INTRODUZIONE

Questa tesi affronta il tema della presenza o no di comportamenti altruistici o cooperativi negli individui, attraverso l'analisi della letteratura in materia e l'utilizzo di un esperimento.

Nel primo capitolo illustreremo le potenzialità del metodo sperimentale per lo studio dell'economia. Inizieremo con una breve storia dell'economia sperimentale per comprenderne le radici, passando in rassegna i primi pionieri, le difficoltà riscontrate nel far accettare un nuovo metodo di studio, i traguardi raggiunti ed i riconoscimenti ottenuti. Verranno elencati i pregi ed i difetti della sperimentazione per averne una visione d'insieme e poterne valutare la reale utilità.

L'uso dei computer per testare teorie, siano esse biologiche od economiche, rappresenta una nuova frontiera. In biologia la creazione della vita *in vitro*, pur essendo un'impresa scientifica degna di nota, alla lunga non ci direbbe sullo spazio della vita possibile molto di più di quello che già sappiamo.

I computer invece offrono un ambiente alternativo in cui provare a "sintetizzare" la vita. Le tecniche di simulazione infatti hanno utilizzato le potenzialità dell'hardware per la creazione della vita e di una realtà *in silicio*. In questo modo è possibile riprodurre condizioni particolari per verificare o ideare nuove congetture.

Lo scienziato normalmente elabora nella sua mente una teoria, che nasce dalle idee e dalle intuizioni in relazione ad una determinato argomento, e costruisce una realtà virtuale dandole vita con le sue capacità mentali: immagina cioè l'evolversi di una realtà.

La simulazione può:

(1) essere utilizzata alla fine del processo mentale per controllare che le idee, elaborate dalla teoria, siano veritiere;

(2) aiutare lo scienziato ad elaborarla, in un continuo passaggio di informazioni dal cervello alla macchina.

Nel secondo capitolo spiegheremo il motivo che spinge gli sperimentatori a verificare tramite gli esperimenti le teorie economiche.

Illustreremo l'utilità dell'agente rappresentativo all'interno di modelli macroeconomici e la sua fragilità nel confronto con le scelte umane.

Metteremo in luce quelli che sono gli elementi determinanti nelle scelte degli agenti umani, quali il sesso, l'età, le condizioni sociali e culturali in cui il gruppo di appartenenza si trova a vivere.

In particolare è facilmente controvertibile l'assunzione classica che vede gli individui interessati solamente a se stessi: molti dei soggetti esaminati in studi specifici (Henrich ed altri, 2001) infatti non sono solo attenti al proprio payoff materiale, ma mostrano atteggiamenti di altruismo e sembrano disposti a mettersi in gioco cooperando tra di loro e anzi punendo comportamenti egoistici anche a costo di ridurre il proprio benessere materiale.

Il terzo capitolo sarà dedicato al tema centrale trattato dalla mia tesi, quello della presenza in economia di comportamenti altruistici e cooperativi. Il comportamento egoista generalmente ipotizzato nell'ambito dei tradizionali modelli economici è smentito da un'ampia serie di osservazioni empiriche (Henrich ed altri, 2001); è necessario ripercorrere però le diverse idee proposte dalla letteratura per spiegare questi comportamenti. Sarà quindi analizza l'ipotesi che l'altruismo derivi da processi di selezione biologica, o culturale e le relazioni con emozioni, incertezza, razionalità e processi di apprendimento.

L'idea economica di altruismo è particolarmente difficile da ricercare (nella letteratura economica e in quella di altre discipline) e da analizzare perché i commenti che contribuiscono alla sua definizione sono svariati e diversa natura. Inoltre l'argomento non può essere affrontato senza tener conto della sua relazione con temi quali coordinamento, apprendimento, istituzioni, teoria della razionalità e delle emozioni.

Passeremo poi a delineare gli strumenti utilizzati per la creazione del nostro programma: parleremo nel quarto capitolo della piattaforma Swarm, un insieme di programmi e librerie standard, da utilizzare per effettuare simulazioni ed analizzare sistemi complessi di comportamento, nell'ambito delle scienze naturali e sociali. Swarm permette di ricreare modelli di

comportamento di un insieme di "agenti" (ABM - Agent Based Modelling) indipendenti al fine di ottenere con regole semplici andamenti collettivi complessi.

Tratteggeremo le caratteristiche principali della programmazione ad oggetti ed illustreremo le sue caratteristiche di astrazione, incapsulamento, ereditarietà e polimorfismo.

Vedremo come l'inserimento di parti di codice, anche complesse, in librerie comuni, permette di sviluppare programmi modellati in maniera uniforme; questa caratteristica della programmazione ad oggetti favorisce la collaborazione tra studiosi rendendo maggiormente leggibile e comprensibile il codice di programmazione.

Nel quinto capitolo parleremo di SWIEE, un programma che permette di utilizzare Swarm per gli esperimenti. La simulazione di Swarm consiste in un mondo artificiale che ne rappresenta uno reale, nel quale interagiscono tra loro agenti artificiali.

SWIEE è stato progettato per integrare questo schema, attraverso l'uso delle classi RMI (chiamate a metodi remoti), permettendo ad agenti reali di interagire con il mondo artificiale e ai ricercatori di studiarne le conseguenze. Ogni agente umano prende parte all'esperimento effettuando le proprie scelte attraverso l'uso di un computer (client) in collegamento con il server che gestisce il programma. Il meccanismo RMI permette in particolare di chiamare i metodi di un oggetto che si trova su di un altro computer, in questo modo il client utilizza un applet presente sul server per svolgere l'esperimento.

L'ultima sezione è dedicata agli esperimenti: quello effettuato ad Alessandria con i solo agenti umani e quello a Torino, dove è stata inserita una fase di apprendimento con agenti artificiali per studiare eventuali influenze nel comportamento degli agenti umani.

L'esperimento di Alessandria si proponeva di analizzare il comportamento umano in situazioni predeterminate con un semplice gioco a somma 10. Il gioco prevedeva la formazione di squadre composte da tre giocatori umani che, investendo una cifra tra un set di numeri a

disposizione, dovevano puntare a raggiungere la somma di 10 per ottenere il punteggio maggiore. Sono stati effettuati quattro turni con regole diverse per confrontare eventuali differenze nel comportamento, nate dalla consapevolezza o meno del numero dei turni a disposizione e nella presenza di un set di numeri fisso o variabile.

A Torino abbiamo approfondito l'argomento dell'altruismo e della cooperazione utilizzando un gioco simile. Le finalità erano le stesse, ma le fasi di gioco erano due differenti: la prima, con giocatori umani in squadra con agenti artificiali guidati da algoritmi, serviva come fase di apprendimento per cercare di "educare" i partecipanti a precisi comportamenti; la seconda, dove le squadre erano composte da soli giocatori umani, serviva da banco di prova per verificare eventuali influenze nelle strategie utilizzate.

Nel capitolo sono riportate e commentate tutte le fasi di programmazione e realizzazione dell'esperimento, nonché tutti i problemi riscontrati e le soluzioni adottate. Sono inoltre commentati i punti salienti del programma, la sua struttura, con le variazioni realizzate e quelle progettate, ma non utilizzate a causa dei problemi tecnici riscontrati.

Saranno analizzati i dati di Alessandria e Torino, li confronteremo tra loro e con le nostre aspettative e cercheremo di trarne le opportune considerazioni.

Saranno infine proposti, nell'appendice, i listati del programma scritto in Java che abbiamo utilizzato.

PARTE PRIMA

Verificare l'economia con tecniche sperimentali

1.1 La sperimentazione in economia

L'economia è una disciplina che, come molte altre, basa i suoi principi sull'osservazione di eventi e sull'applicazione di teorie e modelli che descrivano tali fenomeni.

Tradizionalmente lo studio delle teorie economiche ha fondato le basi sull'analisi di dati statistici derivanti dall'osservazione di mercati naturali e quindi di situazioni reali.

Con la crescente complessità dei modelli teorici e la presenza di vincoli e condizioni sempre più precise e sottili, la verifica delle teorie mediante dati storici è divenuta sempre più difficile. È nata quindi la necessità di studiare i modelli teorici attraverso osservazioni sperimentali, così come già da molto tempo avviene per altre discipline.

L'economia sperimentale come la scienza, infatti, non si limita ad osservare la realtà, ma la riproduce e segue il suo evolversi in maniera sistematica, accurata e professionale.

La professionalità è il punto principale nel lavoro di una scienziato e di un'economista sperimentale e si manifesta soprattutto nel non trarre conclusioni generali da osservazioni singole che potrebbero non essere rappresentative e nel cercare continui riscontri per avvalorare le sue prime impressioni.

Entrambi cercano di controllare tutte le condizioni in cui si effettua l'esperimento e le relative osservazioni in modo da escludere che fattori esterni e imprevisti condizionino i risultati o colui che li analizza.

Essi descrivono i fatti e le osservazioni in modo quantitativo, attraverso l'utilizzo di strumenti più precisi possibili, perché questo metodo aumenta l'accuratezza e l'oggettività delle sue conclusioni.

Come tutte le scienze, l'economia sperimentale mira a conoscere la realtà ma anche, e soprattutto, a comprenderne il funzionamento, a controllarne l'evoluzione e a riprodurne gli effetti.

Le teorie nascono sempre dal bagaglio culturale dello sperimentatore, il quale non sarebbe neppure in grado di prescindere dalle teorie precedenti di cui è a conoscenza, dalle esperienze personali, dalla sua cultura, dalle

sue passioni e convinzioni. Il problema non riguarda solo le scienze dell'uomo; qualunque aspetto della realtà ha conseguenze e implicazioni per gli esseri umani, e così lo sperimentatore deve cercare di astrarsi dal proprio soggettivo modo di vedere la realtà.

Spesso può capitare che il desiderio di raggiungere risultati significati spinga l'osservatore comune a costruire realtà che portino a quei risultati.

Lo scienziato invece, volendo evitare di raggiungere conclusioni non oggettive, cerca di seguire regole precise che limitano i rischi e le possibilità di errore.

E' proprio questo tendere all'oggettività che rende credibili le scienze ed i risultati scientifici.

Per prima cosa la scienza dà molto peso al fatto di essere un'impresa collettiva, dove ogni collega ha il diritto\dovere di controllare, di accertare personalmente, di confermare o smentire una teoria.

La cosa importante, rispetto ad altre discipline come la filosofia dove si può parlare di teorie contrapposte come di diversi punti di vista, è che nella scienza un esperimento correttamente riproposto può avvalorare o inficiare una teoria precedente.

L'evoluzione della tecnologia infatti può essere di aiuto in questo caso allo scienziato e all'economista sperimentale per comprendere errori di concetto o strutturali nelle proprie teorie.

L'uso dei numeri e la descrizione non più, o non solo, qualitativa di un esperimento rende di fatto più facile e meno arbitrario ricavare predizioni precise da confrontare con i fatti osservati; diminuisce anche l'influenza delle visioni del mondo insite nel linguaggio comune usato per descrizioni principalmente qualitative.

Per ottenere la massima oggettività le scienze ricorrono, per dar vita e testare nuove teorie, al metodo degli esperimenti in laboratorio; questo perché rende più stringente la loro verifica empirica.

Le tecniche economiche sperimentali derivano da questo concetto e consistono nel creare un ambiente in miniatura in cui viene simulata, come in "laboratorio", una situazione economica rappresentata in tutti i suoi dettagli, utilizzando una metodologia ampiamente utilizzata in altre discipline quali la fisica, la chimica e la biologia. L'impiego di tale

metodologia in economia offre un modo alternativo di produrre i dati utilizzati per le verifiche empiriche. Gli esperimenti consentono di generare, in maniera controllata, dati di "buona qualità" in quanto creati "ad hoc" per l'indagine che si sta effettuando.

Controllare ogni elemento dell'esperimento permette di analizzare la relazione di causalità che si vuole verificare in un contesto in cui le variabili in gioco non possono alterare i risultati stessi dell'analisi.

L'uso dell'economia sperimentale costituisce quindi un valido supporto alla comprensione delle teorie economiche esistenti e per la scoperta di nuove teorie.

In biologia la creazione della vita *in vitro*, pur essendo un'impresa scientifica degna di nota, alla lunga non ci direbbe sullo spazio della vita possibile molto di più di quello che già sappiamo.

I computer invece offrono un ambiente alternativo in cui provare a sintetizzare la vita. La moderna tecnologia dei calcolatori infatti ha prodotto macchine con potenzialità formidabili per la creazione della vita *in silicio.*

1.2 Le radici della sperimentazione moderna

Colombo viene ancora oggi ricordato come colui che ha scoperto l'America, ma agli scolari si insegna che altri prima di lui, tra cui i Vichinghi, approdarono su quelle spiagge.

La cosa importante che rende agli occhi di tutti Colombo lo scopritore di un "nuovo" continente non è il fatto che sia stato il primo, ma che dal suo viaggio l'America non fu più "dimenticata" (Roth e Sotomayor, 1990).

Perciò il fatto che sia difficile risalire con precisione alle origini della sperimentazione economica o che venga riconosciuto, forse, Bernoulli[1] il

[1] Il paradosso di San Pietroburgo. Il paradosso economico risale al tempo di Pietro il Grande, quando a San Pietroburgo esisteva un casinò che permetteva di giocare qualunque gioco d'azzardo, in cambio di un prezzo d'entrata. Il casinò era, ad esempio, disposto a permettere a un giocatore di giocare a testa e croce con una moneta, e a raddoppiare la posta fino a quando fosse uscito testa per la prima volta. Quanto avrebbe dovuto essere disposto a pagare un giocatore per poter partecipare al gioco? Uno dei fondamenti dell'economia, già nel Settecento, era che una possibile misura dell'aspettativa di guadagno in una data situazione fosse il prodotto del guadagno ottenibile per la probabilità di

primo economista sperimentale, può risultare marginale rispetto ai filoni di pensiero su cui poggia l'economia sperimentale di oggi.

Per questo la nascita, o meglio lo sviluppo, di alcune delle più importanti teorie moderne può essere fatta risalire a tre filoni principali della storia della sperimentazione e all'evoluzione di numerosi lavori, ad essi collegati, che hanno messo in discussione gli assiomi della teoria neoclassica.

Il primo filone si propone di testare le scelte individuali.

Gli esperimenti in proposito sono numerosi; tra questi ricordiamo quello sulla teoria delle curve di indifferenza di Thurstone (1931) base della moderna teoria dell'utilità individuale.

L'esperimento di Thurstone, oltre ad essere il primo di cui siamo a conoscenza, voleva verificare la teoria neoclassica del consumatore, ricavando empiricamente la curva di indifferenza individuale.

Questi sottopose ad ogni soggetto diverse scelte ipotetiche (tra cappelli o giacche, giacche o scarpe, ecc.) e, raccolti ed analizzati tutti i dati, stimò una curva di indifferenza (partendo dall'iperbole) cercando di ridurre al minimo lo scarto dalle osservazioni empiriche.

Il lavoro venne pubblicato sul Journal of Social Psychology, nota rivista di psicologia e non certo di economia, ma il fatto che questi lavori fossero poco considerati dall'economia tradizionale non deve stupire.

Infatti veniva considerato irriverente il loro contrapporsi alle teorie classiche e neoclassiche e non era visto di buon occhio nemmeno

---

ottenerlo. Una misura dell'aspettativa di guadagno totale era allora la somma delle aspettative di guadagno per ogni possibile situazione. Poiché, nel caso del casinò, a ogni tiro il guadagno si raddoppia, ma la probabilità di arrivarci si dimezza, l'aspettativa di guadagno a ogni tiro è sempre la stessa: l'aspettativa di guadagno totale è dunque infinita. Il giocatore dovrebbe allora essere disposto a giocarsi tutto ciò che ha, pur di poter partecipare. Il che contrasta con l'ovvia osservazione che più paga per giocare, minore è la probabilità che riesca a guadagnare più di quanto ha pagato. Nel 1713 Jacob Bernoulli risolse il paradosso notando che il valore del denaro non è assoluto, e dipende invece da quanto se ne ha: una stessa somma vale tanto per chi ne ha molto meno, e poco per chi ne ha molto di più. Per calcolare l'aspettativa di guadagno si deve dunque moltiplicare la probabilità non per il guadagno effettivo, ma per quanto esso vale per il giocatore, che costituisce la sua cosiddetta utilità. Supponendo ad esempio che l'utilità decresca in maniera logaritmica, il guadagno totale cessa di essere infinito per diventare molto piccolo, e il paradosso scompare. La nozione di utilità è da allora entrata a far parte dell'economia, anche se spesso si censura la sua più ovvia conseguenza: che costa di più accontentare un ricco che molti poveri.

l'interdisciplinarità che fin dagli inizi ha caratterizzato l'economia sperimentale.

Dall'inizio del secolo l'economia era considerata, in maniera quasi unanime, una scienza aprioristico-deduttiva nel senso chiarito da Samuelson e Nordhaus:

"One possibile way of figuring out economic laws … is by controlled esperiments … Economist … cannot perform the controlled esperiments of chemists or biologists because they cannot easily control other important factors. Like astronomers or meteorologists, they must be content largely to observe." (Samuelson e Nordhaus, 1985).

Lo studio di Thurstone ispirò le critiche di Wallis e Friedman (1942) che evidenziarono uno degli aspetti più delicati dell'economia sperimentale, cioè la presunta inattendibilità dei risultati.

Secondo queste critiche le scelte dei soggetti sottoposti a test, che si trovavano ad agire in un ambiente artificiale, nel caso specifico risultavano semplicemente ipotetiche. Infatti non sarebbero stati spinti dalle motivazioni supposte dalla teoria o da quelle a cui sarebbero stati sottoposti nella realtà.

Ancora oggi si discute sul fatto che sia necessario o meno remunerare i soggetti che partecipano all'esperimento in modo proporzionale ai risultati ottenuti nel gioco, come suggerito da Friedman e da Sunder.

Queste critiche, a loro volta, influenzarono non poco le elaborazioni e gli esperimenti di Rousseas e Hart (1951), di Mosteller e Nogee (1951) ed il celebre lavoro di Allais (1953).

Il secondo filone riguarda invece le ipotesi su cui si basa l'odierna teoria dei giochi.

L'esperimento di Dresher e Flood[2] ha aperto la strada ad una serie infinita di lavori sulle strategie economiche ed è conosciuto con il nome di *Dilemma del prigioniero* (Flood, 1952; 1958).

Veniva sottoposta ai due giocatori una tabella dei payoff come quella riportata di seguito.

[2] Tenutosi nel 1950 presso la Rand Corporation.

	colonna 1	colonna 2
Riga 1	-1, 2	0.5, 1
Riga 2	0, 0.5	1, -1

I due soggetti non potevano comunicare tra loro e dovevano scegliere una riga o una colonna senza sapere la decisione dell'altro in ognuno dei cento turni che componevano l'esperimento.

La combinazione riga-colonna indicava le vincite in penny che sommate davano il compenso totale per ciascun giocatore.

Il gioco, così proposto, presentava in teoria un equilibrio[3] dato dalla combinazione tra riga 2 e colonna 1.

Infatti per il primo giocatore era sempre più vantaggiosa la riga 2 indipendentemente dalla scelta del suo avversario; stessa situazione per il secondo concorrente con la colonna 1.

A conclusione dell'esperimento Dresher e Flood rilevarono una discordanza tra il comportamento previsto dalla teoria e quello effettivamente tenuto nella realtà dai due concorrenti (Flood, 1952; 1958)[4].

Nel relazionare il lavoro venne inserito anche il commento critico di Nash che attribuiva la differenza ad una serie di variabili diverse tra realtà e teoria.

Per esempio i giocatori affrontavano cento turni e la strategia di ciascuno doveva per forza tenere in considerazione la reazione dell'altro.

Inoltre, secondo Nash, variando alcune condizioni, come coppie diverse in ogni turno, numero dei turni indeterminato, ecc., si sarebbero ottenuti risultati diversi.

Le critiche di Nash diedero il via ad innumerevoli ricerche negli anni successivi.

Tali ricerche svilupparono le sue teorie e in particolare puntarono sull'apprendimento nell'ambito dell'esperimento, che risultava condizionare in maniera vistosa l'esito dello stesso.

Oltre a Nash ricordiamo infine i lavori di Kalisch, Minor e Nering del 1954 e quelli di Schelling (1957).

Il terzo filone invece si incentra sull'*Industrial organization*, capofila di tutti gli esperimenti simulativi che riproducono un mercato artificiale.

[3] Definito e formalizzato da *Nash*, da cui prende il nome.
[4] Queste conclusioni si basavano sull'analisi dei payoffs di una coppia di giocatori.

Tra i primi ricercatori impegnati su questo argomento troviamo Chamberlin (1948), che realizzò un esperimento ricreando un mercato organizzato con compratori e venditori di un bene indivisibile, Siegel e Fouraker (1960), che studiarono il *bilateral monopoly* tra teoria economica classica, teoria dei giochi e psicologia.

Tutti e tre le direttrici sono state profondamente influenzate dalla pubblicazione *Theory of Games and Economic Behavior* del 1944 di von Neumann e Morgenstern.

Abbiamo quindi tracciato le linee generali che gettarono le basi per gli studi moderni sull'economia sperimentale, ma la vera rivoluzione in quel campo è avvenuta solo dopo gli anni '60.

Tra gli anni '60 e '70 nacquero numerosi laboratori permanenti, grazie anche al supporto della National Science Foundation, che garantivano non solo la nascita e l'evoluzione di nuovi esperimenti e teorie economiche, ma anche una certa continuità nel lavoro.

Questo filo conduttore sollecitò lo sviluppo e l'utilizzazione di nuovi strumenti e nuove tecniche per simulazioni sempre più realistiche.

