane, Markus (1999), *Investments*, Singapore, McGraw-Hill

Cifarelli D. Michele (1995), *Elementi di calcolo delle probabilità*, Torino, Giappichelli Editore

Jalla E. (1977), *Principi di statistica teorica*, Torino, Giappichelli Editore

Terna, P. (1996), *Economia e simulazione: una rivoluzione nel metodo* da *Sistemi intelligenti* Anno VIII, numero 3, dicembre 1996, Torino.

The bulletin ot the santa fe institute: summer 1996 (1996), *Economics and the modern theories of cognitive bahavior*, Santa Fe

Minar N., Burkhart R., Langton C., Askenazi M. (June 21, 1996), *The swarm simulation system: a toolkit for building multi-agent simulations*, Santa Fe

Lin F.R., Tan G. W., Shaw M. J. (October 31, 1996), *Multi-Agent enterprise modelling*, Working paper number 96-0134, Department of Business Administration, University of Illinois

Kilpatrick H. E. (2001), *Complexity, Spontaneous Order, and Friedrich Hayek: Are Spontaneous Order and Complexity Essentially the Same Thing?*, Vol. 6 No. 4, John Wiley & Sons Inc.

Unknown (1995), *Rational Economic Man – The Human Factor*, The Economist December 24th 1994 – January 6th 1995

Hahn F. (1994), *Una retrospettiva intellettuale*, Moneta e Credito No. 188, Dicembre 1994

Rosser Jr. J. B. (1999), *On the Complexities of Complex Economic Dynamics*, Journal of Economic Perspectives – Vol. 13 No. 4 Pages 169 – 192

Terna P. (1998), *Simulation Tools for Social Scientists: Building Agent Based Models with Swarm,* Journal of Artificial Societies and Social Simulation, Vol. 1 No. 2

Searle J. R. (1990), *La Mente è un programma? – Intelligenza Artificiale: un dibattito*, Le Scienze No. 259 – Marzo 1990

Terna P. (1994), *Reti Neurali e Modelli con Agenti Adattivi*, Dipartimento di Scienze Economiche e Finanziarie G. Prato – Facoltà di Economia, Torino

Axtell R. L., Epstein J. M. (1994), *Agent-Based Modelling: Understanding Our Creations*, The Bulletin of Santa Fe Institute: Winter 1994

Terna P. (1995), *Economic Experiments with Swarm: A Neural Network Approach to the Self-Development of Consistency in Agents' Behavior* da Economic Simulation in Swarm: Agent-Based Modelling and Object Oriented Programming – Chapter 3 – Vol. 14, Venezia

Terna P. (1995), *Cognitive Agents Behaving in a Simple Stock Market Structure* da Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming – Chapter 8 – Vol. 14

Gilbert N., Terna P. (1999), *How to Build and Use Agent-Based Models in Social Science*, Mind & Society, 1, 2000, Vol. 1, pp. 57 – 72, Fondazione Rosselli

Borsa Italiana S.p.A (2003), *Regolamento dei Mercati Organizzati e Gestiti dalla Borsa Italiana*, http://www.borsaitaliana.it

Axelrod R. (1997), *Advancing the Art of Simulation in the Social Sciences*, Conte R, Hegselmann R. & Terna P., Simulating Social Phenomena, Berlin.

Luna F., Stefansson B. (2000), *Economic Simulation in Swarm: Agent-Based Modelling and Object Oriented Programming*, Kluwer Academic Publishers, Boston/London.

Parisi D. (2001), *Simulazioni – La realtà rifatta a computer*, Il Mulino, Bologna.

Larking D., Wilson G. (1995), *Object-Oriented Programming and the Objective-C Language*, Next Developer's Library, Redwood City.