Negli anni '80-'90 la crescita del numero di esperimenti, portati a termine con più o meno successo, divenne esponenziale e furono superate molte delle diffidenze riguardo alla sperimentazione ancora presenti nell'ambiente dell'economia tradizionale.

Le idee portate avanti dagli sperimentalisti cominciavano finalmente a riscuotere consenso fino al culmine dell'assegnazione del premio Nobel per l'economia ad uno di loro, Maurice Allais.

Grazie a questo riconoscimento ancora oggi non solo la sperimentazione viene utilizzata per "testare" nuove e vecchie teorie, ma studi diversi sullo stesso argomento consentono di metterlo a fuoco da più punti di vista, rendendo l'analisi più precisa e completa.

Oggi gli economisti che riconoscono il metodo sperimentale delle scienze naturali applicabile all'economia sono molti. Si consideri ad esempio la seguente affermazione di Hey (1991):

"This experimental way constitutes, of course, the prevailing methodology in many of the hard sciences; physics, chemistry and biology all consistently use the experimental approach. For such sciences,

experimental tests of theories are crucially important: only by subjecting a theory to test under the same controlled conditions as those under which the theory itself was generated can a theory be properly tested. The same logic should apply in economics, especially if economists continue to borrow other aspects of metodology of the hard sciences."

1.3 Il ruolo delle simulazioni

Per simulazione si intende una teoria scientifica espressa non con simboli, ma come programma di computer.

Infatti i concetti, i meccanismi, i processi, i fattori postulati da una teoria non vengono descritti da parole o simboli matematici, ma vengono incorporati in righe di codice.

"Il programma gira nel computer e riproduce i fenomeni che la teoria intende spiegare. E' per questo che le teorie espresse in forma tradizionale, cioè con le parole, i simboli della matematica, gli schemi grafici, si limitano a spiegare la realtà. Le teorie espresse come simulazioni invece la riproducono. Le simulazioni ci fanno capire la realtà ricreandola nel computer"(Parisi, 2001).

Nelle simulazione il computer svolge gran parte dei compiti dello scienziato e deriva previsioni empiriche da una teoria. Quando il programma con la simulazione infatti gira nel computer produce e raccoglie risultati, ovvero fenomeni, che non sono altro che le predizioni empiriche derivate dalla teoria.

Le simulazioni però non vengono utilizzate solo per "testare" le predizioni derivanti da una teoria espressa sotto forma di programma di computer, ma servono anche nella fase precedente.

Infatti tramite il programma e i risultati prodotti è possibile elaborare teorie o valutarne tutte le caratteristiche e le implicazioni derivanti anche quando sono ancora in fase di progettazione.

Possiamo quindi considerare il metodo simulativo come l'evoluzione di un metodo scientifico, quello degli esperimenti mentali, usato solo marginalmente e implicitamente per la ricerca di nuove teorie.

Lo scienziato normalmente elabora nella sua mente una teoria, che nasce dalle idee e dalle intuizioni in relazione ad una determinato argomento, e costruisce una realtà virtuale dandole vita con le sue capacità mentali: immagina cioè l'evolversi di una realtà.

La simulazione può essere utilizzata solo alla fine del processo mentale e serve per controllare che le idee, che hanno dato via alla teoria, siano veritiere; oppure può aiutare lo scienziato ad elaborarla, in un continuo passaggio di informazioni dal cervello alla macchina.

In questo modo vengono assegnati compiti diversi a mente e computer per ottimizzare il lavoro e per sfruttare a pieno il potenziale di ogni elemento. Infatti l'attività, normalmente svolta dal cervello, di immaginare il modificarsi della realtà al mutare di determinate variabili viene lasciato al computer, mentre lo scienziato elabora i risultati e li confronta con gli obiettivi della sua ricerca.

Le simulazioni vengono viste come un metodo per capire la realtà, ma riproducendone una parte possono essere considerate come realtà stessa.

Da tempo infatti si utilizzano simulazioni fisiche per riprodurre condizioni instabili o difficilmente osservabili, per studiarne le possibili evoluzioni.

Pensiamo alla galleria del vento dove vengono studiate le reazioni aerodinamiche di specifiche macchine a velocità e a pressione controllate, qui viene riprodotta una condizione particolare senza rischio e in modo da poterne analizzare i risultati. Come una vera prova testa la solidità e la veridicità di una applicazione pratica derivante da una teoria.

L'utilizzo del computer può essere affiancato da elementi reali esterni alla macchina per migliorare la resa della simulazione o per studiare comportamenti reali difficilmente riproducibili in un programma.

Se si vuole studiare, per esempio, un comportamento umano è indispensabile l'interazione tra un computer e un cervello non solo in fase di elaborazione della teoria, ma anche durante la simulazione. Con un'operazione di isolamento si cercano di eliminare tutte le interferenze e registrare solo quello che è oggetto dello studio.

Elaborata la teoria appropriata per rilevare un data reazione la sinergia uomo-macchina è ancora indispensabile: il computer simula la

situazione giusta per riprodurre il fenomeno e registra la reazione dell'essere umano.

1.3 I vantaggi delle simulazioni

Molti sono i vantaggi nell'uso delle simulazioni. Alcuni appaiono scontati e vengono subito individuati, ma altri hanno bisogno di un'analisi più approfondita per essere apprezzati.

Il primo vantaggio è legato al fatto che viene utilizzata una macchina per derivare predizioni empiriche dalle teorie e questo comporta una diminuzione degli errori nello svolgimento della simulazione.

E' chiaro che la capacità di calcolo e la precisione di una mente umana non possono essere paragonate alla velocità e alla sicurezza di una macchina.

Gli errori dello scienziato possono derivare dai suoi limiti cognitivi, ma anche dal suo desiderio di credere che certe predizioni empiriche derivino dalla teoria.

Con le simulazioni questo pericolo è scongiurato perché il processo è automatizzato nel computer. Se la simulazione produce certi risultati, coerenti con la teoria, vuol dire che la teoria espressa nel programma di simulazione effettivamente implica quei risultati.

Dal punto di vista del calcolo numerico il calcolatore riesce ad elaborare molto più velocemente e con maggiore sicurezza. C'è però sempre il rischio di incappare in errori di programmazione che portano a risultati fuorvianti o falsamente sorprendenti.

I modelli macroeconomici sono infatti facilmente soggetti ad errori concettuali o solamente di esecuzione che portano a risultati sbagliati. L'attenzione nel controllare i risultati non deve essere minore rispetto al metodo tradizionale, se non si vogliono raggiungere conclusioni errate (Axtell ed Epstein, 1994).

Un secondo vantaggio derivante dalle simulazioni è che la teoria alla base del programma deve forzatamente essere ben chiara e definita e deve essere articolata in modo specifico e dettagliato. Una teoria debole o mal progettata non può dar vita ad un programma senza presentare mancanze

strutturali o creare errori nei risultati, che risultano differenti da quelli previsti.

Un terzo vantaggio permette all'economista sperimentale di moltiplicare le predizioni che possono essere tratte da una teoria. Con la possibilità di controllare le variabili ambientali in cui si svolge la simulazione; sono infatti moltissime le variazioni sul tema centrale di una teoria, oltre al fatto che ripeterle può essere utile per testarne la solidità.

Le teorie scientifiche sono sottoposte a due tipi di verifica, quella interna e quella esterna. Per verifica esterna si intende il riscontro nella realtà delle predizioni estratte dalla teoria, se questa verifica empirica risulta positiva la teoria viene considerata veritiera, altrimenti viene smentita.

La verifica interna, invece, è meno conosciuta poiché avviene in forma privata, implicita e non soggettiva.

Le simulazioni, automatizzando il processo di derivazione delle predizioni empiriche, rendono oggettiva e sistematica la verifica interna delle teorie. Quando infatti una teoria è espressa come simulazione sul computer i risultati sono indubbiamente predizioni derivate dalla teoria, c'è inoltre la possibilità di creare una grande quantità di dati per rendere la verifica più ricca, differenziata e completa.

Con queste simulazioni si possono inoltre riscontrare un'altra serie di vantaggi derivanti dal fatto che queste sono laboratori virtuali ed non incorporano solo tutti i benefici dei laboratori sperimentali reali, ma non ne ereditano i limiti.

Una simulazione permette di riprodurre e studiare fenomeni che per ragioni fisiche non possono essere controllati in un laboratorio reale, per limiti di grandezza o di durata. Un fenomeno infatti crea problemi insormontabili o troppo onerosi sia se occupa uno spazio troppo grande, sia se la sua dimensione è talmente piccola da risultare difficilmente osservabile e manipolabile.

Grazie alle simulazioni possono essere riprodotti fenomeni accaduti nel passato ed ora non più presenti nella realtà. Si può trattare di un passato astronomico, geologico, biologico e storico ma che può essere riprodotto e studiato solo grazie al metodo simulativo.

Inoltre possono essere studiati tutti quei fenomeni che non possono essere scomposti o estrapolati dal contesto per non perdere le loro caratteristiche essenziali. Grazie alle capacità di memoria e di calcolo del computer è infatti possibile controllare e registrare una grande quantità di dati.

Infine si possono superare non solo problemi fisici, ma anche morali ed etici manipolando situazioni che nella realtà non è possibile realizzare per studiarne le possibili evoluzioni.

Ma il vantaggio più grande che può derivare dall'uso delle simulazione è rappresentato dalla possibilità di studiare sistemi complessi. Una scienza che adotti le simulazioni come strumento di ricerca tenderà ad evolvere in due direzioni: vedere la realtà composta fondamentalmente da sistemi complessi ed essere meno "disciplinare", cioè meno divisa in discipline diverse e sottodiscipline.

La scienza moderna ha finora puntato sullo studio di situazioni semplici, scartando la possibilità di riprodurre sistemi complessi, ma …

"*…la distinzione tra sistemi lineari e non lineari è fondamentale, e spiega chiaramente le ragioni della difficoltà di portare alla luce i principi sottostanti alla dinamica dalla vita* " (Langton, 1992).

In un sistema semplice una singola causa produce un singolo effetto, per cui conoscendo la causa possiamo prevederne l'effetto prodotto; possiamo cioè condizionare il verificarsi della causa, ovvero possiamo fare in modo che la causa si realizzi se l'effetto è desiderato o impedirlo se l'effetto è indesiderato.

L'effetto in un sistema semplice non sempre è prodotto da una sola causa, ma il ruolo che ciascuna causa ha nel produrre l'effetto è separabile delle altre e quantificabile.

I sistemi semplici finora studiati hanno caratteristiche tali da facilitare lo studio dei fenomeni:

- "gli stati successivi di un sistema semplice sono prevedibili se si conoscono quelli precedenti (…);
- se un sistema semplice si trasforma nel tempo, il modo in cui si trasforma è prevedibile;

- se un sistema semplice è perturbato da un evento esterno, l'effetto della perturbazione sul sistema è commisurato all'entità della perturbazione (…);
- se due sistemi semplici partono da condizioni iniziali diverse, il loro sviluppo nel tempo sarà diverso e questa diversità sarà tanto più grande quanto più sono diverse le condizioni iniziali (..);
- un sistema semplice può venire isolato dal contesto, cioè il fatto di operare in contesti diversi non cambia il funzionamento del sistema;
- un sistema semplice tende a non essere coinvolto in rapporti di causazione reciproca (…);
- (…) tende a non essere parte di una gerarchia di sistemi (…);
- un sistema semplice è composto di parti il cui ruolo nel determinare il comportamento complessivo del sistema è ben individuabile;
- un sistema semplice può essere riprodotto in copie identiche."

Ma la realtà che possiamo osservare è soprattutto composto da sistemi complessi. La caratteristica principale è opposta a quella di un sistema semplice: i sistemi complessi sono in genere composti di moltissimi elementi, uno diverso dall'altro; questi elementi si influenzano reciprocamente dando vita a proprietà globali non deducibili e non prevedibili.

Gli stadi futuri non sono in genere molto prevedibili pur conoscendo gli stadi precedenti; quindi anche le successive evoluzioni del sistema e le sue trasformazioni sono poco prevedibili.

Le perturbazioni esterne non solo sono per gran parte imprevedibili, ma condizionano il sistema in modo da modificarlo in maniera non commisurata all'entità della perturbazione: una piccola perturbazione può cioè modificare radicalmente un sistema complesso, mentre una grande può essere assorbita senza avere conseguenze sul sistema.

Al contrario dei sistemi semplici quelli complessi sono molto sensibili alle condizioni di partenza, in questo modo due sistemi uguali che partono

da condizioni leggermente diverse possono dare risultati radicalmente opposti.

Infine i sistemi complessi non possono essere riprodotti in copie identiche che diano gli stessi risultati.

Il lavoro svolto col computer nelle simulazioni permette di ricreare un sistema complesso, modificarne i singoli componenti ed osservarne le possibili evoluzioni. Nelle simulazioni il ricercatore, oltre a poter manipolare, può studiare a fondo il sistema aprendo la "scatola nera" ed osservandone il funzionamento.

Per studiare i sistemi semplici è possibile osservarli solamente e lasciare che la mente umana elabori la sequenza che porta alla comprensione del fenomeno. Per capire i sistemi complessi invece bisogna essere in grado di riprodurli.

Grazie alle simulazioni e all'uso delle potenzialità del computer è possibile ricreare modelli complessi e, dovendo per forza formalizzare i singoli componenti del sistema per creare il programma, è possibile capire a fondo il funzionamento del sistema stesso.

1.3.2 Le critiche alle simulazioni

Sicuramente appare chiaro che come ogni altra cosa le simulazioni hanno delle limitazioni e dei difetti. Alcuni di questi vanno realmente considerati quando si analizza nel suo complesso il metodo simulativo e bisogna cercare di circoscriverli evitandone i trabocchetti che potrebbero invalidare o confondere i risultati raggiunti.

Alcune delle critiche fatte alle simulazioni non hanno basi reali; sono cioè il frutto della mancata comprensione della natura del metodo in esame o, ancora peggio, le manifestazioni di un boicottaggio per contrastare lo svilupparsi di un metodo innovativo.

Secondo l'opinione di alcuni le simulazioni sono una rappresentazione troppo semplificata dalle realtà. In effetti è unanimemente riconosciuto che la realtà sia estremamente complessa e variegata e che il raggiungimento di risultati significativi e la descrizione di fenomeni empirici richiedano un

grande impiego di tempo e di lavoro, ma non è corretto sminuirne le potenzialità.

Una simulazione è prima di tutto una teoria e come tale cerca di estrapolare dalla realtà un fenomeno da studiare e spiegare; lo si fa operando certamente delle semplificazioni, ma soprattutto cercando di cogliere l'essenziale al di là della complicatezza e della varietà dei fenomeni osservati.

Stabilito questo si può discutere ed interrogarsi sul fatto che le semplificazioni fatte siano quelle giuste, ovvero se quello che è stato estrapolato dal contesto serve e basta a spiegare un dato fenomeno preso in esame, ma questo è valido per tutte le teorie tradizionali o simulative.

Bisogna d'altronde ammettere che dalle simulazioni, poiché cercano di riprodurre la realtà oltre a spiegarla, ci si aspetta che includano sempre maggiori aspetti e dettagli e questo non può che andare a loro vantaggio. Mentre col tempo una simulazione può migliorare, essere più realistica e diventare espressione sempre più complessa di un fenomeno reale, il metodo tradizionale rimane statico nel tempo permettendo di analizzare solo fenomeni singoli e semplici.

I veri difetti naturalmente esistono, come per ogni altra cosa, e devono essere attentamente considerati.

Alcuni dei problemi attuali delle simulazioni sono dovuti principalmente al fatto che ancora non si è imparato a sfruttarne a pieno e nella giusta maniera tutte le potenzialità. Ci si può quindi aspettare che in un futuro prossimo molti di questi problemi potranno essere superati o ridimensionati.

Molti di questi infatti derivano dal fatto che il metodo sperimentale non è ancora penetrato in misura sufficiente all'interno delle discipline scientifiche tradizionali, dove esiste una maggiore familiarità con la letteratura empirica e dove si potranno in futuro sfruttare nel migliore dei modi tutti i vantaggi della collaborazione e dell'integrazione dei due metodi.

Uno dei problemi riscontrabili nell' analisi di un esperimento è la scarsa verifica esterna effettuata, sia perché il riscontro nella realtà è ancora basso, sia perché l'attenzione si concentra principalmente sulla programmazione e sullo studio dei risultati. I ricercatori spesso si

accontentano di costruire la simulazione, di osservarne e analizzarne i risultati, di usare la simulazione come laboratorio sperimentale virtuale per determinare come variano i risultati variando le condizioni; non si preoccupano molto di stabilire qual è la corrispondenza tra i risultati della simulazione e la realtà empirica.

C'è infine il rischio concreto di commettere errori nella programmazione utilizzando parametri arbitrari inseriti per completare il sistema e che non corrispondono alla realtà.

1.4 Simulazioni: una rivoluzione per le scienze dell'uomo

Le simulazioni rappresentano un nuovo strumento per tutte le scienze permettendo il raggiungimento di risultati impensabili con i metodi tradizionali.

Mentre per le scienze della natura le simulazioni sono un'aggiunta ad un apparato teorico e metodologico già molto robusto, per le scienze dell'uomo rappresentano un traguardo rivoluzionario perché rimuovono alcune delle debolezze strutturali delle scienze dell'uomo e superano alcuni dei problemi che rendono lo studio degli esseri umani particolarmente difficile.

Il metodo della simulazione affronta il problema principale delle scienze dell'uomo che è quello della scarsa interazione tra teorie e fatti empirici. Infatti o le teorie mancano del tutto, o dalle teorie è impossibile derivare predizioni da confrontare con le evidenze empiriche.

Le simulazioni, come già detto, sono teorie che vengono applicate ad un programma per il computer e qualsiasi scienza che utilizzi simulazioni lavora necessariamente su teorie precise, esplicite e dettagliate.

Sicuramente da teorie ben definite, incorporate in simulazioni, non si possono ricavare che predizioni empiriche corrette escludendo che le teorie tradizioni possano non portare a predizioni veritiere.

Le scienze dell'uomo non possono usufruire del metodo sperimentale tradizionale (che si basa su esperimenti in laboratorio), non potendo estrapolare il fenomeno che si intende studiare dal suo contesto né si possono isolarne le cause presunte.

Tutte queste restrizioni vengono cancellate dall'adozione del metodo simulativo. In una simulazione è possibile riprodurre non solo il fenomeno, ma anche il suo contesto permettendo alle scienze dell'uomo di sfruttarne le potenzialità al massimo.

Grazie alle simulazioni, è possibile correggere un difetto congenito delle scienze che studiano i comportamenti umani, il frequente utilizzo di dati empirici qualitativi e non quantitativi per la definizione di una teoria. Utilizzare metodi quantitativi facilita il compito della scienza, rendendo più oggettiva e precisa la derivazione di predizioni empiriche e permettendo di confermare o smentire la teoria attraverso un chiaro e sicuro confronto delle predizioni con i dati empirici.

PARTE SECONDA

Il modello canonico delle scelte individuali

2.1 L'agente rappresentativo: un punto di partenza, ma non di arrivo

In tutti gli studi macroeconomici è essenziale e centrale la figura dell'agente rappresentativo. È il risultato di operazioni puramente matematiche e non necessariamente ha contenuti economici; nonostante ciò viene utilizzato per le sue caratteristiche semplificative che permettono la realizzazione di modelli complessi. Quanto più il modello rappresenta realisticamente la realtà, tanto più i limiti dell'agente rappresentativo vengono alla luce: le scelte individuali non necessariamente coincidono con il comportamento dell'agente rappresentativo; la reazione ad un'eventuale variazione nei parametri del modello originale può non essere quella dell'agente; in caso di scelta tra più possibilità l'agente può prendere una decisione diametralmente opposta a quella del singolo individuo (Kirman, 1992).

Recenti studi sul campo hanno portato alla luce consistenti differenze tra i comportamenti reali e le rappresentazioni che utilizzate per formalizzare gli agenti economici (Roth, 1991; Fehr e Gachter, 2000; Camerer, 2001).

Uno dei punti che appaiono falsi è l'assunzione classica che vede gli individui interessati solamente a se stessi: molti dei soggetti esaminati infatti non sono solo attenti al proprio payoff materiale, ma mostrano atteggiamenti di altruismo e sembrano disposti a mettersi in gioco cooperando tra di loro e anzi punendo comportamenti egoistici anche a costo di ridurre il proprio benessere materiale.

La prima giustificazione trovata per spiegare l'entità di queste deviazioni dal modello dell'agente rappresentativo è la presenza di realtà diverse che condizionano il comportamento dei soggetti all'esperimento.

Possono le differenze culturali e ambientali essere estrapolate dal comportamento umano per ricondurlo alle linee guida del modello canonico? La variazione delle condizioni economiche e sociali possono ulteriormente condizionare tale comportamento?

Il modello tradizionale non tiene conto delle condizioni personali del singolo soggetto. Dai test fatti però elementi quali il sesso, l'età, la

condizione sociale ed economica e l'appartenenza ad un gruppo specifico sembrano essere determinanti o perlomeno fortemente condizionanti nelle scelte dell'individuo.

Uno studio recente (Henrich ed altri, 2001)[5], condotto in dodici paesi diversi su quattro continenti, ha fotografato altrettante realtà differenti analizzando le scelte di micro società e riportando l'influenza di un'ampia varietà di condizioni economiche e culturali.

Il campione scelto non è casualmente disomogeneo: sono infatti rappresentate tre società che si basano sulla raccolta del foraggio, sei che praticano l'orticultura, quattro gruppi di allevatori nomadi, e tre di piccoli coltivatori sedentari, per differenziare al massimo le condizioni culturali e studiarne i reali effetti.

Nell'ultimatum game (UG) viene assegnato ad uno dei due giocatori, il "*proposer*", una somma di denaro equivalente ad uno o due giorni di paga nella società di appartenenza e si chiede di offrirne ad una seconda persona, il "*respondent*", una percentuale.

Il *respondent* può accettare l'offerta, nel qual caso entrambi ottengono le somma proposte, o rifiutarla, impedendo ad entrambi di ottenere un qualche profitto.

Se entrambi i giocatori decidessero secondo il modello canonico proposto dalla letteratura, il *proposer* saprebbe che qualunque offerta positiva fatta al *respondent* sarebbe comunque accettata e così offrirebbe la più piccola quantità possibile.

Questo secondo la teoria perché nella pratica, cioè negli esperimenti, le cose sono andate in maniera diversa.

Nella maggior parte dei casi i soggetti dell'esperimento hanno partecipato in forma anonima non conoscendo l'identità delle persone con cui sarebbero stati accoppiati. Questo per evitare che l'identità del patner condizionasse la scelta o alterasse i dati, infatti in società piccole come quelle selezionate non era possibile escludere che i partecipanti all'esperimento si conoscessero. Inoltre le scommesse del gioco sono sempre state basate su premi in denaro (o più semplicemente in tabacco o

[5] Il gruppo di ricerca, che ha scritto il paper, è stato creato dalla fondazione MacArthur *Research Group on the Nature and Origin of Norms and Preferences*, con l'appoggio dell'Università del Michigan, del Massachusetts e di Zurigo e del Santa Fe Institute.

altri beni) per motivare nella giusta maniera i giocatori e rendere veritiere le loro scelte[6].

I partecipanti all'esperimento sono stati selezionati in base alla loro comprensione del gioco e all'interesse dimostrato ai ricercatori: dopo aver eliminato chi mostrava di non aver capito perfettamente le spiegazioni infatti si sono potuti analizzare i dati raccolti escludendo eventuali errori di comprensione nelle scelte compiute.

I dati raccolti sono stati riassunti nella Tabella 1, che riporta la situazione riscontrata in tutti i gruppi analizzati.

Contrariamente a quanto previsto dal modello standard, in nessun gruppo è stato rilevato una percentuale media della somma offerta al *respondent* inferiore al 25%.

I risultati sono stati sorprendentemente diversi tra loro, ci sono state realtà, tra cui Torguud (Mongolia) e Mapuche (Cile), in cui il tasso ha raggiunto il 30-40% ed altre, Ache (Paraguay) e Lamelara (Indonesia), dove è stato offerto un tasso medio superiore al 50%.

Comparando questi dati con altri raccolti in società industriali con un tasso di scolarizzazione più elevato e condizioni economiche sociali simili (Roth 1991) dove la media si aggirava intorno al 44%, stupisce il range trovato in questo studio che riporta medie tra il 26% ed il 58%.

Paragonando anche i valori modali si evidenzia un'ulteriore differenza, infatti mentre nelle società industriali erano saldamente intorno al 50%, secondo questi dati variavano tra il 15% e 50%.

[6] Anche in questo caso l'uso della leva economica è stato considerato il miglior modo per motivare i partecipanti.

Group	Country	Mean offer	Modes (% of sample)		Rejection Rate	Rejection 20% of pot
Machiguenga	Peru	0.26	0.15/0.25	72%	1/21	1/10
Hadza (Small Camp)	Tanzania	0.27	0.20	38%	8/29	5/16
Tsimané	Bolivia	0.37	0.5/0.3/0.25		0/70	0/5
Quichua	Ecuador	0.27	0.25	47%	2/13	1/2
Hadza (all camps)	Tanzania	0.33	0.20/0.50	47%	13/55	9/21
Torguud	Mongolia	0.35	0.25	30%	1/20	0/1
Khazax	Mongolia	0.36	0.25			
Mapuche	Chile	0.34	0.50/0.33	46%	2/30	2/10
Au	PNG	0.43	0.03	33%	8/30	1/1
Gnau	PNG	0.38	0.04	32%	10/25	3/6
Hadza (Big Camp)	Tanzania	0.40	0.50	28%	5/26	4/5
Sangu (farmers)	Tanzania	0.41	0.50	35%	5/20	1/1
Unresettled	Zimbabwe	0.41	0.50	56%	3/31	2/5
Achuar	Ecuador	0.42	0.50	36%	0/16	0/1
Sangu (herders)	Tanzania	0.42	0.50	40%	1/20	1/1
Orma	Kenya	0.44	0.50	54%	2/56	0/0
Resettled	Zimbabwe	0.45	0.50	70%	12/86	4/7
Ache	Paraguay	0.51	0.50/0.40	75%	0/5	0/8
Lamelara	Indonesia	0.58	0.50	63%	0/2	0.37

Tabella 1: la terza colonna riporta le medie delle offerte, la quarta i valori modali insieme alla relativa percentuali, la quinta e la sesta segnalano i tassi di rifiuto da parte del respondent.

Possiamo riscontrare che anche i tassi di rifiuto segnalano importanti differenze nel campione, infatti nelle società industriali troviamo un tasso

del 20%, mentre nella tabella passiamo da uno 0% con un'offerta media inferiore al 30% in Bolivia, ad uno 0% con il 41% di offerta media in Equador; troviamo anche paesi in cui si rifiutano valori bassi sistematicamente e ad altri in cui si rifiutano sia valori bassi che valori alti.

Al fine di avvalorare i risultati del primo studio, sono stati effettuati altri "social game" in alcuni dei paesi testati.

I dati del gioco sui beni pubblici hanno confermato una divergenza con quanto previsto dal modello dell'agente rappresentativo. Il gioco chiedeva ai partecipanti di contribuire ad un ammontare comune che sarebbe stato ridistribuito alla comunità dai ricercatori.

Il modello classico prevedeva un comportamento da free rider, con una contribuzione pari a zero. Già con gli studenti universitari si sono rilevati risultati diversi con sì un valore modale di contribuzione pari a zero, ma con una media tra il 40% ed il 60%.

I risultati raccolti in sette dei paesi selezionati per il primo studio sono stati ancora una volta contrastanti tra loro: alcuni con medie basse intorno al 20%, altri addirittura senza casi estremi di free riding o contribuzione totale.

In tre di queste società è stato presentato anche un "dictator game", dove i gruppi partecipanti hanno dato anche in questo caso riposte diverse dal modello e dalla società industrializzata.

Il dictator game chiedeva di scegliere una parte della somma a loro disposizione da assegnare, a fondo perduto, ad un gruppo di persone che occupavano una posizione passiva nella società.

Il modello prevedeva media zero come il primo valore modale nell'ambiente universitario, mentre il secondo valore modale del 50% portava la media nuovamente intorno al 40%. Diversamente da entrambi si sono riscontrati nell'ultimo studio valori con risultati finali a volte apparentemente contrastanti.

2.2 Come giustificare le differenze

Le indubbie differenze riscontrate tra i vari paesi soggetti all'esperimento sono in prima istanza giustificabili con le specifiche

caratteristiche del gruppo di appartenenza, come le istituzioni e le norme cultuali di comportamento.

Dall'analisi dei dati e delle società i ricercatori hanno identificato due diversi fattori che spiegano in parte queste differenze.

- L'importanza della cooperazione, ovvero quanto peso ha la cooperazione nell'economia e nella produzione del gruppo analizzato.
- L'integrazione del mercato, quante persone fanno giornalmente affidamento sul mercato degli scambi nella loro vita all'interno della società.

Dall'analisi dei risultati si nota che in alcuni paesi la cooperazione è meno importante nello svolgimento della vita del gruppo sociale, come in Machiguenga ed in Tsimanè dove esistono soprattutto piccoli nuclei famigliari quasi completamente indipendenti e che hanno raramente contatti tra di loro. Al contrario altre società basano la loro economia sull'aiuto reciproco, come i cacciatori di balene di Lamelara che escono in mare con grosse barche e riescono a procurarsi il cibo solo grazie a manovre corali e accerchianti.

Questo condiziona la scelta di dividere tra tutti il surplus posseduto o pensare al benessere personale sopra tutto. Un alto grado di cooperazione può a conti fatti condizionare il comportamento nella distribuzione dei beni nell' ultimatum game.

L'abitudine a commerciare nella vita quotidiana, la seconda variabile individuata, può essere utile in giochi come quelli proposti e condizionare la scelta dell'ammontare da offrire nonché l'eventuale rifiuto.

La regressione ottenuta con entrambe le variabili ha un'alta possibilità di spiegare le divergenze, infatti i loro coefficienti positivi hanno un alto valore (circa 0.3) ed insieme spiegano quasi il 68% della varianza.

La differenza nell'approccio al quesito proposto può essere ricondotto all'ambiente e alla società di appartenenza con specifici riferimenti nella vita quotidiana.

Per esempio gli alti valori riscontrati nelle offerte (maggiore del 50%) in Au e Gnau riflette la cultura basata sul dono che condiziona pesantemente queste società. Tra queste comunità è in uso l'abitudine di ricambiare in un

periodo futuro i doni ricevuti, anche quelli non richiesti; per questo motivo il tasso di rifiuto, anche di alte percentuali, è così elevato.

Viene così giustificata la presenza simultanea di offerte consistenti e di rifiuti apparentemente immotivati.

Seguendo lo stesso ragionamento è possibile spiegare che i bassi valori delle offerte e l'alta percentuale dei rifiuti del Hadza sono lo specchio di una realtà in cui la condivisione è scarsamente considerata nella graduatoria dei valori della società.

Diversamente la comunità di Achè non ha rifiutato basse offerte, ma nonostante ciò la maggior parte dei partecipanti (98%) ha fatto offerte superiori al 40%. Questo coincide perfettamente con le caratteristiche etnografiche basate sulla condivisione del cibo e sulla cooperazione nella realizzazione di progetti comuni, nonostante l'assenza di paura per una eventuale punizione. Infatti i cacciatori al ritorno da una battuta lasciano le prede uccise ai margini del campo, spesso dicendo che la caccia ha avuto un esito infruttuoso; sarà poi qualcun altro a trovare la cacciagione e dividerla tra tutto il villaggio senza che il merito sia attribuito ad alcuno.

Infine a Machiguenga è stato registrato il più basso tasso di cooperazione tra i gruppi testati, questo rispecchia la vita dei componenti basata sulla scarsa importanza data alla condivisione del lavoro con elementi estranei al nucleo famigliare.

Sicuramente i risultati di tale ricerca stupiscono per quanto si discostano da quelle previste dal modello canonico, questo però non può bastare per smentire in toto il modello dell'agente rappresentativo (come sottolineato dagli stessi responsabili dello studio).

Bisogna interrogarsi almeno su due punti essenziali emersi dall'analisi dei risultati: innanzitutto il modello canonico è basato su un concetto di egoismo che spingerebbe ad un interessamento esclusivo per la situazione personale, ma i partecipanti si sono invece sistematicamente dimostrati poco attenti a massimizzare il proprio payoff.

Il secondo punto su cui riflettere è l'enorme differenziazione riscontrata nei vari gruppi testati che sembra derivare dalle caratteristiche intrinseche delle singole società.

Inoltre queste differenze possono essere previste dalla semplice analisi dell'ambiente socio-economico-culturale presente, dati che il modello tradizionale non prende assolutamente in considerazione.

Il modello canonico dell'agente rappresentativo sembra non reggere il confronto con le esperienze reali e soprattutto sembra determinante l'apporto della struttura economica della vita quotidiana che dovrebbe condizionare una eventuale revisione del modello che vuole prevedere le scelte dell'individuo.

Altri studi hanno sottolineato le debolezze del modello canonico e hanno dimostrato l'importanza dell'analisi empirica per identificare quelli che sono gli elementi da considerare e da includere nei modelli di previsione, elementi quali il sesso, l'età, le condizioni sociali e familiari.

Sicuramente quello che emerge dall'opinione generale è che l'agente rappresentativo, con tutti i suoi difetti e le limitazioni che impone ai modelli macroeconomici, rimane il modo più semplice per riprodurre il comportamento umano. Studiando però modelli più dettagliati si può puntare a formalizzare meglio un carattere specifico dell'individuo umano; sarà pur sempre una semplificazione della realtà, ma sempre più veritiera.

È per questo che studi sul comportamento umano in micro mondi possono risultare utili all'evoluzione del modello canonico ed alla definizione più dettagliata dell'agente rappresentativo.

PARTE TERZA

Altruismo e cooperazione

3.1 Altruismo ed egoismo in economia

Per poter studiare il comportamento altruistico e cooperativo nell'*uomo economico* bisogna partire dall'analisi della letteratura (economica, ma anche psicologica e sociobiologica) che si occupa di altruismo, inteso come negazione di un comportamento puramente rivolto alla massimizzazione del proprio benessere.

Il comportamento egoista generalmente ipotizzato nell'ambito dei tradizionali modelli economici è smentito da un'ampia serie di osservazioni empiriche (Henrich ed altri, 2001); bisogna ripercorrere però le diverse idee proposte dalla letteratura per spiegare questi comportamenti. Vengono quindi analizzate le ipotesi che l'altruismo derivi da processi di selezione biologica, o culturale e l'eventuale relazione con emozioni, incertezza, razionalità e processi di apprendimento.

L'idea economica di altruismo è particolarmente difficile da ricercare (nella letteratura economica e in quella di altre discipline) e da analizzare perché i commenti che contribuiscono alla sua definizione sono svariati e diversa natura. Inoltre l'argomento non può essere affrontato senza tener conto della sua relazione con temi quali coordinamento, apprendimento, istituzioni, teoria della razionalità e delle emozioni.

In economia che cosa si intende con la parola altruismo? Molti si sono dimostrati perplessi quando ho cercato di spiegare che ricercavo e studiavo comportamenti altruistici in economia, ritenendo i due termini opposti e non conciliabili in uno stesso concetto.

Il fatto è che con altruismo in economia si indicano tutti quegli atteggiamenti che hanno un comportamento non immediatamente e direttamente rivolto al puro interesse personale. In letteratura tale atteggiamento è indicato anche con altri termini quali *kindness, fairness, loyalty, cooperation*. Non dimentichiamo che altruismo è il termine che si contrappone naturalmente ad egoismo, quello che normalmente viene usato nel linguaggio comune ed è quindi quello che utilizzeremo principalmente in questa tesi.

Di conseguenza, facendo riferimento alla teoria economica standard, possiamo dire che un individuo egoista è colui il quale è motivato esclusivamente dal proprio benessere.

Nel costruire la funzione di utilità dell'individuo possiamo provare ad inserire anche il consumo di altri agenti e di conseguenza il loro benessere. Considerando come unico obbiettivo quello di massimizzare la propria funzione di utilità, si finisce per tener conto degli altri solo marginalmente e riproducendo ancora una volta un comportamento egoistico.

In un'impostazione di questo tipo, nessun comportamento potrà essere realmente altruista (Sacco e Zamagni, 1994, 231).

E' però possibile introdurre comportamenti non puramente egoistici (restando in un'ottica di scelta razionale) in una prospettiva diversa, facendo cioè riferimento ai vantaggi indiretti che è possibile ottenere; il riferimento ai vantaggi indiretti permette inoltre di giustificare l'altruismo ponendosi in una prospettiva di selezione naturale.

In questo modo abbiamo evidenziato il legame esistente tra altruismo e razionalità. Una buona parte della letteratura si è posta infatti l'obiettivo di giustificare, in termini di razionalità, l'altruismo osservabile empiricamente.

Alla razionalità sono state affiancate anche altre motivazioni quali apprendimento, istituzioni, comunicazioni, per cercare di giustificare comportamenti altruistici anche al di fuori del contesto della massimizzazione.

3.1.1 Definizioni di altruismo

Secondo il concetto standard di agente rappresentativo, il comportamento ritenuto naturale è quello egoista. Kahneman, e altri (1986, 102) osservano che nella letteratura economica tale assunzione assume forme differenti. In alcuni casi si tratta di un'ipotesi del tipo *as if* con natura prevalentemente metodologica. All'estremo opposto c'è invece la posizione che nega, anche da un punto di vista sostanziale, la possibilità di qualsiasi comportamento altruistico. I sostenitori di questa idea guarderebbero con imbarazzo ai comportamenti altruistici osservati nella realtà, attribuendo loro scarsa rilevanza.

Nella teoria dei giochi ed in quella economica standard si possono distinguere, nel comportamento dell'agente rappresentativo, tre diversi aspetti di "*privateness*".

Il benessere di un individuo è, come già accennato prima, legato innanzitutto al proprio consumo e non prevede simpatie o antipatie per gli altri (*self-centered welfare*). Ciascun individuo si pone l'obiettivo di massimizzare il proprio benessere (o il suo valore atteso) e non è interessato a quello degli altri (*self-welfare goal*). Le scelte compiute sono mirate esclusivamente al perseguimento dei propri obiettivi senza essere vincolate dal riconoscimento dei fini perseguiti dalle altre persone (*self-goal choice*), (Sen 1985, 347).

Riporto, qui di seguito, tre delle più diffuse definizioni di altruismo che si possono ritrovare in economia.

La teoria dei giochi prevede che un agente razionale coinvolto in una situazione del tipo *dilemma del prigioniero* sceglierà di tradire ("confessare"). Su questa linea Bergstrom e Stark (1993) definiscono altruista la scelta di cooperare ("non confessare"). Da questo concetto partono e si reggono gli esperimenti descritti in Andreoni e Miller (1993), Rabin (1993) e Andreoni (1995) nel quale con il termine *kindness* si indica il comportamento dei soggetti che contribuiscono positivamente al finanziamento di un bene pubblico.

In contraddizione con quanto detto prima, partendo dall'idea che l'individuo economico mira a massimizzare la propria funzione di utilità, Becker (1976) definisce come altruista un individuo la cui funzione di utilità include il consumo degli altri.

Non possiamo dimenticare il *moral hazard*, un problema di opportunismo che si può riscontrare in situazioni di asimmetria informativa (Arrow 1986). Una situazione tipica in cui si può riscontrare questo tipo di rischio è il rapporto di lavoro tra impresa e dipendenti. Ci sono però molti studi che evidenziano come il rapporto tra impresa e lavoratore possa essere basato o guidato verso lealtà e collaborazione.

3.2 Razionalità e altruismo finalizzate alla selezione naturale

Il ruolo dell'altruismo negli studi economici basati su modelli, viene sopratutto visto nell'ottica della sopravvivenza del gruppo e della selezione naturale e culturale derivante dal suo evolversi.

Il primo dei modelli che porterò ad esempio è quello proposto da Becker (1976). Questi ha immaginato un ambiente in cui vivono due individui, uno egoista e uno altruista ed entrambi decidono quanto mangiare e quanto destinare alla produzione. L'altruista include nella propria funzione di utilità anche il consumo del partner, può infatti decidere di donare una parte della sua ricchezza all'individuo egoista.

La funzione di utilità dell'altruista prevede che il trasferimento di risorse continui fino a quando l'utilità marginale del consumo dell'individuo altruista sia pari all'utilità marginale dell'egoista. Nel decidere poi quanta parte destinare alla produzione, l'altruista limiterà le proprie decisioni in modo da non causare riduzioni nel reddito del suo compagno anche a fronte di un guadagno personale.

La situazione crea un'interrelazione che condiziona le decisioni. Entrambi prendono in considerazione così la ricchezza complessiva, l'egoista infatti eviterà di danneggiare l'altro individuo se tale danno si rifletterà sui trasferimenti che riceve.

La situazione non è casuale e può essere riprodotta e complicata con l'introduzione di più agenti, confrontando le differenze in ambienti con un numero maggiore di egoisti o di altruisti.

Analoghe conclusioni sull'utilità dell'altruismo si trovano, ad esempio, negli studi di Rotemberg (1994)

Il modello proposto prevede che gli individui scelgano razionalmente un livello di altruismo. In questo modo le singole funzioni di utilità vengono rimpiazzate da una sola che comprende le "esigenze" dell'intera comunità. A questo punto massimizzare la nuova funzione di utilità, oltre ad essere espressione dell'altruismo e della cooperazione del gruppo, mantiene la propria caratteristica di assoluta razionalità e permette di raggiungere livelli di utilità altrimenti impossibili da ottenere.

Un'estensione (Becker, 1976) del modello base è stata studiata per analizzare il caso in cui l'utilità degli agenti dipenda dalla fitness genetica (cioè dalla probabilità di riprodursi) invece che dal consumo. Anche in questo caso l'altruismo ha riportato risultati decisamente migliori di un comportamento puramente egoistico e può quindi risultare vincente in un processo di selezione naturale.

Bergstrom e Stark (1993) nel loro modello hanno proposto un sistema per cui la strategia di un individuo può essere determinata sia da fattori genetici che dall'imitazione dei "vicini". Lo studio dimostra che anche per gli agenti egoisti può rivelarsi ottimale scegliere atteggiamenti altruistici per massimizzare il proprio payoff[7].

Secondo le indicazioni ricavate da questi esempi un comportamento altruistico produce vantaggi per gli altri e potenzialmente ha un effetto negativo per chi lo attua. Da questo si possono ricavare due definizioni diverse di altruismo, secondo Simon (1983 e 1993), una forte (che prevede il sacrificio della propria "capacità di adattamento" a vantaggio di altri) e una debole (che è invece semplicemente basato su un interesse personale temperato).

3.3 Altruismo ed evoluzione biologica

L'idea alla base della sociobiologia è che le preferenze degli individui non siano arbitrarie ma siano piuttosto il prodotto dei processi evolutivi e che un comportamento cooperativo possa esserne il risultato

Nel mondo animali si possono individuare numerosi casi di comportamento altruistico e cooperativo che condizionano in maniera significativa la selezione naturale.

Tra tanti comportamenti, Cosmisdes e Tooby (1992) descrivono quello di una particolare specie di pipistrelli vampiri, i quali "donano" spontaneamente una parte del sangue predato ai propri simili che non sono riusciti a procurarselo durante la notte di caccia.

Questo comportamento, apparentemente altruistico, ha in biologia spiegazioni diverse. La prima fa riferimento all'idea di *group selection*. Il

[7] Nel modello in esame il payoff rappresenta la sola probabilità di riprodursi.

tasso di estinzione dei gruppi sembra condizionare l'evoluzione della specie maggiormente rispetto a quello individuale: i gruppi caratterizzati da individui altruistici infatti avrebbero maggiori capacità di sopravvivenza e quindi si espanderebbero a svantaggio dei gruppi formati da egoisti. Questa conclusione non è però unanime, infatti l'unità riproduttiva è l'individuo e non il gruppo (diversamente dalla selezione culturale che può giustificare una selezione del gruppo (Hayek, 1967)).

La *selezione della razza* (*kin selection*) può essere un'altra spiegazione, che giustifica solo l'altruismo rivolto esclusivamente ai propri parenti. L'obiettivo dell'individuo è quello di permetterebbe ai geni presenti nella famiglia di sopravvivere e di propagarsi (Hamilton, 1964)(*inclusive fitness* (Dawkins, 1976))[8].

Nel caso dei pipistrelli, e non solo, l'altruismo non è rivolto esclusivamente a membri della propria famiglia, ma si aiutano anche tra non consanguinei.

Per questo motivo è stato proposta una terza spiegazione centrata sul concetto di *altruismo reciproco* o scambio (differito) di favori (si vedano Williams, 1966 e Trivers, 1971)[9]. Ogni svantaggio conseguito oggi con un gesto altruista è ricompensato dal vantaggio di beneficiare, domani, del comportamento favorevole degli altri[10].

Il modello si complica se lo scambio di benefici non è immediato, perché per l'altruista c'è il rischio di non vedere ricambiato un favore. Per risolvere questo inconveniente è necessario inserire un meccanismo in grado di eliminare chi non coopera, cioè chi non ricambia l'altruismo. Tali meccanismi sembrano effettivamente esistere ed infatti sono stati individuati comportamenti punitivi con chi non coopera, siano essi pipistrelli[11] o esseri umani (Cosmisdes e Tooby, 1992, 180; Ostrom, 2000, 143).

[8] Teoria fondata sull'ipotesi di *gene egoista* introdotta da Dawkins. I geni sarebbero egoisti e interessati a sopravvivere e riprodursi, se necessario, anche a svantaggio dell'organismo che li ospita.

[9] Lo scambio di favori, o di regali, è analizzato, in una prospettiva economica da Akerlof (1984) sul quale torneremo.

[10] Sui vantaggi del comportamento cooperativo da un punto di vista biologico si veda anche Focus 88, febbraio 2000, pag. 40.

[11] I pipistrelli, ricordati nell'esempio, tendono a cedere il sangue soprattutto ad individui dal quale hanno ricevuto un analogo favore.

3.4 Selezione culturale

Anche le norme comportamentali subiscono una selezione culturale che potrebbe essere la chiave, assieme a quella naturale, per giustificare la presenza dell'altruismo in una società.

Secondo le conclusioni raggiunte da Hayek (1967)[12] gli individui avrebbero la tendenza a seguire le regole imposte dalla società in cui vivono. Queste regole vengono selezionate dal gruppo per raggiungere la massima efficienza.

Quando una certa tendenza alla cooperazione è utile ad un gruppo sociale è possibile che ne diventi una norma.

"Le proprietà individuali importanti per l'esistenza e la preservazione del gruppo, e di conseguenza anche per l'esistenza e la preservazione degli stessi individui, hanno preso forma tramite la selezione di quelle provenienti dagli individui viventi in gruppi che a ogni stadio dell'evoluzione hanno mostrato la tendenza ad agire in base a regole tali che hanno reso il gruppo più efficiente" (Hayek, 1967, 153).

Un ruolo determinante nell'evoluzione e nell'apprendimento di tali regole lo svolge il mercato, affiancato da altra istituzioni quali la reputazione, che permette agli individui egoisti di collaborare per il benessere comune.

La reputazione è riproposta anche in Tullock (1985 e 1999) con un modello di mercato in un contesto tipo dilemma del prigioniero ripetuto, riprendendo un concetto di Adam Smith. Il nuovo modello prevedeva una terza possibilità, oltre a quelle classiche del dilemma del prigioniero, quella di non giocare con un certo patner. In questa maniera ognuno ha interesse ad avere una buona reputazione e quindi, anche nelle situazioni in cui avrebbe vantaggio a defezionare, non lo fa.

3.5 Il tit for tat

L'evoluzione biologico-culturale di una strategia non egoistica riscontrata nei modelli presentati viene avvalorata dalla teoria del *tit for tat*

[12] Riproponendo l'idea di "group selection" vista nella letteratura biologica.

(in italiano “colpo su colpo”) proposto in un saggio sulla cooperazione (Axelrod, 1984). Axelrod invitò una serie di studiosi a proporre algoritmi per un torneo basato su una serie di iterazioni del dilemma del prigioniero. Furono presentate e poste a confronto, utilizzando simulazioni al computer, moltissime strategie. La popolazione era quindi composta da una serie molto ampia di algoritmi diversi. Una selezione naturale basata sui pay-off ottenuti dalle varie strategie escludeva turno dopo turno quelle meno efficienti, permettendo a quelle più efficaci di svilupparsi e diffondersi. La strategia vincente risultò essere appunto quella del "tit for tat": coopera se il tuo avversario ha cooperato nel turno precedente, altrimenti defeziona.

Skyrms (1996) presenta un esercizio concettualmente simile a quello di Axelrod. Anche in questo caso si parte da un semplice gioco: una versione semplificata dell'ultimatum game in cui due individui devono accordarsi sulla divisione di una torta. A certe condizioni si vede che strategie altruiste possono risultare vincenti e diffondersi a livello sociale. Questi modelli, insieme a molti altri, possono essere utilizzati per rappresentare forme di apprendimento che portano ad una selezione culturale o genetica.

La strategia ha un punto debole, risulta vantaggiosa per chi lo attua solo se altri partecipanti al gioco si comportano nello stesso modo.

Per questo sono state create numerose varianti del modello base, una delle quali prevede la presenza di un meccanismo di *correlazione* che faccia accoppiamenti non casuali fra i giocatori. In questo modo individui che hanno strategie simili hanno maggiori probabilità di giocare assieme.

Questo meccanismo, simile concettualmente al tit for tat, permette di essere altruisti con chi si dimostra altruista e viceversa.

3.6 Il ruolo dell’incertezza

Abbiamo già accennato al possibile ruolo dell’incertezza nel determinare comportamenti altruistici.

In tutti i modelli il comportamento degli agenti è condizionato dall’incertezza sull’evolversi della situazione globale. Per questo motivo gli agenti sviluppano una "preferenza per i tipi di azioni le cui conseguenze

siano prevedibili e paura nei confronti di quei tipi di azioni le cui conseguenze siano imprevedibili" (Hayek, 1967, 164).

La presenza dell'incertezza nelle decisioni degli individui potrebbe essere la causa della nascita e dello sviluppo di norme e comportamenti sociali da seguire. La scelta verso l'equilibrio, sia esso singolo o multiplo, è quella che garantisce la minore incertezza sul risultato. Come osserva Ochs (1995, 195) *"in games with multiple equilibria the required rationality is ... a social phenomenon, rather than a characteristic of individuals*".

In questo modo parte dell'altruismo dimostrato potrebbe derivare dalla difficoltà di valutare le conseguenze della propria scelta e dall'incertezza riguardo a quella del patner.

Questa teoria è stata introdotta a seguito dei risultati dello studio di Shafir e Tversky (1992), in cui si paragonavano i tassi di cooperazione in un esperimento del tipo dilemma del prigioniero ad un turno unico con i tassi di cooperazione in un analogo esperimento in cui i soggetti conoscono in anticipo la scelta compiuta dal rivale: i tassi di cooperazione risultano sensibilmente più bassi. La minore incertezza porta ad una diminuzione sensibile della tendenza all'altruismo.

Andreoni (1995), presumendo che la confusione sia un componente rilevante nel comportamento altruistico, analizza in un contesto sperimentale il possibile ruolo della non completa comprensione della struttura dei pay-off. Dai dati raccolti si può pensare che almeno il 50% dalla cooperazione derivi da una non perfetta comprensione.

In una tesi contrapposta, Sen (1985, pag. 344), sostiene invece che il senso di identità può spiegare la scelta di cooperare nell'ambito del dilemma del prigioniero in quanto spinge l'individuo a non ragionare in termini di obiettivi personali e ad accettare di seguire regole di condotta utili per il gruppo invece di un comportamento individuale massimizzante. In questo caso la cooperazione non nasce dall'incertezza ma al contrario, dall'esistenza di una conoscenza comune.

3.7 Le conseguenze dell'apprendimento

Un altro elemento determinante per il comportamento in un gioco basato sulla cooperazione, è sicuramente l'apprendimento. Il suo evolversi nel tempo condiziona le scelte dell'individuo ed in alcune costruzioni concettuali fa da tramite tra incertezza ed evoluzione culturale delle norme.

L'apprendimento è basato sulla percezione di regole o schemi nell'azione comune della società in cui si opera e su meccanismi di imitazione, anche inconsapevoli, per la creazione di modelli mentali.

Nella letteratura troviamo due filoni separati sulle conseguenze che avrebbe l'apprendimento sulle scelte dell'agente altruista. Uno, partendo dall'idea che la selezione naturale ha allentato la sua pressione sull'evoluzione del cervello umano, ritiene che l'altruismo sarebbe in larga parte frutto di eredità culturali e anche dall'esperienza personale (Witt, 1985 e Simon, 1993). Ma è altrettanto vero che se le preferenze per l'altruismo fossero legate all'apprendimento, potrebbero cambiare ed evolvere in continuazione.

L'altro, sostenendo che l'altruismo dipende per gran parte dalla non completa comprensione del gioco, si basa sul concetto che l'apprendimento disilluda l'agente e riduca la sua dose di altruismo (Ledyard, 1995). In questo caso, in realtà si utilizza una diversa idea di apprendimento.

3.8 Uno sguardo alla psicologia

Le emozioni sono la principale causa di un comportamento altruistico; è pertanto necessario rivolgere l'attenzione anche alla letteratura psicologica.

In particolare ritroviamo teorie legate all'empatia e alla simpatia ritenuti mediatori di diversi comportamenti morali (Eisenberg e Miller, 1987).

L'empatia è una risposta emotiva provocata da uno stato d'animo o da una condizione vissuta da un altro individuo. Sentire le emozioni che provano le persone che ci stanno intorno; pericolo, paura, gioia o noia sono sensazioni che empaticamente si percepiscono, anche solo vedendo la

persona che le prova, e si condividono. I vari lavori recensiti da Eisenberg e Miller (1987), Hoffman (1981) e Krebs (1982), evidenziano come l'indagine in campo psicologico sia rivolta innanzitutto a determinare se empatia ed altruismo siano effettivamente collegati. La risposta è generalmente positiva.

In caso di pericolo, per esempio, il comportamento altruistico di portare aiuto a chi si trova in difficoltà sembra naturale (Hoffman, 1981). In un caso di questo tipo, l'empatia sembra essere un sentimento innato, presumibilmente universale (non legato cioè a date culture) e giustificabile sulla base di un'ipotesi di selezione naturale.

Anche nei bambini si riscontra una particolare sensibilità agli stati d'animo delle presone che li circondano. I sentimenti degli adulti stimolano le corrispondenti emozioni nel bambino il quale impara come siano piacevoli anche i comportamenti che rendono felici gli altri (Aronfreed, 1970).

Il comportamento sociale che viene adottato nella società sarebbe quindi legato alle emozioni tramite un processo di apprendimento di tipo auto rinforzante.

L'empatia è sì un fattore importante, ma non l'unico, per spiegare l'altruismo. Ci sono infatti situazioni che non provocano empatia.

Molte situazioni provocano particolari sensazioni, ma non è detto che spingano ad atti altruistici: per esempio in caso di pericolo l'empatia stimola semplicemente l'individuo ad aiutare chi è in pericolo ma non lo spinge necessariamente a prestargli soccorso (Hoffman). Infatti la tendenza ad aiutare è contrastata dall'egoismo (che resiste anche in presenza di altruismo). La scelta del comportamento finale nasce infatti da un'attenta valutazione dei costi-benefici, per scegliere quello che è più vantaggioso per sé e per la società.

Rabin (1993) inserisce in un contesto di teoria dei giochi, alcune nozioni psicologiche relative all'altruismo e al comportamento verso gli altri. Gli individui sono in genere generosi solo verso chi si è comportato allo stesso modo nei loro confronti ma tendono a comportarsi scorrettamente (anche a costo di subire delle perdite) con chi non è stato equo[13].

La simmetria nel comportamento da parte degli individui, riscontrata negli studi di psicologia, ci riporta alle teorie del *tit for tat*, dell'altruismo puro.

Anche il rapporto tra impresa e lavoratore può essere condizionato da un comportamento egoista o altruista (Akerlof e Yellen), infatti il rendimento e lo sforzo profuso da un lavoratore, dipendono dall'onestà con cui ritiene di essere trattato dall'impresa.

Le imprese possono quindi ritenere utile pagare un salario superiore a quello di "market clearing" e non ridurre il salario in caso di recessione e in presenza di disoccupazione (si tratta della teoria dei "salari di efficienza" proposta inizialmente da Shapiro e Stigliz, 1984) (Novarese, 2000).

3.9 L'importanza dell'altruismo in economia

Abbiamo individuato diversi punti interessanti frutto degli studi fatti in materia di altruismo e cooperazione, che possono essere interessanti anche in chiave economica.

Innanzitutto abbiamo riscontrato che alcuni comportamenti apparentemente altruistici possono essere il risultato di processi di selezione. Questa selezione può avvenire a livello biologico e culturale. Le due componenti della selezione sono difficilmente estrapolabili dal loro contesto e poiché è impossibile escludere che una delle due cause abbia partecipato alla nascita di un determinato comportamento dobbiamo considerarle compartecipi.

Esistono diverse forme di comportamento altruistico, motivate da fattori diversi. Alcuni sono dettati da un riflesso "quasi automatico", in altri no.

L'evoluzione non è l'unico elemento importante da considerare, infatti ci sono anche fattori individuali legati alla personalità, all'esperienza e alla psicologia dell'individuo.

[13] Il desiderio di punire chi si comporta disonestamente è evidenziato anche dagli esperimenti di Kahneman, Knetsch e Taler (1986, pp 105-108) e sembrerebbe emergere anche nelle prove di Andreoni (1995).

Negli esperimenti di laboratorio emerge una certa tendenza alla cooperazione, anche se non perfetta, che tende a diminuire con il passare dei turni.

La letteratura ha formulato diverse ipotesi per questa diminuzione: i soggetti potrebbero imparare nel corso del gioco la soluzione corretta, oppure l'altruismo iniziale potrebbe essere dettato da motivi strategici, o ancora potrebbe dipendere dalla percezione (vera o meno) che gli altri non stiano collaborando e da un conseguente desiderio di rivalsa.
La contrapposizione tra altruismo ed egoismo tocca vari ambiti dell'economia.

Molti lavori trattano l'argomento riguardo alla nascita della società in contrapposizione ad uno stato di natura. Collegati a questo troviamo diversi filoni di analisi della contribuzione e del finanziamento dei beni pubblici attraverso l'uso di strumenti quali il dilemma del prigioniero e l'ultimatum game.

A livello più pratico troviamo studi riguardanti le diverse forme di incentivazione al lavoro e strettamente legati alla teoria d'impresa. Infatti l'altruismo potrebbe rappresentare una soluzione ai problemi di agenzia. Le differenti motivazioni dei lavoratori sono uno degli argomenti utilizzabili per spiegare le diverse performance delle imprese e dei sistemi economici (Sen, 1985 e Morishima, 1982).

PARTE QUARTA

Swarm

4.1 Che cos'è Swarm

Swarm è un progetto dell'Istituto di Studi sulla Complessità di Santa Fe (New Mexico, USA), poi trasformatosi in una organizzazione non-profit per lo sviluppo della simulazione ad agenti.

È nato nel 1995, da una gruppo di ricercatori, con l'obiettivo di creare un insieme di programmi e librerie standard, da usare per simulare ed analizzare sistemi complessi di comportamento, nell'ambito delle scienze naturali e sociali.

Il programma può essere scaricato via Internet da tutti gli interessati dal sito www.swarm.org; nel corso degli anni si sono costituiti diversi gruppi che confrontando i rispettivi lavori attraverso la rete hanno partecipato allo sviluppo del programma base.

Swarm è costituito da un insieme di librerie scritte in Java e in Objective C, rilasciate in licenza open source (sotto la GNU General Public License).

Permette di ricreare modelli di comportamento di un insieme di "agenti" (ABM - Agent Based Modelling) indipendenti al fine di ottenere un andamento collettivo che segue rigorose regole; per questa sua caratteristica Swarm può essere utilizzato in moltissime applicazioni quali per esempio modelli di economia, applicazioni di intelligenza artificiale, legami chimici tra molecole, ecc. Gli elementi fondamentali che formano una applicazione sono gli "agenti" e il "mondo"; i primi vengono definiti dal programmatore e costituiscono la struttura del sistema che si vuole realizzare, mentre il mondo può essere omesso o rappresentato in due o tre dimensioni dove gli agenti si muovono. Il comportamento di ogni singolo agente viene modificato a seconda dello stato del mondo istante per istante.

L'idea era di avere uno strumento che permettesse ai ricercatori di sviluppare le loro applicazioni senza dovere spendere tempo su problemi tipici di programmazione informatica (per esempio, la generazione di grafici o di un'interfaccia grafica, o il salvataggio dei risultati delle simulazioni).

Swarm consente una efficiente gestione del tempo, attraverso una perfetta sincronizzazione delle azioni interna al programma, e la possibilità

di interrogare, in ogni momento, il sistema sullo stato delle sue variabili attraverso sonde facilmente programmabili.

Inoltre, offrendo una struttura standard per le simulazioni, permette ai ricercatori di riprodurre i propri e gli altrui lavori.

Nei modelli basati su agenti, i sistemi sono costruiti come insiemi di entità autonome, eventualmente dotate di capacità di decisione e di apprendimento. E' cioè possibile introdurre, nei meccanismi comportamentali degli agenti, tecniche matematiche (funzioni a rete neurale, algoritmi genetici, classifier system...) che simulino l'apprendimento (più o meno "adattivo", più o meno razionale).

Ciascuna entità viene modellata a sé, e il comportamento del sistema viene ricostruito dall'interazione complessa dei singoli elementi (bottom-up). Un vantaggio di questo approccio a building block è che il modello può essere adattato e modificato con estrema semplicità. Per esempio, risulta facile introdurre nuovi "agenti" (rappresentino essi individui, contatori di una rete elettrica o commutatori di traffico in una rete telefonica) con un comportamento differente dagli altri (individui più informati, o contatori "intelligenti", o anche semplicemente commutatori guasti).

Allo sperimentatore è sufficiente modellare il comportamento (semplice) di ciascun agente, e le regole (semplici) che governano le interazioni dei diversi agenti. Non deve occuparsi di descrivere la complessità dell'interazione di tutti gli agenti insieme (anche se deve saperla interpretare - ma a questo riguardo esistono buone regole metodologiche che facilitano il compito).

La metodologia si presta all'analisi di problemi complessi (caratterizzati da dinamiche non lineari non banali) e 'distribuiti', dove cioè il comportamento del sistema dipende dai comportamenti individuali dei suoi componenti (e non - per esempio - l'opposto, in cui il comportamento dei singoli costituenti dipende esclusivamente dalle decisioni "centrali").

4.2 Librerie di Swarm

Le librerie in Swarm sono l'esempio di come è stato progettato e costruito il programma, infatti ogni libreria contiene una serie di opzioni che possono essere utilizzate semplicemente importandola all'inizio del file:

"import swarm.Globals"

L'inserimento di parti di codice, anche complesse, in librerie comuni, permette di sviluppare programmi modellati in maniera uniforme; questa sua caratteristica favorisce la collaborazione tra studiosi rendendo maggiormente leggibile e comprensibile il codice di programmazione.

È consuetudine, a proposito di comprensibilità, commentare tra le righe di codice le procedure utilizzate e la struttura del programma per permettere una più veloce lettura in caso di correzioni e modifiche da parte dello stesso programmatore o, soprattutto, da parte di altri. Per questo i commenti sono scritti quasi esclusivamente in inglese, per poter mettere a disposizione della comunità di Swarm i propri lavori e dare così il proprio contributo.

In aggiunta alle librerie che Swarm utilizza per la compilazione bisogna considerarne altre che permettono la creazione dei modelli bidimensionali, di algoritmi genetici e di reti neurali. Queste sono solo alcune delle librerie che permettono a Swarm di risolvere svariati problemi di sistemi complessi. All'interno di ogni libreria sono implementati gli oggetti che grazie ai loro"metodi" permettono la caratterizzazione di ogni agente che si crea nel proprio modello. Ognuno di questi metodi svolge delle funzioni particolari che possono essere richiamate dal programmatore includendo la libreria nel suo codice sorgente e utilizzando il metodo riferendolo all'agente.

4.2.1 SwarmObject

Questa libreria comprende gli aspetti fondamentali per la creazione del singolo agente. In essa inoltre è possibile implementare anche il codice relativo alla creazione di eventuali interfacce utenti grafiche. Quando si scrive il codice l'agente deve appartenere a questa classe.

4.2.2 Space Library

La libreria è utilizzata per la proiezione di un agente in uno spazio a due dimensioni. Vi sono molti metodi che possono essere usati quali per esempio il Discete2d, che permette di definire le coordinate dell'agente nelle due dimensioni e di prelevare dal mondo il valore assunto in una determinata posizione; oppure il Diffuse2d che permette di implementare l'equazione di diffusione con evaporazione settando i parametri relativi.

4.2.3 Simulation Tool

Permette di avere un insieme di librerie che possono essere utilizzate per ottenere un interfaccia con il mondo esterno. Vi sono librerie programmate per il caricamento di dati da un file per essere utilizzati come configurazione iniziale, oppure per il salvataggio di dati relativi al singolo agente che si utilizza durante una simulazione; inoltre è possibile avere dei supporti di grafica che permettono di creare delle finestre con bottoni per l'user, visualizzando eventuali parametri della simulazione.

4.2.4 Analysis Tool

Questa è una libreria che permette la creazione di grafici relativi ad eventuali parametri della simulazione con la possibilità di effettuare operazioni statistiche su una serie di dati quali media, entropia, determinazione del massimo o del minimo.

4.2.5 Collection Library

In questa libreria è possibile definite strutture che possono essere applicate ai singoli agenti per esempio gli Array, le Liste, i Set e le Map che, implementati in modo generale, permettono il loro utilizzo per un qualsiasi modello rendendola così molto versatile.

4.2.6 Activity Library

Tale libreria è utilizzata per creare attività e azioni di gruppo. Essa è la responsabile di tutte la dinamica della simulazione. Le azioni sono formate da un insieme di messaggi tra gli agenti implementati, chiamate di funzioni, e azioni di gruppo. Tale libreria garantisce il cambio di stato di un singolo agente in un determinato instante di tempo; in definitiva, questa libreria funge da temporizzatrice delle azioni.

4.2.7 Probe Library

Fanno parte di questa libreria il Probes, ProbeMaps e ProbeDisplay che servono per le interazioni dinamiche tra gli oggetti in una simulazione. Tali librerie permettono di settare e leggere eventuali variabili o parametri che possono cambiare istante per istante. Inoltre è possibile visualizzare tali valori in caselle grafiche in real time; per esempio il valore della costante di diffusione, il numero in oggetti creati, il tempo trascorso ecc.

4.2.8 Random Library

In questa libreria sono implementati alcuni algoritmi per la generazione di numeri random con distribuzioni di tipo uniforme o gaussiana. Grazie ai suoi metodi è possibile settare il range di valori e il tipo di algoritmo: SWB (Subtract with Borrow lagged Fibonacci), LCG (Linear Congruential Generator), ACG (Additive Congruential Generator) ed altri.

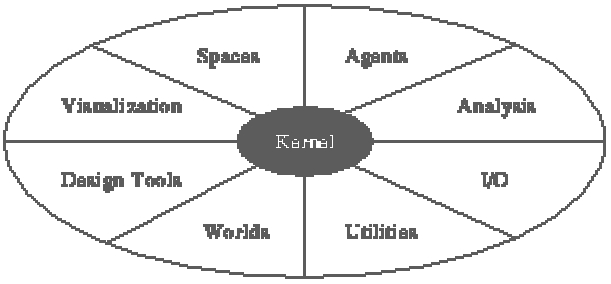
4.2.9 Defobj Library

Questa Libreria permette di avere un set di oggetti che si possono utilizzare con una programmazione orientata agli oggetti all'interno dello Swam. Esso definisce un'interfaccia specifica per l'implementazione di oggetti utilizzando un linguaggio Objective C con le librerie standard, inoltre permette l'allocazione di variabili in memoria, messaggi di errore e supporti per il debug.

4.3 Come creare una simulazione (struttura)

La costruzione di una qualsiasi applicazione Swarm richiede la conoscenza del linguaggio di programmazione Objective-C o Java.

Swarm è costituito da un set di librerie, orientate agli oggetti, raggruppate tutte in un unico kernel.



Creare una applicazione vuol dire creare una collezione di agenti che seguono determinati criteri o eventi che devono far parte di uno schedule ovvero di una struttura formata da una sequenza di azioni da eseguire. Le azioni svolte da un singolo, o da un gruppo di agenti, vengono selezionate in base all'informazione contenuta nel messaggio che il kernel spedisce, dallo stato della simulazione istante per istante, formando in questo modo una rete di connessioni che permette il cambiamento di stato di un agente.

La struttura generale è volutamente molto rigida, rifiutando i collegamenti diretti tra i diversi livelli del programma. Si richiede in questo modo un'attenzione ed uno sforzo maggiore nello scrivere il codice della struttura, soprattutto all'inizio, ma permette una più facile manipolazione e dà maggiori garanzie per la riuscita degli esperimenti.

4.3.1 Model Swarm

Il Model Swarm permette di contenere la vera implementazione in codice del modello che si vuole simulare. Esso è formato da un set di segnali di ingresso e di uscita che rappresentano i primi i parametri (numero di agenti, coordinate bidimensionali degli agenti nel mondo, parametri di diffusione, le azioni degli agenti, lo scheduler delle azioni, eventuali strutture correlate con il modello), mentre le uscite rappresentano i valori dei parametri a seconda dello stato. La parte fondamentale del movimento di un singolo agente è rappresentato all'interno dell'ActionGroup che permette di valutare i segnali di ingresso e di selezionare il movimento relativo.

4.3.2 Observer Swarm

Tale oggetto permette di osservare e misurare una simulazione, ovvero calcola i parametri di ingresso e di uscita definiti nel Model. Le attività dello schedule dell'Observer consistono nel ricevere come segnali di ingresso le uscite del Model e creare opportuni grafici. Per questa sua attività all'interno dell'Observer, è necessario richiamare delle librerie di grafica che permettono la creazione del mondo e dei singoli oggetti. Tali grafici vengono creati nel primo step dall'istanza EZGraph e dalle sue azioni, mentre nei successivi step, dove è richiesto l'aggiornamento istante per istante, viene utilizzato AverageSequence che disegna il grafico in base al valore di uscita.

4.3.3 Main o Start

Il Main o Start rappresenta l'oggetto di partenza dove si crea la sequenza di istanze da avviare. Crea il pannello per la gestione della simulazione e lo rende attivo.

4.3.4 RuleMaster

Il RuleMaster è l'oggetto che contiene tutte le regole di comportamento che dovranno seguire gli agenti ed è stato inserito dal Gruppo Torinese Utilizzatori di Swarm (creato dal prof. Terna).

Viene organizzato come un magazzino di regole. Ogni situazione che si vuole gestire deve essere formalizzata e destinata a portare a termine un compito preciso. Ogni agente, definito in altri file, interroga il RuleMaster per saper quali scelte compiere e solo dopo aver ottenuto una risposta agisce.

4.4 Programmazione ad oggetti

La programmazione ad oggetti su cui si basa Swarm è utile non solo perché permette di costruire simulazioni complesse, ma perché i file strutturati in questo modo risultano chiari e leggibili a tutti. Ogni elemento di un programma viene messo in una classe separata che definisce variabili e comportamenti (metodi); il programma principale usa esempi (instances) di queste classi (anche se è necessario un solo esempio della classe) e li fa interagire insieme.

La programmazione ad oggetti ha quattro proprietà essenziali:

- *Astrazione*: il programma (code) che descrive un oggetto è scritto in una classe, poi nel programma principale gli oggetti sono creati come esempi (instances) della classe.
- *Incapsulamento*: gli oggetti nascondono i propri metodi e dati, separando l'interfaccia dell'implementazione. In pratica si può vedere cosa fa un oggetto, ma le regole che lo guidano sono definite da un'altra parte. L'oggetto può essere così modificato senza dover stravolgere i programmi che lo utilizzano.
- *Ereditarietà*: ogni classe di oggetti può essere derivata da altre classe già definite. In questi casi, la classe "figlia" eredita tutte le variabili e i metodi della classe `madre', li può modificare e ne può aggiungere altri.

- *Polimorfismo*: si possono avere metodi con lo stesso nome, ma che hanno parametri diversi (si chiama anche overloading).

Si deve notare che Swarm non permette il polimorfismo. I metodi creati in Swarm devono dunque avere nomi unici.

4.4.1 Alcuni termini di Java

Il linguaggio usato nella programmazione di Swarm è l'Objective-C ed alcuni termini chiave meritano una breve spiegazione.

Innanzitutto riportiamo i termini riguardanti le *class*, definizione di un oggetto e "fabbrica" dello stesso, quali *superclass*, classe dalla quale l'oggetto eredita comportamento (metodi) e stati (variabili) in maniera ricorsiva e *subclass*, classe che eredita comportamenti e variabili da una superclass.

Le *instance object* invece definiscono un oggetto (esempio di una classe), che è stato creato e a cui è stato dato uno spazio in memoria. Un oggetto contiene 2 tipi di informazione: variabili (instance variable) e metodi (method). Le *instance variable* sono variabili pubbliche per tutti i metodi di un oggetto e danno informazioni sul suo stato, mentre i *method* sono funzioni che definiscono il comportamento dell'oggetto, e che possono essere richiamati tramite un messaggio mandato all'oggetto.

4.5 Gestire più agenti e conservare i dati

Per poter gestire e registrare l'interazione tra più agenti, siano essi virtuali o umani, vengono utilizzati oggetti specifici e già, in parte, pronti nelle librerie di Swarm: liste (Lists), mappe (Maps), vettori (Arrays).

4.5.1 List

Il protocollo List di Swarm implementa una lista con dimensioni flessibili (cioè, la lista cresce quando sono aggiunti elementi, e decresce quando vengono cancellati elementi). La lista permette di elencare oggetti,

e di eseguire, in modo semplice, azioni ripetitive sugli oggetti contenuti nella lista.

nameOfList= new ListImpl(getZone())

A questa lista si possono applicare diversi metodi:

- addFirst, addLast aggiunge un oggetto all'inizio o alla fine della lista.
- removeFirst, removeLast cancella il primo o l'ultimo elemento della lista.
- getCount riporta il numero di elementi nella lista.
- remove(aMember) per cancellare aMember dalla lista.
- removeAll cancella tutti gli oggetti della lista, ma non la lista stessa.
- deleteAll cancella tutti gli oggetti della lista, e cancella anche la lista dalla memoria.
- atOffset(i) per ritornare l'elemento che si trova in posizione i nella lista.

IMPORTANTE: nelle collezioni di Swarm, la numerazione degli elementi inizia da 0. Dunque, il primo elemento della lista si trova alla posizione 0, il secondo elemento si trova alla posizione 1, ecc.

4.5.2 Array

Il vettore si usa quando si ha bisogno che gli oggetti siano inseriti in un ordine specifico. La grande differenza con la lista è che il vettore viene dichiarato con la sua dimensione. Non si possono aggiungere oggetti fuori di questa dimensione (quindi non si possono usare i metodi addFirst o addLast). Dato che la dimensione del vettore non si può cambiare, non si possono cancellare elementi dal vettore. Per il resto i metodi applicabili sono gli stessi della lista.

Il vantaggio del vettore sulla lista è che gli oggetti possono essere trovati molto velocemente, perché il loro indice è un intero.

4.5.3 Map

La mappa può essere vista come una tabella con due file di oggetti. Nella prima si trovano i nomi degli oggetti, nella seconda, si trovano gli oggetti che vogliamo conservare.

Quindi, per mettere un oggetto in una mappa, dobbiamo anche specificare il suo nome. Possiamo poi chiedere alla mappa di ritornare, o cancellare l'oggetto che corrisponde ad una chiave particolare.

4.6 Schedule

Uno schedule è come un ordine del giorno, dove si scrive cosa fare ad ogni ora. Uno schedule è dunque un altro oggetto di Swarm nel quale possiamo inserire messaggi individuali, o gruppi di messaggi (ActionGroups). Fondamentale è la gestione della durata dell'intervallo fra le azioni; generalmente, il repeatInterval vale 1, il che vuol dire che le azioni sono ripetute ogni periodo.

Una volta creato lo schedule (generalmente nel metodo buildActions), tutti gli oggetti della simulazione sono pronti, devono essere solo attivati.

4.7 La classe Activity

Tramite i metodi activateIn, le librerie di Swarm integrano le azioni degli oggetti della simulazione. Il metodo main() segnala allo Swarm più alto nella gerarchia (nel nostro programma, il modelSwarm) di attivarsi in null (Java), dopo aver creato le sue azioni e i suoi schedule. Questo fa sì che il modelSwarm segnali al prossimo livello di Swarm di attivarsi, ecc; in questo modo, vengono sincronizzate le attività ai vari livelli della simulazione.

I metodi activateIn richiamano oggetti della classe Activity.

Sul piano pratico, risponde a metodi come run, stop, next, terminate, ecc, ma generalmente, la classe non viene usata direttamente; è il

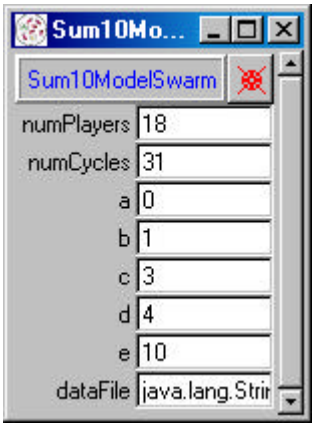
ControlPanel che permette di gestire la simulazione dal mouse, e che nasconde gli oggetti della classe Activity.

4.8 L'interfaccia grafica di Swarm

L'ObserverSwarm è la classe che ci permette di osservare e controllare l'andamento della simulazione. Lo strumento principale per la gestione, come accennato sopra, è il ControlPanel, che contiene 5 bottoni

- *start*: dà inizio alla simulazione (si ferma solo se è prevista una fine)
- *stop*: mette in pausa la simulazione
- *next*: la simulazione compie uno step
- *save*: salva la disposizione delle finestre
- *quit*: termina la simulazione (anche se non è ancora finita).

Le funzioni dell'ObserverSwarm ci permettono di creare delle sonde (Probes) per vedere i valori dei parametri degli oggetti della simulazione. Abbiamo la possibilità di cambiarli prima di lanciare la simulazione (basta cambiare il valore nella finestra e premere il tasto 'return'). Questo dimostra i vantaggi della programmazione con gli oggetti: è altrettanto semplice far girare il programma con 3, 10 o 100 agenti.



4.9 Messaggi d'errore

Tutti i file devono essere compilati (in questo e nei lavori precedenti qui riportati è stato usato Jdk), in questo modo possono essere segnalati immediatamente un'ampia serie di errori.

Due sono i tipi principali:

- *syntax*: errori di sintassi
- *semantic*: errori nel uso di funzioni di Java o javaSwarm, a volte collegati ad errori di sintassi.

I messaggi di errore in Java sono generalmente chiari, con proposte di soluzioni; Java dà anche l'indicazione del file e della riga in cui ha trovato l'errore.

I messaggi di errore sono spesso collegati tra loro, nel senso che risolverne uno può servire a risolverne altri.

Molti errori però non vengono segnalati in fase di compilazione, ma appaiono solo in fase di esecuzione, alcuni degli errori di runtime possono però essere evitati mediante l'inserimento di blocchi try/catch.

PARTE QUINTA

SWIEE

5.1 Il progetto SWIEE

SWIEE[14] è un progetto nato dalle idee del dottor Riccardo Boero supportato dal GTUS[15] (Gruppo Torinese Utilizzatori Swarm), un gruppo dell'Università di Torino che sviluppa i suoi programmi con Swarm.

Il progetto sfrutta alcuni vantaggi di Swarm per la costruzione degli ABM (Agent Based Model), quali la flessibilità dei modelli che rappresentano situazioni reali, la possibilità di riutilizzare un codice scritto in maniera semplice e quella di trasmettere simulazioni e risultati ad altri che possono replicarli e verificarli.

Gli studi sperimentali in economia necessitano di programmi, di software, per costruire gli esperimenti e per strutturarli nella maniera più chiara e comprensibile possibile. La struttura necessita di un mondo, di agenti virtuali che interagiscono e di un magazzino dove stoccare ordinatamente tutti i dati raccolti, indispensabili per l'analisi dei risultati.

L'idea alla base del progetto, ideato e realizzato dal dott. Riccardo Boero, era di creare una piattaforma per gli esperimenti economici, da poter utilizzare, per esempio, per test sui beni pubblici o sui comportamenti sociali.

La tipica simulazione di Swarm consiste in un mondo artificiale che ne rappresenta uno reale, nel quale interagiscono tra loro agenti virtuali simili a quelli reali.

SWIEE è stato progettato per integrare questo schema permettendo ad agenti reali di interagire con il mondo artificiale programmato e ai ricercatori di studiarne le scelte.

In questo modo un ricercatore non deve programmare partendo da zero, ma può usufruire di un lavoro ben strutturato e deve solo completare la parte riservata allo specifico esperimento. Naturalmente è stato studiato per un utilizzo prettamente economico, ma è altrettanto indicato per tutte le scienze sociali.

Dal punto di vista tecnico, uno dei punti di forza di SWIEE è certamente il linguaggio di programmazione utilizzato per costruire le

[14] http://SWIEE.econ.unito.it
[15] http://eco83.econ.unito.it/swarm

simulazioni: Java. All'inizio del progetto veniva utilizzato, per sviluppare i programmi di Swarm, l'Objective-C, poi è stato preferito il Java che è più conosciuto e non meno flessibile. Attualmente è possibile programmare con entrambi i linguaggi a seconda delle esigenze e delle conoscenze in materia.

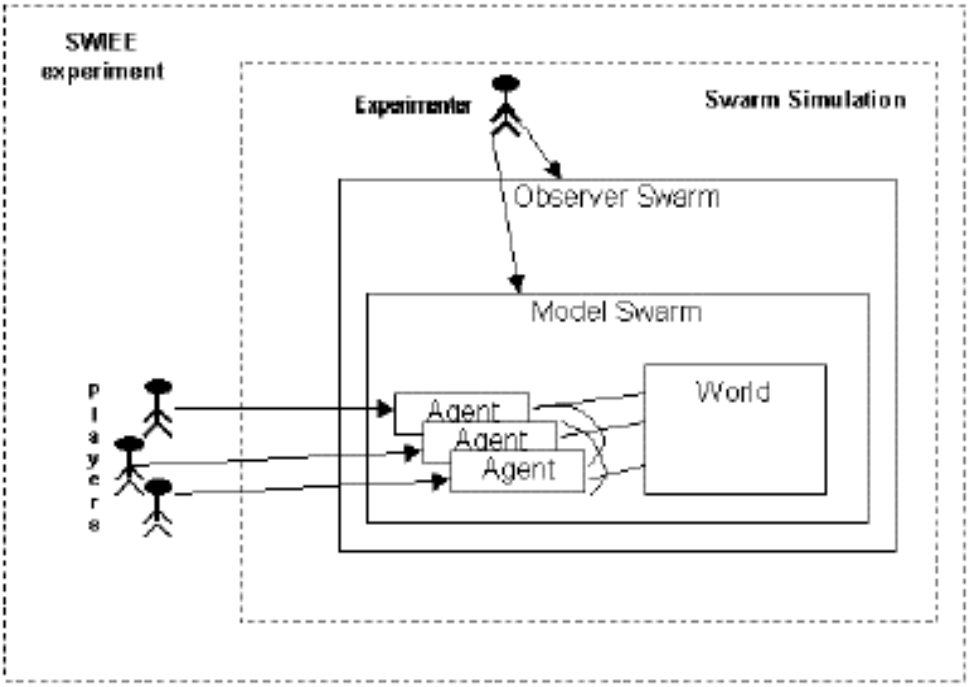
SWIEE è, come Swarm, un progetto open source ed utilizza i protocolli TCP/IP per la comunicazione tra client e server.

Infatti l'aggiunta di agenti umani al sistema prevede l'inserimento di altri computer, al di fuor del server, che permettono di interagire con il mondo virtuale. I giocatori umani compiono le loro scelte guidando agenti virtuali privi di capacità decisionali. In questo modo ogni agente interagisce con il sistema allo stesso modo; l'unica differenza sta nel set delle regole di comportamento a disposizione: deciso a priori quello degli agenti virtuali, dalle scelte del giocatore umano quello degli altri. La connessione tra i client e il server avviene tramite la rete che può essere quella locale o Internet.

Il progetto di SWIEE si è sviluppato seguendo due linee guida: costruire semplici esperimenti mantenendo il più possibile la struttura di Swarm e sviluppare uno strumento completo e versatile.

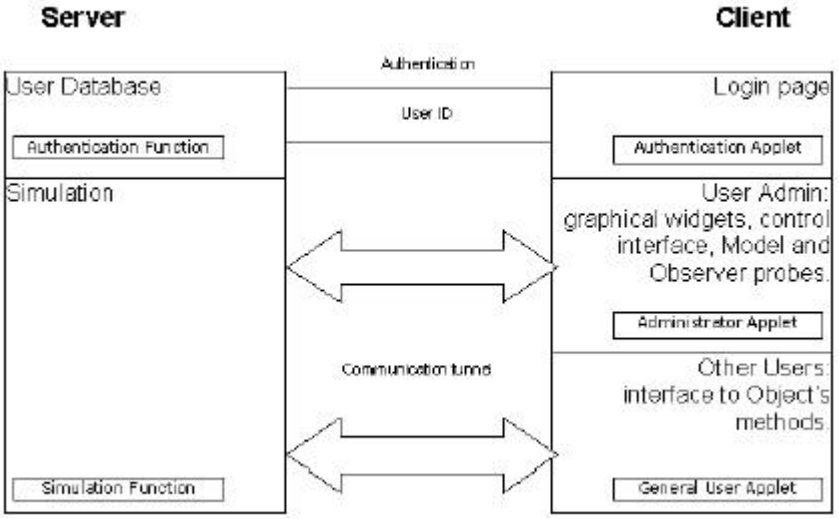
Per seguire entrambe le linee guida apparentemente divergenti, sono stati eseguiti due step diversi nella progettazione del software:

- Il primo prevedeva di creare il collegamento tra agenti virtuali "vuoti" ed giocatori umani che li guidano, mantenendo la struttura standard di Swarm (chiamato Semplice). Si voleva creare la struttura portante, adatta a supportare i primi esperimenti con soli agenti umani, più snella possibile in modo da essere facilmente utilizzata da altri.



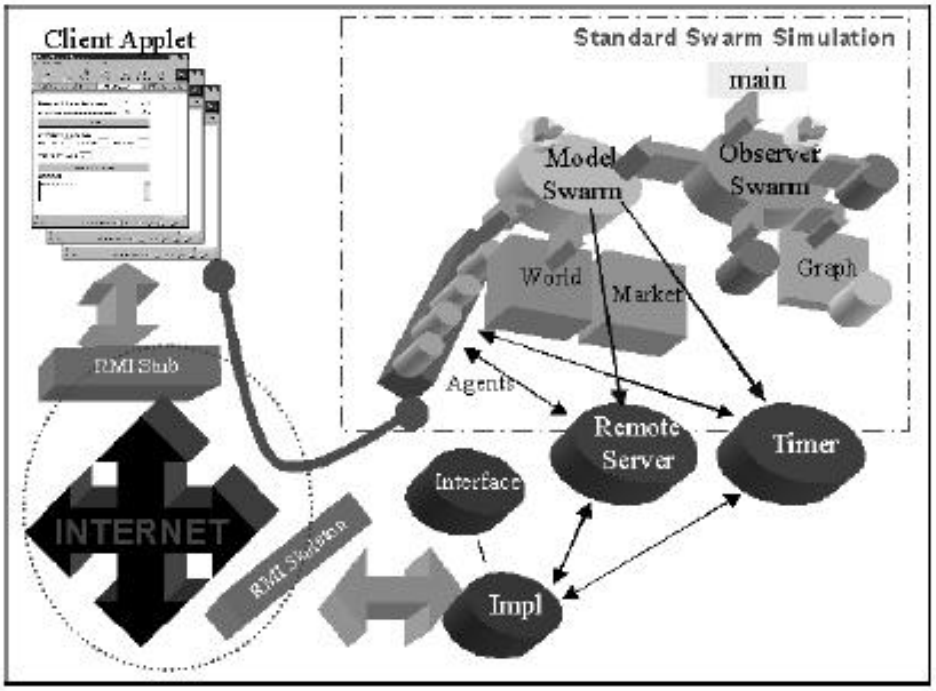
Struttura Semplice

- Il secondo comprendeva un ulteriore passo verso la totale automatizzazione del programma, con sistemi più "complessi" per l'autenticazione del giocatore e l'inserimento di agenti virtuali autonomi e della struttura adatta ad accoglierli (Struttura Complessa). L'idea finale era quella di permettere una evoluzione del gioco via Internet, mantenendo le stesse garanzie di sicurezza nell'identificazione del giocatore che non si deve più muovere da casa. Si vuole aprire così la porta ad esperimenti multipli, magari con sessioni giornaliere, per testare l'evoluzione del comportamento o le strategie del singolo giocatore.



Struttura Complessa

Rispetto a Swarm la struttura della piattaforma ha subito delle variazioni:



- è stato inserito un oggetto (Timer) per gestire il tempo all'esterno della struttura primaria; infatti mentre Swarm ha una struttura interna per sincronizzare le azioni degli agenti virtuali, SWIEE ha introdotto una serie di controlli per sopperire a inevitabili ritardi nelle scelte dei giocatori umani.

- per collegare gli agenti virtuali con gli umani (attraverso gli applet) si è scelto di utilizzare le classi RMI (chiamate a metodi remoti) di Java.

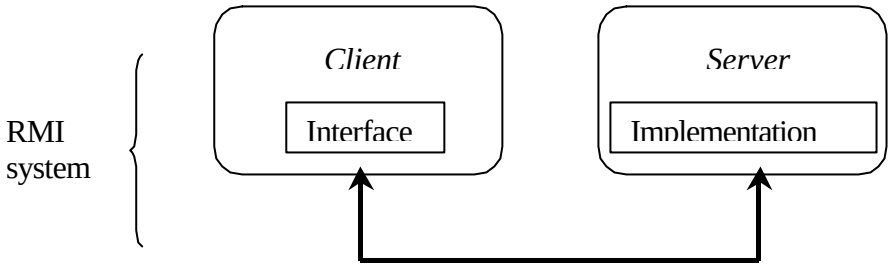
5.1.1 Chiamate a metodi remoti

L'utilizzo di Swarm scritto in Java e di Internet (o in alternativa di una rete locale LAN) permettono l'impiego di un collegamento remoto tra due applicazioni: il client con il quale interagisce il giocatore umano da una parte, e il server della simulazione Swarm dall'altra. Java ha molte funzioni in grado di gestire i protocolli di comunicazione TCP/IP e le classi che permettono l'uso di Swarm per applicazioni di economia sperimentale sono le RMI, chiamate remote dei metodi. Poiché un programma Java non viene compilato in linguaggio macchina, ma in un byte-code utilizzabile su ogni piattaforma (tramite una Java Virtual Machine), le RMI consentono di stabilire una comunicazione tra oggetti presenti su diverse JVM. Il meccanismo RMI permette in particolare di chiamare i metodi di un oggetto che si trova su di un'altra macchina. I parametri del metodo devono essere comunicati all'altra macchina, all'oggetto deve essere chiesto di eseguire il metodo ed il valore restituito deve essere inviato al client. RMI gestisce tutti questi dettagli.

È così possibile partecipare all'esperimento dal client senza dover installare tutti i programmi necessari per far funzionare SWIEE, ma semplicemente accedendo in remoto al server ed aprire la pagina html che contiene l'applet.

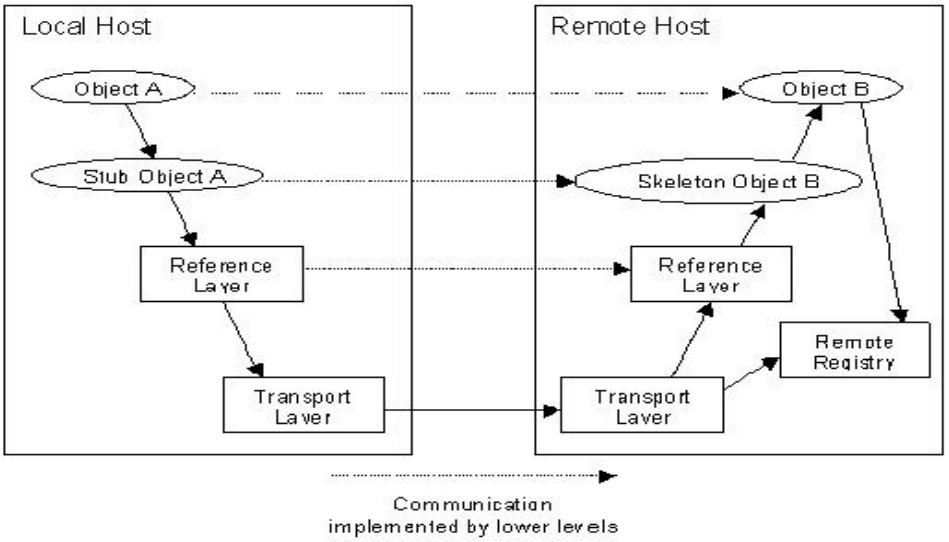
5.2 L'architettura delle RMI

L'aspetto chiave dell'architettura è rappresentato dalla separazione rigida tra *interface* ed *implementation*, infatti il client può accedere solo all'*interface*, mentre l'*implementation* rimane ad uso esclusivo del server.



Il sistema delle RMI completo è composto da molte parti diverse tra loro:

- l'interfaccia per i *remote service*
- l'implementazione dei *remote service*
- i file stub e skeleton
- un server che accolga i *remote service*
- un RMI *Naming* che permetta ai client di trovare i *remote service*
- un programma client che utilizzi i *remote service*
- un http server (un web server), se non si vuole installare programmi sulle macchine client.



5.3 Stub e ordinamento dei parametri

Quando si desidera richiamare un metodo remoto che si trova in un oggetto remoto, il codice client chiama un metodo ordinario di Java che è incapsulato in un oggetto surrogato, detto stub. Lo stub si trova sulla macchina client e impacchetta i parametri usati dal metodo remoto in un blocco di byte. Questa operazione di formattazione dei parametri utilizza, per ciascun parametro, una codifica indipendente dal dispositivo. Gli oggetti sono codificati con il meccanismo di serializzazione. Il processo di decodifica dei parametri viene chiamato ordinamento dei parametri. Scopo di tale ordinamento è la conversione dei parametri in un formato adatto al trasporto da una macchina virtuale all'altra.

Il metodo stub sul client crea un blocco di informazioni contenente:

- un identificatore dell'oggetto remoto da utilizzare
- una descrizione del metodo da attivare.

5.4 Manuale di installazione di SWIEE

L'installazione descritta nel seguente manuale è stata effettuata con le seguenti versioni dei programmi:

- Apache v.1.3 http://httpd.apache.org/dist/binaries/win32
- Swarm v.2.1.1 http://www.swarm.org/release-swarm.html
- JDK v.1.3.0 http://java.sun.com/j2se/1.3/download-windows.html

5.4.1 Apache

Apache è un progetto nato per creare un webserver stabile, affidabile e veloce per piattaforme Unix. Il webserver fa in modo che il PC diventi un server web (solo per quanto riguarda il protocollo HTTP, ovviamente; per altri servizi quali SMTP, FTP, POP e simili sarà necessario rivolgervi ad altri programmi); il webserver non è solo utile, ma addirittura necessario

per l'esecuzione locale degli script: senza di esso, potremo solamente eseguire gli script direttamente dal prompt dei comandi, e non attraverso le pagine web come è necessario in molte applicazioni.

Configurazione di Apache:

Nella directory c:\Programmi\Apache Group\Apache\Conf aprire il file Httpd.conf e controllare la voce "MaxKeepAliveRequests" nel contesto:

```
# MaxKeepAliveRequests: The maximum number of
requests to allow
# during a persistent connection. Set to 0 to
allow an unlimited amount.
# We recommend you leave this number high, for maximum
performance.
MaxKeepAliveRequests 100
```

"100" indica il numero massimo di utenti connessi contemporaneamente, è importante verificare che sia sufficientemente elevato ed è possibile sostituire il numero con quello necessario.

Nello stesso file sostituire alle voci Document Root la root di Swarm.
Per esempio, così:

```
# DocumentRoot: The directory out of which you
will serve your
# documents. By default, all requests are taken
from this directory, but
# symbolic links and aliases may be used to point
to other locations.
DocumentRoot "C:/Swarm-2.1.1"
```
(prima correzione)

```
# Each directory to which Apache has access, can
be configured with respect
```

```
# to which services and features are allowed
and/or disabled in that
# directory (and its subdirectories).
# First, we configure the "default" to be a very
restrictive set of
# permissions.
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

# Note that from this point forward you must
specifically allow
# particular features to be enabled - so if
something's not working as
# you might expect, make sure that you have
specifically enabled it
# below.
# This should be changed to whatever you set
DocumentRoot to.
<Directory "C:/Swarm-2.1.1"> (seconda correzione)
```

5.4.2 Swarm

Swarm è nato nel 1995, nel Santa Fe Institute (New Mexico, USA). L'obiettivo era creare un insieme di programmi e librerie standard, da usare per simulare ed analizzare sistemi complessi di comportamento, nell'ambito delle scienze naturali e sociali.

L'idea era di avere uno strumento che permettesse ai ricercatori di sviluppare le loro applicazioni senza doversi preoccupare di risolvere problemi tipici di programmazione informatica (interfaccia utente, salvataggio ed archiviazione dati, gestione dell'ambiente grafico, ecc.). Inoltre, Swarm fornisce una struttura per simulazioni ed un linguaggio comune che le rende accessibili agli altri ricercatori.

Verifica delle impostazioni di Swarm:

Nella directory c:\Swarm\bin aprire il file jdkcswarm e verificare nella seguente riga di comando che la versione di jdk indicata corrisponda a quella installata.

```
JDKPATH=`regtool get
'HKLM\SOFTWARE\JavaSoft\Java Development
Kit\1.3\JavaHome' 2>/
```

La versione di default indicata è la 1.2, nel nostro caso, quindi, l'abbiamo sostituita con la 1.3.

Nella stessa directory aprire il file jdkswarm e verificare nella seguente riga di comando che la versione di jdk indicata corrisponda a quella installata.

```
JDKPATH=`regtool get 'HKLM\SOFTWARE\JavaSoft\Java
Development Kit\1.3\JavaHome' 2>/
```

La versione di default presente è la 1.2, nel nostro caso quindi l'abbiamo sostituita con la 1.3.

5.4.3 Java 2 Sdk

Il software Java 2 Sdk è un ambiente di sviluppo per la costruzione di applicazioni, applet e componenti che possono essere sviluppate in una piattaforma Java. Tale software include strumenti utili per l'elaborazione e il collaudo di programmi scritti nel linguaggio di programmazione Java. Questi strumenti sono stati progettati per essere utilizzati dalla riga di comando. Solo l'appletviewer possiede un interfaccia grafica.

5.4.4 Autoexec.bat

Nel file Autoexec.bat è necessario verificare la presenza della seguente riga di comando;

```
SET PATH=c:\Programmi\jdk1.3.0_02\bin;c:\swarm-
2.1.1\sum10
```

con jdk installato nella directory c:\Programmi\jdk1.3.0_02

5.5 Sequenza dei comandi per l'avvio dell'esperimento

1) Lanciare il programma Apache (comparirà una finestra dos che servirà passivamente per l'esecuzione dell'esperimento)
2) Dal terminale di Unix, nella directory che contiene i file in java dell'esperimento, compilare gli stessi con la seguente istruzione:
   jdkcswarm *.java
3) (Se la versione di jdk è inferiore alla 1.3 dal prompt di dos è necessario inserire nella directory dell'esperimento il comando
   rmic)
4) Dal prompt di dos, nella directory dell'esperimento, inserire il comando
   start rmiregistry 1098
5) Dal terminale di Unix lanciare
   jdkswarm "nome dell'esperimento"

PARTE SESTA

Esperimenti

6.1 I precedenti esperimenti di Alessandria

Il pioniere in Italia nel campo sperimentale è stato un gruppo di ricercatori dell'università di Trento, che ha dato poi il via a tutta una serie di esperimenti simulativi in diverse università.

Ad Alessandria, in particolare, il laboratorio sperimentale è nato accumulando una certa esperienza grazie alla disponibilità e alla collaborazione dell'università di Trento.

Il primo esperimento ad Alessandria è stato realizzato presso la Facoltà di giurisprudenza nel dicembre 2000 ed ha coinvolto 18 giocatori dislocati in più stanze.

I giocatori erano studenti reclutati nel corso di economia della stessa Facoltà ed hanno partecipato a quattro turni differenti, avendo a disposizione sul gioco solo le informazioni strettamente necessarie.

A causa del ridotto numero del campione scelto, però, i dati raccolti per ciascun set sono stati considerati come parziali e da subito si è prospettata la necessità di approfondire lo studio con ulteriori esperimenti.

Il gioco *Somma 10*, presentato ai 18 giocatori, consiste nel trovare la combinazione giusta delle scelte di ogni componente della squadra per ottenere il massimo guadagno possibile.

Ogni squadra è composta da tre giocatori, rimane invariata per tutta la durata dell'esperimento e l'identità dei suoi componenti è sconosciuta agli altri partecipanti. Ognuno deve scegliere un numero da giocare in una gamma di possibilità messe a disposizione dal computer.

La scelta del numero è condizionata da due diversi ed opposti obiettivi:

- il primo è ottenere che la somma dei tre numeri scelti dal gruppo sia tale da raggiungere il più alto punteggio possibile secondo le regole dettate all'inizio del gioco;
- il secondo è spendere il meno possibile, visto che ogni numero ha un costo che è proporzionato al suo valore (il numero 0 ha costo 0,… , il numero 10 ha costo 10).

I punteggi decisi per l'esperimento sono i seguenti e sono uguali per tutti.

Se la somma dei tre numeri scelti dai membri della squadra è pari a 10, ciascuno vince una cifra pari a 40 meno il numero dichiarato.

Se la somma dei tre numeri è superiore a dieci, ciascuno vince una cifra pari a 30 meno il numero dichiarato.

Se la somma è inferiore a dieci nessuno vince nulla, ma sopporta un costo pari al numero dichiarato.

Lo stesso gioco si ripete per 36 turni.

Sono stati effettuati quattro tipi di esperimenti diversi ognuno con l'intento di svelare un lato del comportamento del giocatore.

	insieme di numeri	**comunicazione numero turni**
Esperimento 1	0, 1, 3, 4, 10 per tutti i giocatori	si
Esperimento 2	0, 1, 3, 4, 10 per tutti i giocatori	no
Esperimento 3	a rotazione 0, 1, 2, 5, 10 0, 1, 5, 6, 10 0, 1, 3, 5, 10	si
Esperimento 4	a rotazione 0, 1, 2, 5, 10 0, 1, 5, 6, 10 0, 1, 3, 5, 10	no

Il primo esperimento prevedeva un set di numeri fissi (0, 1, 3, 4, 10) ed uguali per tutti i giocatori che erano stati messi a conoscenza del numero totale dei turni di gioco.

Il secondo prevedeva, diversamente dal precedente, che i giocatori non sapessero il numero dei turni di gioco, in modo da verificare se il comportamento effettivamente sarebbe cambiato come sostiene la teoria dei giochi.

I primi due esperimenti puntavano su un set di numeri fissi per aiutare i giocatori a raggiungere un accordo; negli altri due, invece, il set di numeri era diverso per ognuno in ciascun turno seguendo un sistema a rotazione.

All'inizio del gioco erano infatti a disposizione del computer tre set di numeri ([0, 1, 2, 5, 10], [0, 1, 5, 6, 10] e [0, 1, 3, 5, 10]), uno per ogni giocatore della squadra; nei turni successivi la sequenza dei tre set ruotava in modo regolare; ogni quattro turni i numeri si ripetevano.

L'apprendimento organizzativo ed il coordinamento sono stati resi più difficili da raggiungere proprio per una maggiore varietà nelle possibilità di scelta.

Il terzo esperimento prevedeva di comunicare ai giocatori il numero dei turni mentre il quarto la escludeva, sempre per verificare il variare delle scelte in funzione delle condizioni.

L'esperimento ha incontrato anche qualche problema. Il più importante è stato la dislocazione delle aule in cui si teneva, perché se da un lato questa dispersione era garanzia di sicurezza e precludeva ogni possibilità di passaggio di informazioni tra i giocatori, dall'altro impediva il controllo ed la coordinazione essenziali per lo svolgimento del gioco.

L'analisi dei risultati ha fornito spunti interessanti di riflessione.

Tab. 1. Media della somma di numeri dichiarati per gruppo

1	11,0
2	10,6
3	11,3
4	10,2

La somma della terna è stata in media sempre superiore a 10, in pratica la paura di ottenere un punteggio negativo li ha spinti a giocare all'eccesso guadagnando un po' meno ma in maniera sicura e costante. I gruppi 2 e 4 hanno medie delle somme più vicine a 10 e sembrano più efficienti, mentre notiamo che le medie dei gruppi 1 e 3 sono decisamente più alte. In teoria conoscendo il numero dei turni ci saremmo aspettati un comportamento meno stabile, soprattutto verso la fine, come fa intendere la teoria dei giochi. Invece le strategie utilizzate dagli studenti sono state coerenti fino alla fine senza mostrare atteggiamenti "egoistici" nell'ultimo turno (non punibili dagli altri compagni-avversari), rispecchiando l'idea generale del bisogno di cooperazione necessaria per vincere.

Tab. 2. Media punti individuali ottenuti in ciascun turno per gruppo

Gruppo 1	26,0
Gruppo 2	28,0
Gruppo 3	24,3
Gruppo 4	32,3

Le medie dei punti individuali risultano essere intorno a 25 punti per turno; solo il quarto gruppo si discosta in maniera significativa dagli altri con una media sorprendentemente superiore ai 30 punti.

La combinazione tra il non sapere il numero preciso dei turni e la presenza di un set di numeri variabile sembra essere ottimale rispetto agli altri casi. Questa spinge maggiormente alla cooperazione e rende più elevate le medie dei punti anche se prevede un comportamento non del tutto efficiente.

Tab. 3. Distribuzione punti individuali ottenuti in ciascun turno per gruppo

	1	**2**	**3**	**4**	**Tot**
-6	0	0	3	1	4
-5	0	0	24	10	34
-4	19	8	0	0	27
-3	33	59	19	8	119
-2	0	0	22	11	33
-1	6	11	33	14	64
0	53	15	10	4	82
20	74	40	33	7	154
24	0	0	21	2	23
25	0	0	157	45	202
26	74	62	0	0	136
27	35	50	51	14	150
28	0	0	40	4	44
29	6	9	15	7	37
30	57	12	11	2	82
35	0	0	72	173	245
36	81	126	0	0	207
37	162	252	66	173	653
38	0	0	66	173	239
40	48	4	5	0	57

Tab. 4. Varianza dei punteggi individuali ottenuti in ciascun turno per gruppo

1	194,4
2	182,1
3	176,0
4	112,5

A completamento dell'analisi sulla maggiore efficienza dei gruppi 2 e 4, notiamo che i punteggi, oltre ad avere medie più efficienti e più elevate, hanno minore varianza. In generale, un po' sorprendentemente, c'è minore varianza proprio nei gruppi in cui sembrava più difficile coordinarsi.

Tab. 5. Distribuzione dei numeri dichiarati dai singoli giocatori nel corso del gioco per gruppo

Cifra	**1**	**2**	**3**	**4**	**Tot**
0	134	29	25	6	194
1	12	20	48	21	101
2	0	0	128	188	316
3	230	361	136	195	922
4	174	196	0	0	370
5	0	0	253	228	481
6	0	0	24	3	27
10	98	42	34	7	181

La selezione dei numeri è condizionata dalla scelta di una determinata strategia. Infatti troviamo nei primi turni (con il set fisso) con una maggiore frequenza lo 0 e il 10 ed i numeri centrali 3 e 4, mentre le strategie con i set variabili sono più complesse ed articolate e lo comprova il fatto che i numeri hanno distribuzione più varia.

Tab. 6. Distribuzione delle somme delle terne in ciascun turno

	1	**2**	**3**	**4**
0	11	1		
1			1	
3	2	2		1
4	4	2	1	
5			2	2
6	2	4	7	2
7	4	6	2	1
8	7	4	15	6
9	7	12	9	4
10	105	128	70	173
11	25	22	19	4
12	7	1	21	4
13	8	2	36	11
14	3	8	6	3
15		2	8	2
16	2	8	1	
17	1	8	3	1
18	5	2	8	
20	18	2	4	2
21		1	2	
23	1		1	
24		1		
30	4			

Media per gruppo delle varianze delle scelte di ciascun giocatore

8,311	3,214	4,072	2,246

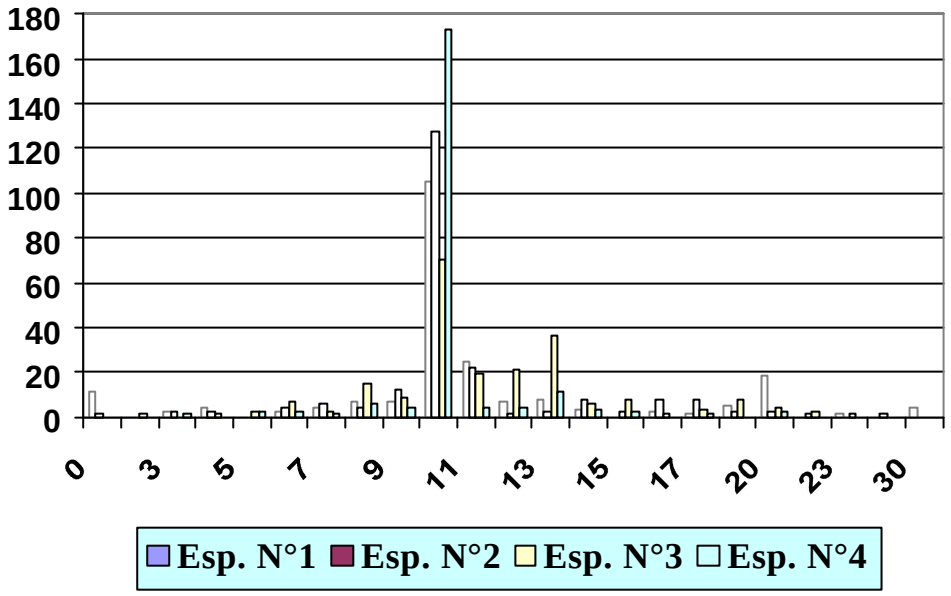
6.1.1 L'analisi generale dei dati

La lettura dei risultati può essere fatta in due modi:

1) Confronto tra esperimenti in cui il set di numeri è fisso (1 e 2) ed esperimenti in cui il set di numeri è variabile (3 e 4)

A priori non era affatto scontato che i soggetti riuscissero a coordinarsi, dato che l'insieme dei numeri variava in continuazione. A posteriori invece questo si è rivelato un buon modo per "guidare" i giocatori verso una strategia più cooperativa e meno egoistica.

Distribuzione delle somme



Negli esperimenti 3 e 4 i soggetti raggiungono risultati più efficienti ed equi. Infatti c'è un numero maggiore di casi in cui la somma risulta pari a 10 (e un numero maggiore di casi in cui la somma è pari o superiore a 10, cioè di casi in cui si ottiene un payoff positivo).

Nei turni 3 e 4 la somma non risulta mai pari a 0 (infatti ci sono pochissimi casi in cui i giocatori dichiarano 0), è maggiore la media dei punti ed è minore la varianza.

2) Confronto tra esperimenti in cui i giocatori conoscevano il numero di turni (1 e 3) ed esperimenti in cui non lo conoscevano (3 e 4)

In questo caso le differenze sono ancora più nette di prima: i giocatori che non conoscono il numero di turni raggiungono risultati più efficienti ed equi.

La media delle somme raggiunte nei vari turni è più vicina a 10 per i gruppi 2 e 4. La media di punti è nettamente più alta e la varianza più bassa. Ci sono molti più turni in cui la cifra dichiarata è pari a 10 (o maggiore di 10).

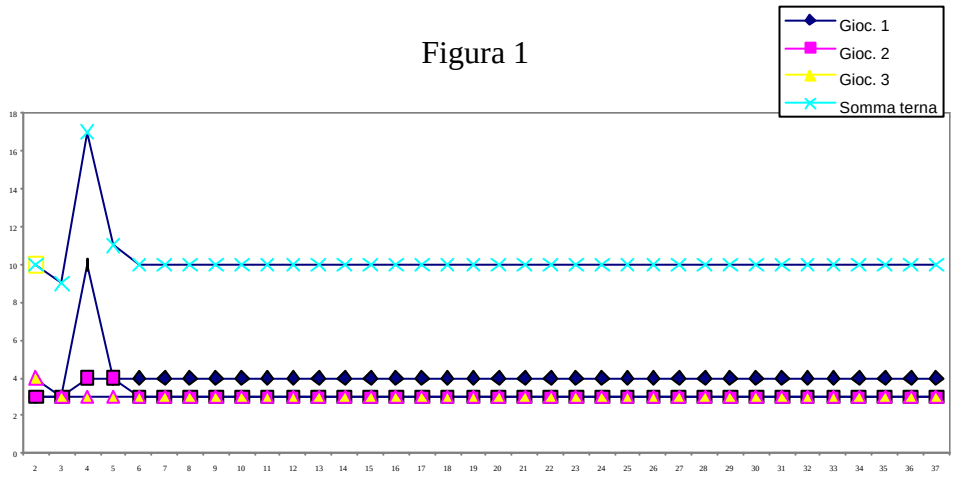
Negli esperimenti 1 e 3 non si registrano però variazioni particolarmente significative nei turni finali (la maggiore efficienza dei gruppi 3 e 4 si manifesta lungo tutto il gioco e sembra essere causata solo parzialmente da comportamenti strategici nei turni finali).

6.1.2 Le diverse strategie

Le strategie nascono in corrispondenza delle situazioni che si presentano alla terna.

Ogni terna sceglie la strategia che ritiene migliore o adotta quella su cui i tre si accordano prima.

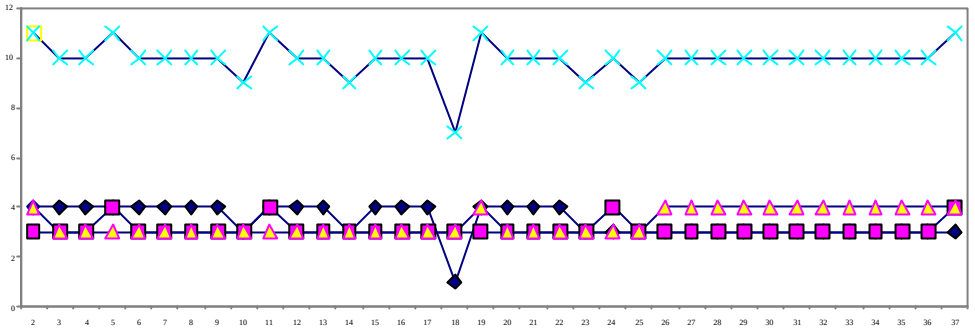
Figura 1



Gli equilibri stabili raggiunti dai terzetti negli esperimenti 1 e 3 prevedono che due giocatori dichiarino sempre (per tutti i turni) 3 e che un giocatore si sacrifichi dichiarando sempre 4 (come in figura 1). Non ci sono quindi equilibri stabili in cui i giocatori si alternano a dichiarare 3 e 4.

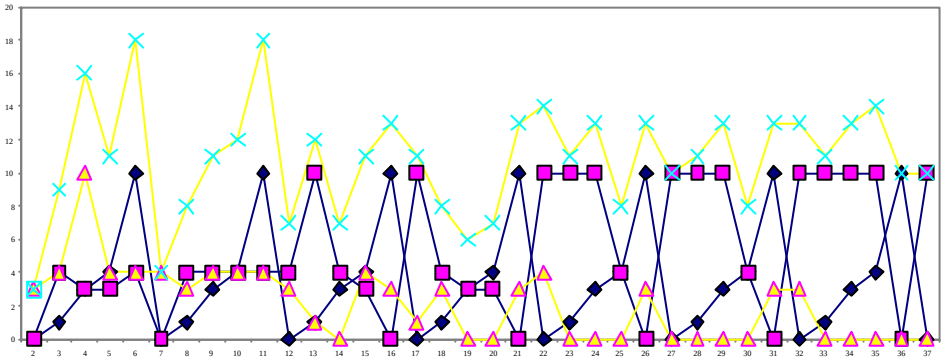
I casi in cui hanno provato ad alternarsi nel sacrificio ci sono stati, ma hanno portato a risultati inferiori per tutti (come in figura 2)

Figura 2



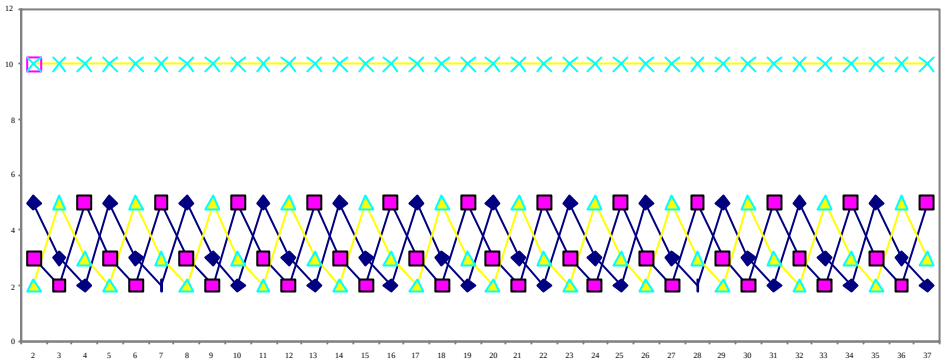
Non ci sono nemmeno equilibri in cui i giocatori si alternano a dichiarare 10-0-0 (in linea generale non è più difficile da ottenere di un equilibrio 4-3-3, ma comporta evidentemente un maggiore rischio), ma solo sporadici tentativi falliti (come in figura 3).

Figura 3



Nel 3° e 4° gruppo i giocatori sono stati costretti a variare la propria scelta di turno in turno, così l'unico modo per ottenere un punteggio ottimale era quello di accordarsi e cooperare. L'idea di partenza era quella di costringere la squadra ad alternarsi nel sacrificio e molti hanno subito capito qual'era la strategia giusta. Trovare l'intesa restava comunque una cosa complicata, ma l'immediato accordo della terna in figura 4 ha del miracoloso (erano in tre aule diverse).

Figura 4



6.1.3 Possibili estensioni del gioco

Una possibile estensione in grado di mostrare le potenzialità di SWIEE riguarda l'utilizzo di agenti artificiali negli esperimenti. I vantaggi dell'introduzione di agenti non umani sono vari:

1) Si può pensare di sottoporre gruppi diversi di individui ad esperienze diverse (facendoli giocare con agenti artificiali diversi) e poi valutare gli effetti di tale "formazione" in una fase successiva del gioco.

2) Gli agenti artificiali possono anche svolgere un altro compito. Come ricordato, in questo esperimento entrano in gioco due aspetti: altruismo e ricerca di coordinamento. Se l'individuo sa di interagire con un agente artificiale (che utilizza una data strategia) difficilmente sarà motivato dall'altruismo e quindi il suo comportamento sarà guidato dalla ricerca di un payoff.

Alcuni individui ad esempio accettano payoff costantemente più bassi degli altri. Tale scelta è motivata da forme di altruismo "sacrificarsi per il bene della squadra" oppure da forme di razionalità non troppo dissimili da quelle previste dalla teoria in relazione ad esempio all'ultimatum game (meglio guadagnare qualcosa che nulla)?

Da una prima analisi si può constatare che con l'esperienza il punteggio migliora. Dopo un primo periodo di assestamento per capire come funziona fisicamente il gioco e per imparare ad applicare le regole

spiegate a voce, hanno potuto constatare che modificare il set di possibili scelte migliora leggermente la capacità di coordinamento.

6.2 L'esperimento di Torino

A Torino abbiamo effettuato un esperimento, modificato rispetto a quello di Alessandria, con un set di valori fissi (0,1,3,4,10) introducendo una fase di "apprendimento" con agenti artificiali. Il gioco mantiene le regole dell'esperimento precedente, ma si divide in due parti: nella prima interagiscono agenti umani con giocatori artificiali guidati da un algoritmo, nella seconda i gruppi vengono ricomposti con i soli agenti umani.

La fase di apprendimento è stata inserita per verificare se fosse possibile influenzare la strategia che il giocatore umano avrebbe applicato nella seconda fase del gioco.

Il comportamento che hanno tenuto gli agenti artificiali (altruisti o egoisti) è stato studiato a tavolino per permetterci di valutare se la fase di apprendimento:

- non aveva nessun effetto sul comportamento
- se dopo aver giocato con agenti egoisti si adottavano atteggiamenti altruisti
- se dopo aver giocato con agenti egoisti si preferiva un comportamento egoista.

Per poter scongiurare eventuali errori nella lettura dei risultati, abbiamo inserito un piccolo questionario prima dell'esperimento per testare la comprensione del gioco.

- Sapendo che gli altri due partners giocheranno rispettivamente 0 e 10, che numero dichiari?
- Sapendo che gli altri due partners giocheranno rispettivamente 3 e 4, che numero
dichiari?
- Sapendo che gli altri due partners giocheranno entrambi 0, che numero dichiari?
- Sapendo che gli altri due partners giocheranno 3, che numero dichiari?

Queste semplici domande sono state utili per valutare anche quanto tempo era necessario per poter comprendere a pieno il funzionamento del gioco.

Quando abbiamo pensato a come spiegare il gioco abbiamo prestato molta attenzione al significato di ogni parola utilizzata per poter essere il più chiaro possibile nel descrivere le regole e le finalità

Nonostante le particolari attenzioni rivolte a questa fase dello svolgimento dell'esperimento, dall'analisi delle risposte date prima di cominciare sono venute alla luce numerose incomprensioni. Purtroppo infatti quando si spiegano le regole di un gioco si crea nella mente di chi ascolta una sorta di deformazione del gioco stesso: alcuni sono più attenti o più disposti a capire e comprendono il giusto funzionamento subito, alcuni intuiscono male e ci mettono parecchi turni a correggere la falsa idea che hanno del gioco, altri continuano fino alla fine a giocare ad un altro gioco.

Innanzitutto era indispensabile appurare se nella fase di apprendimento i giocatori umani avessero seguito gli agenti artificiali adattandosi alla loro strategia per poter valutare le eventuali conseguenze.

Dai dati si può riscontrare una generale disponibilità all'adattamento del giocatore umano al gioco degli artificiali, anche se questo non si riflette quasi mai nel gioco tra umani. Il fatto che non sapessero della presenza di agenti artificiali, fa pensare che adottare una strategia semplice o giocare un numero fisso aiuti a "scegliere" la propria strategia, altruista o egoista che sia.

6.2.1 Gli agenti artificiali

Ogni giocatore umano nella prima fase (15 turni) era affiancato da 2 agenti artificiali. La metà dei giocatori ha compiuto l'apprendimento con una coppia di agenti altruisti (uno forte e l'altro debole) e l'altra metà con agenti egoisti (sempre uno forte e l'altro debole).

Abbiamo inserito due agenti diversi tra loro per rendere meno prevedibile il loro comportamento e per permettere alla terna di raggiungere con più combinazioni 10 ottenendo il massimo punteggio.

- *Altruista debole*:

```
        if (timer.cycles == 0) {
          b = 3;
        }
        else {
            if ((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2) < 10) {
              if ((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2 + 1) < 10){
                if ((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2 + 3) < 10){
                  if ((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2 + 4) < 10){
                    b = 10;
                  }
                  else
                    b = 4;
                }
                else
                  b = 3;
              }
              else
                b = 1;
            }
            else
               b = 0;

        }
           }
```

Al primo turno gioca 3, poi gioca adattandosi a ciò che succede nel turno precedente per arrivare almeno a 10. Se, per esempio, i suoi due compagni hanno giocato 0 e 4 il suo algoritmo lo guiderà a giocare 10.

- *Altruista forte*: gioca sempre 4.

- *Egoista debole*:

```
   if (timer.cycles == 0) {
      b = 3;
   }
   else {
      if (((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2) < 7 ||
(a.playerArray[a.id1].choice2 + a.playerArray[a.id2].choice2)
> 10)){
         b = 0;
      }
      else
         b = 3;
      }
   }
```

Al primo turno gioca 3, poi se la somma dei due compagni nel turno precedente è <7 o >10 gioca 0 (infatti se fosse <7 dovrebbe

giocare troppo alto per un agente egoista, se fosse >10 cercherebbe di ridurre al minimo la spesa), altrimenti gioca 3.

- *Egoista forte*: gioca sempre 3.

6.2.2 Variazioni nel programma

Oltre all'inserimento degli agenti artificiali il programma ha subito numerose variazioni, alcune sono state inserite nella versione finale utilizzata, altre a causa dei problemi riscontrati con i PC saranno utilizzate solo nei prossimi esperimenti.

1) Era stato riscontrato un problema di perdita di concentrazione nei soggetti all'esperimento precedente tenuto ad Alessandria, che ha rallentato notevolmente lo svolgimento allungando i temi di esecuzione del singolo turno. Infatti i soggetti dovevano effettuare manualmente l'aggiornamento dei dati dopo aver effettuato la scelta, ripetendo l'operazione più volte fino a quando tutti non avevano inviato la propria scelta.

Abbiamo così modificato nel Sum10Client il sistema di trasmissione dei dati verso il server, ma soprattutto abbiamo automatizzato l'aggiornamento dei risultati al fine di ogni turno per dare il via libera ad una nuova scelta a tutti i giocatori contemporaneamente.

```
boolean check = false;
while (!check) {
long t = System.currentTimeMillis() + 5000;
textArea1.setText("Attendere.");
while (System.currentTimeMillis() < t) ;
if (!(askBlock2())) {
update();
check = true
```

Una volta effettuata la scelta, utilizzando l'orologio interno del computer su cui si sta operando, ogni 5 secondi il client chiederà il via libera

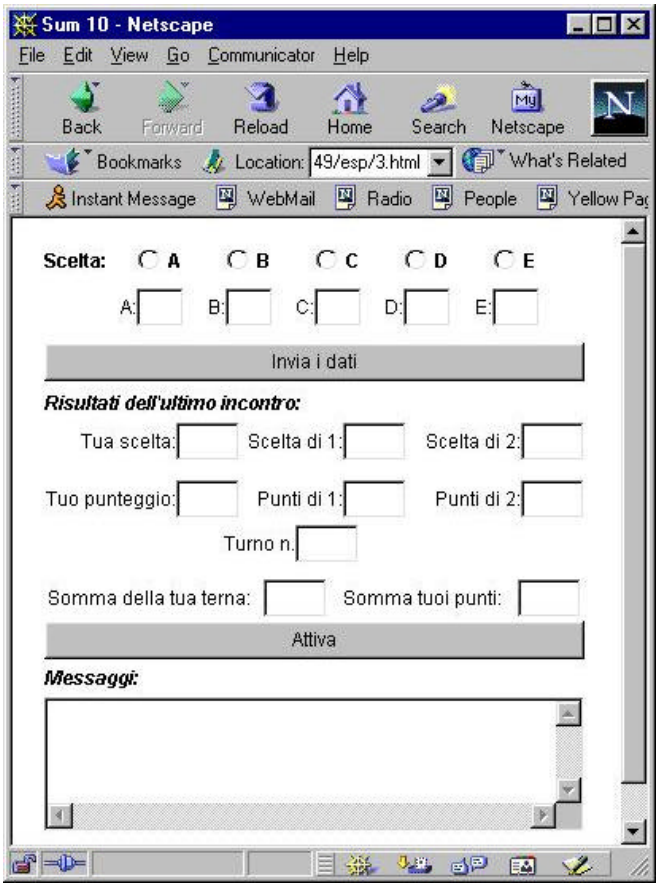
al server per iniziare un nuovo turno. Solo quando il server avrà raccolto le scelte di tutti aggiornerà i risultati del gioco sullo schermo e permetterà di effettuare una nuova scelta. Questo automatismo è stato "sacrificato" per rendere l'applet più leggero possibile, ma ha dato buoni risultati nelle prove effettuate riducendo i tempi di attesa e di completamento dell'esperimento.

2) Un'altra modifica apportata al programma la troviamo nel Player, dove abbiamo affiancato ad agenti umani gli agenti artificiali. Gli uni giocano con gli altri contraddistinti solo dal numero identificativo (id) dato loro al momento della creazione. Ogni agente segue due differenti passi: il primo permette all'agente guidato dall'umano di scegliere liberamente e all'agente virtuale di interpellare il RuleMaster, il secondo controlla che tutti, artificiali ed umani, abbiamo trasmesso la loro scelta.

3) La modifica più importante è stata l'introduzione del RuleMaster che "suggerisce" all'agente virtuale il comportamento da tenere secondo strategie di gioco precise formalizzate dagli algoritmi (studiati e progettati a tavolino per tale scopo) presentati sopra. Anche in questo caso esiste un solo RuleMaster per tutti gli agenti e solo il numero identificativo permette al singolo agente di seguire una precisa strategia.

4) Dopo numerosi tentativi abbiamo deciso di dotare l'applet di un pulsante per l'attivazione. Ogni applet viene inizializzato nel momento in cui viene richiamato; prima l'attivazione avveniva nel momento in cui l'applet veniva caricato dal client e rimaneva attivo anche nei tempi tecnici per la sistemazione in aule dei giocatori e per la spiegazione del gioco. Sui computer di Torino gli applet non reggevano il tempo necessario al completo svolgimento dell'esperimento: dopo averlo alleggerito al massimo, abbiamo deciso di renderlo attivo solo per il tempo strettamente necessario.

Potevamo fare caricare l'applet al giocatore al momento giusto, ma abbiamo preferito controllare noi l'apertura in modo che fosse pronto per tutti nello steso momento. Con la presenza del pulsante per l'attivazione è stato possibile spiegare il gioco con l'applet visualizzato sullo schermo dei client davanti ai giocatori e attivarlo solo all'inizio del gioco.



Le informazioni a disposizione del giocatore comprendevano, oltre al punteggio totale e al numero del turno di gioco, anche i dati relativi al turno precedente, con le scelte della terna ed i relativi punteggi.

6.2.3 Strategie di gioco

In fase di programmazione avevamo previsto la nascita di strategie precise tra agenti umani, ma durante il gioco se ne sono spontaneamente create delle altre concettualmente diverse.

Avevamo ipotizzato due strategie possibili:

- una era di giocare un numero fisso durante tutto l'esperimento (giocatori egoisti ed altruisti);
- l'altra era di coordinarsi per giocare a turno il numero più elevato.

Dall'analisi dei risultati si possono invece individuare anche altre strategie oltre a quelle previste:

- c'era chi giocava "con la squadra" per raggiungere il massimo punteggio possibile, anche sacrificandosi per più turni a giocare un numero più elevato;
- c'era chi invece giocava "contro la squadra" cercando di spendere il meno possibile senza preoccuparsi di raggiungere 10.

All'interno delle strategie cooperative si possono individuare due diverse terne di numeri giocate durante l'esperimento: 4,3,3 e 10,0,0. L'alternanza nel giocare il numero più alto non si è verificata in maniera sistematica, anche se è stata tentata, per la difficoltà nel passare da un equilibrio (4,3,3…) all'altro (3,4,3…). Hanno ottenuto punteggi più elevati infatti le terne in cui uno dei giocatori ha accettato di "sacrificarsi" per il bene (suo e) della squadra spendendo più degli altri.

Chi ha giocato "contro la squadra" ha sistematicamente giocato 0 senza dare nessun apporto personale al punteggio ottenuto dalla terna. Non può essere considerato un atteggiamento da egoista puro, come da noi ipotizzato, perché non è riscontrabile il tentativo di massimizzare il profitto minimizzando i costi, ma semplicemente di spendere il meno possibile.

6.2.4 Preparativi all'esperimento

Il progetto che ha ispirato il nostro esperimento è partito dalla Facoltà di Giurisprudenza di Alessandria dove sono stati effettuati i primi esperimenti simulativi l'anno scorso.

Abbiamo collaborato con il dott. Novarese del Dipartimento di scienze giuridiche ed economiche dell'Università del Piemonte orientale fin dall'inizio e questo ci ha permesso di acquisire la loro esperienza nel campo degli esperimenti evitando di partire da zero ed incappare in errori di ingenuità già riscontrati precedentemente.

Per effettuare l'esperimento è stato utilizzato Swarm come piattaforma informatica, ed in particolare il programma SWIEE sviluppato dal dott. Boero, laureato all'Università di economia di Torino.

SWIEE si basa sulla combinazione tra la solidità e la praticità di Swarm e la versatilità del linguaggio di programmazione Java, risultando

uno strumento perfetto come base per la programmazione di ogni esperimento economico.

Per svolgere l'esperimento presso la sede della Facoltà di economia di Torino abbiamo usufruito dell'aiuto del dipartimento di scienze economiche e finanziarie "Giuseppe Prato" per organizzare l'esperimento e delle aule del Liases per effettuarlo.

Avendo scelto di ricercare comportamenti altruistici e cooperativi attraverso una variante del gioco *Somma 10*, siamo stati indecisi se coinvolgere studenti del primo anno con poca esperienza di teorie economiche o studenti più "esperti" del terzo anno.

Alla fine la scelta è ricaduta su una terza possibilità, ovvero abbiamo coinvolto entrambe le classi.

In questo modo abbiamo utilizzato un campione più omogeneo ed è stato possibile analizzare i risultati ottenuti anche in base all'esperienza del singolo giocatore, per valutare se questa influenza o meno le scelte all'interno del gioco.

Abbiamo potuto coinvolgere così più giocatori grazie alla promozione fattaci da alcuni professori del dipartimento tra gli studenti dei loro corsi di economia.

All'inizio abbiamo chiesto la collaborazione dei docenti più vicini all'esperimento:

-il professor Bagliano per Economia monetaria del terzo/quarto anno

-il professor Frigero per Economia politica 1 del primo anno

-il professor Sembenelli per Economia applicata e Econometria del terzo/quarto anno

-il Professor Terna per Tecniche di previsione economica (reti neurali) e Dinamica economica del terzo e quarto anno.

Poi abbiamo allargato l'invito ad altri corsi avendo riscontrato che le adesioni erano scarse e procedevano a rilento:

-la professoressa Fornero per Economia politica 2 del secondo anno

-il professor De Battistini per Economia politica 2 del secondo anno

-il professor Bortolotti per Economia delle istituzioni (non profit) del terzo/quarto anno

-il professor Siniscalco per Economia politica del primo anno

L'annuncio del gioco e la spiegazione delle condizioni per la partecipazione è stata fatta agli studenti ben tre mesi prima dell'esperimento.

A tutti è stato infatti letto in aula lo stesso regolamento per evitare di creare disparità nella conoscenza delle informazioni.

Il contenuto del "comunicato" è stato preparato, studiato e discusso in una riunione con tutti gli organizzatori, avendo cura di non svelare il funzionamento del gioco (che è stato descritto solo prima dell'esperimento), ma di incuriosire gli studenti ed invogliarli a partecipare senza creare in loro false illusioni.

Il testo è il seguente:

Esperimenti economici al Liases

Giovedì 3 maggio 2001, dalle ore 18 in poi, si terrà presso le aule del Liases della Facoltà di Economia in corso Unione Sovietica 218 bis, un esperimento economico collegato a due tesi di laurea.

La sperimentazione consisterà in un gioco a punteggio, da effettuare su computer, della durata di circa un'ora.

I migliori classificati riceveranno premi in buoni libri (proporzionati ai risultati ottenuti, per un massimo di 30.000 lire a testa) da spendere presso la libreria Celid all'interno della Facoltà.

Non sono richieste conoscenze specifiche.

Chi fosse interessato è pregato di dare la propria adesione entro il 31 marzo.

Verranno in seguito pubblicati gli elenchi con l'indicazione precisa dell'ora e dell'aula, affiggendoli presso le bacheche del Dipartimento di scienze economiche e finanziarie G. Prato e del Palazzo del lavoro.

*Per chiarimenti rivolgersi al docente relatore delle tesi in questione, Pietro Terna **pietro.terna@unito.it**, o agli studenti che svolgono le tesi: Andrea Gianoglio **andrea.gianoglio@tin.it** e Filippo Pavesio **f.pavesio@tin.it**.*

Allegato al comunicato c'era un foglio di adesione sul quale abbiamo potuto raccogliere i primi dati sugli studenti disposti a svolgere l'esperimento.

Foglio di adesione *(scrivere in stampatello)*

Nome	***Cognome***	***Matricola***	***Indirizzo e-mail (preferito) o recapito telefonico***

Fortunatamente la maggior parte degli studenti che ha aderito all'esperimento ha indicato come recapito un indirizzo e-mail; in questo modo è stato possibile anche mandare diversi avvisi per sollecitare la loro partecipazione, come previsto in fase di progettazione.

Il primo avviso è stato spedito una settimana prima dell'esperimento volendogli ricordare solamente l'impegno preso; insieme alle e-mail abbiamo dovuto spedire anche numerosi Sms per contattare coloro che avevano indicato solo il numero di cellulare.

Purtroppo il numero dei partecipanti stimato dagli elenchi è diminuito perché alcuni hanno risposto al nostro messaggio scusandosi per non poter partecipare e alcuni indirizzi sono tornati al mittente essendo inesistenti.

Infatti molti degli indirizzi in corsivo erano poco leggibili probabilmente per via della fretta con cui erano stati scritti e in fase di riscrittura avevamo notato questa difficoltà avendo ipotizzato che alcuni potessero essere sbagliati.

Il secondo avviso è stato spedito il giorno prima della data fissata per l'esperimento con l'indicazione precisa dell'ora per ciascun gruppo. Abbiamo dato per scontato che tutti leggessero la posta quotidianamente, forse analizzando la cosa a posteriori avremmo dovuto anticipare almeno di un giorno l'ultimo avviso. Infatti ho personalmente raccolto le lamentele di alcuni che, pur essendo venuti all'ora indicativa data nel primo messaggio, sostenevano di non aver ricevuto il secondo non avendo letto la posta il giorno prima.

Abbiamo dovuto riflettere anche sulla motivazione da dare ai giocatori per rendere credibile il risultato dell'esperimento.

L'esperienza di Alessandria anche in questo caso ci è stata molto utile come termine di paragone; infatti in quel caso l'esperimento è stato inserito come seminario all'interno di un corso di economia della facoltà di giurisprudenza.

In questo modo la partecipazione all'esperimento è diventa parte integrante del programma del corso ed il suo risultato ha influenzato in maniera lieve, ma allettante il voto finale.

Per il nostro esperimento abbiamo deciso di muoverci in maniera differente.

Eravamo infatti indecisi sullo strumento da utilizzare quando, fortunatamente, ci sono venuti in aiuto il codice deontologico dell'economista sperimentale e le indicazioni della letteratura in materia, che giudicano la leva economica la migliore.

Utilizzando premi in denaro si rende la ricerca e l'esperimento credibile anche al di fuori del contesto universitario locale.

Per evitare di dover pagare i premi in contanti abbiamo pensato di mettere in palio buoni libri da ritirare presso la libreria della Facoltà.

La scelta del valore da assegnare alle singole ricompense ci ha posto di fronte ad un nuovo bivio.

Non sapevamo se dividere i vincitori in classi di merito in base al punteggio finale ottenuto e pagare cifre prestabilite o ripagare le vincite in maniera proporzionale.

Poiché il punteggio finale dipendeva non solo dall'abilità del singolo, ma soprattutto dalla capacità di coordinamento del gruppo, appresa nei turni di training, e la graduatoria invece era personale, non volevamo svantaggiare troppo coloro che giocavano con compagni meno motivati o meno disposti a collaborare e così alla fine abbiamo deciso di utilizzare le classi di merito.

Scelto il metodo ci restava solo da decidere il valore dei singoli premi in maniera da invogliare i giocatori al massimo impegno e ricompensare il maggior numero di persone possibile.

Avevamo ripartito i fondi a nostra disposizione in questo modo:

- 8 premi da £30.000
- 8 premi da £15.000

14 premi da £10.000.

Visto poi che nelle due aule si sarebbero svolti due esperimenti leggermente diversi e che le informazioni date ai due gruppi di giocatori sarebbero state diverse, non ci sembrava giusto compilare una sola graduatoria per le due aule.

Infatti la differenza nelle informazioni possedute o nelle condizioni in cui si effettuava la fase di apprendimento poteva condizionare lo svolgimento ed il risultato finale del gioco.

Per questo abbiamo deciso di stilare due graduatorie diverse, una per aula, e premiare i migliori delle due classifiche (4 da £30.000, 4 da £15.000 e 7 da £10.000 per aula).

Scelto il campione abbiamo dovuto organizzare l'esperimento vero e proprio, con tutti i problemi tecnici legati alla realizzazione simultanea del maggior numero possibile di casi.

Grazie alla disponibilità del prof. Margarita, responsabile del Liases, non solo abbiamo avuto a disposizione le sale informatiche dove svolgere fisicamente l'esperimento con più di cinquanta computer in rete, ma abbiamo usufruito anche del supporto pratico dei tecnici, utile per la risoluzione dei non pochi problemi informatici sorti durante la preparazione dallo stesso.

Le aule che hanno ospitato l'esperimento contengono ognuna venticinque computer numerati ed identificabili su quattro file che sono collegati in una rete locale condivisa da tutte le macchine della facoltà.

Il server utilizzato è un Pentium 200 con 64 MB di Ram, visto che il programma non richiede una grande velocità di esecuzione, mentre i computer su cui "giocavano" gli studenti sono Pentium 100.

All'inizio pensavamo di dividere l'esperimento in due sessioni da svolgersi in due giorni diversi per cercare di coinvolgere più giocatori possibili.

Poi abbiamo deciso di fare tutto in un giorno solo con due turni consecutivi. Abbiamo convocato i giocatori del secondo turno circa un'ora dopo rispetto a quelli del primo turno e li abbiamo trattenuti in una stanza adiacente a quella dell'esperimento. In questo modo quelli di un turno

completavano il gioco e lasciavano l'aula senza incontrare i giocatori del successivo impedendo così il passaggio di informazioni.

Temevamo infatti che il contatto tra i "turni" alterasse i risultati condizionando il comportamento nel gioco.

Per progettare e realizzare l'esperimento abbiamo collaborato con il dott. Novarese, che ci ha informati del lavoro svolto dal suo gruppo, dei risultati ottenuti e delle analisi sui dati effettuate.

In questo modo abbiamo potuto individuare alcuni temi da approfondire e concentrare i nuovi test su obbiettivi precisi.

6.2.5 Problemi nello svolgimento dell'esperimento

L'esperimento lo si può preparare a tavolino, ma non sempre tutto quello che si è programmato si svolge senza intoppi, e non tutti i problemi che ci si è preparati ad affrontare puntualmente si verificano.

Infatti pensavamo di trovarci di fronte un problema di partecipazione, non essendo affatto sicuri della coerenza di coloro che si erano iscritti negli elenchi e che erano stati più volte contattati da noi via e-mail o sms. Invece l'affluenza è stata ottima e qualitativamente molto positiva, infatti la maggior parte sembrava motivata e impaziente di giocare.

La leva economica è sicuramente servita come stimolo, ma molti studenti erano curiosi e interessati all'esperimento i sé; ne è la prova che molti alla fine mi hanno chiesto spiegazioni sul suo funzionamento e sui codici del programma; molti inoltre erano interessati anche alle analisi dei risultati.

Tra loro alla fine discutevano di strategie e di ipotesi sullo svolgimento del programma mostrando un interesse per noi sorprendente.

Sono arrivati all'ora prevista praticamente tutti quelli convocati per il primo turno e sono stati fatti accomodare in un'aula adiacente al Liases preparata per l'occasione.

Nell'assegnazione dei posti abbiamo fatto particolare attenzione a non mettere vicini i componenti di una stessa squadra, sparpagliandoli per le aule, per impedire che si influenzassero.

Il primo turno era composto da 18 persone, 9 per aula.

Hanno compiuto senza intoppi due turni interi, poi il programma si è bloccato impedendo il proseguimento; abbiamo tentato di riavvare la simulazione, ma è si bloccata nuovamente in entrambe le aule dopo pochi turni.

È stato deciso allora di effettuare l'esperimento a gruppi di sei per non appesantire il programma e poter controllarne meglio lo svolgimento.

Purtroppo i primi 18 concorrenti erano a quel punto "compromessi" e non hanno potuto partecipare nuovamente all'esperimento. Non ci sembrava giusto escluderli dalle graduatorie solo per un intoppo nell'esecuzione del programma, così abbiamo deciso di dar loro un premio di partecipazione di £10000, dovendo così riprogrammare i premi.

Abbiamo assegnato ai due gruppi 3 premi da £30000 e 4 da £15000 eliminando quelli da £10000 (in pratica già destinati alle prime "cavie").

Dovendo ristabilire i turni il nostro timore era che la maggior parte dei concorrenti non fosse disposta ad aspettare fino a tardi per completare il proprio. Invece data loro la "cattiva notizia" hanno fatto la fila per iscriversi nei cinque turni da noi previsti. Se avessimo avuto maggiore tempo saremmo riusciti a far giocare tutti; purtroppo molti hanno, a malincuore, dovuto rinunciare a partecipare.

Molti hanno aspettato fino alle 21 per giocare senza mostrare segni di insofferenza, anzi apparendo divertiti e partecipando attivamente al gioco.

Il problema principale che ha condizionato pesantemente lo svolgimento del gioco è legato alle macchine utilizzate. Infatti, come già spiegato, il programma gira fisicamente su un server e le "stazioni di gioco" si collegano a questo per aprire l'applet personalizzato per ciascuna; le comunicazioni tra i due computer avvengono tramite la rete della facoltà.

La lentezza delle "stazioni" ha condizionato il passaggio di informazioni a tal punto da bloccare Netscape ed impedire praticamente il corretto svolgimento dell'esperimento.

Probabilmente il problema è legato al *timeout* che blocca l'applet a causa del ritardo con cui si aggiorna dal server, ma non c'è modo di sapere con certezza assoluta se sia questa la reale causa. Certo è che questo inconveniente non era stato riscontrato con computer leggermente più

potenti, pur dovendo reggere applet molto più "pesanti", che non hanno mostrato nessuna difficoltà a portare a termine l'esperimento.

Le ipotesi sul problema di fondo sono state ridotte, dopo un'attenta analisi della situazione, a due. La prima delle due riguarda la versione molto vecchia di Netscape presente sui computer in aula, ma non sui server utilizzati che sono stati integralmente preparati da noi con le ultime versioni dei programmi. Questa si è ha portata dietro tutta una serie di programmi satellite tra i quali la *Java virtual machine*; nel nostro esperimento infatti permette di ricreare un ambiente virtuale che regge l'applet scritto in linguaggio Java. Il fatto che sia di vecchia concezione impedisce ai dati di fluire attraverso la rete bloccando l'applet stesso.

La seconda ipotesi è direttamente legata alla lentezza delle macchine usate ed in particolare alla quantità di Ram presenti, solo 16, che in fase di progettazione avevano considerato sufficienti.

Alla fine la combinazione di questi e di altri problemi, magari non riscontrati direttamente, hanno impedito la completa realizzazione del nostro progetto.

Alla fine, dopo aver registrato un gran numero di piccoli o grandi problemi, siamo riusciti a completare l'esperimento con 27 persone.

I primi due gruppi hanno effettuato i turni di training con agenti virtuali egoisti, mentre i seguenti due con agenti altruisti.

L'ultimo gruppo è stato messo a conoscenza del fatto che durante il periodo di training avrebbero giocato con agenti virtuali, mentre agli altri è stato tenuto nascosto.

6.2.6 I dati raccolti

I dati raccolti durante l'esperimento sono stati trascritti in diversi file a seconda del turno di gioco e del server che lo reggeva. I due server sono stati denominati con le lettere (*a* e *b*) e i turni con i numeri (da 1 a 4 per il server *a* e da 1 a 5 per il *b*).

Da quei file sono state tratte le seguenti tabelle riassuntive. Sono state elencate le scelte dei soli agenti umani, i primi quindici turni quindi riportano dati riferiti a tre terne diverse.

LEGENDA

- *OBS*: indica il numero dell'osservazione.
- *Turno*: indica il numero del turno a cui si riferiscono i dati della riga.
- *Somma*: indica la somma della terna (nei primi quindici turni non è indicato il valore perché le tre scelte sono riferite alla fase di apprendimento con gli agenti artificiali).
- *Scelta1\2\3*: indica la scelta del giocatore1\2\3.
- *Appr1\2\3*: indica il tipo di apprendimento effettuato dal giocatore 1\2\3 (afad=altruista forte e debole) (edef=egoista debole e forte).

GIOCO=arun1

OBS	*TURNO*	*SOMMA*	*SCELTA1*	*SCELTA2*	*SCELTA3*	*APPR1*	*APPR2*	*APPR3*
1	1		4	0	0	afad	edef	afad
2	2		3	4	1	afad	edef	afad
3	3		4	4	0	afad	edef	afad
4	4		3	4	0	afad	edef	afad
5	5		3	4	0	afad	edef	afad
6	6		3	4	1	afad	edef	afad
7	7		3	4	3	afad	edef	afad
8	8		3	4	0	afad	edef	afad
9	9		3	4	3	afad	edef	afad
10	10		3	4	3	afad	edef	afad
11	11		3	4	3	afad	edef	afad
12	12		3	4	3	afad	edef	afad
13	13		3	4	0	afad	edef	afad
14	14		3	4	3	afad	edef	afad
15	15		3	4	0	afad	edef	afad
16	16	6	3	3	0	afad	edef	afad
17	17	7	3	3	1	afad	edef	afad
18	18	6	3	3	0	afad	edef	afad
19	19	16	10	3	3	afad	edef	afad
20	20	6	3	3	0	afad	edef	afad
21	21	9	3	3	3	afad	edef	afad
22	22	10	3	3	4	afad	edef	afad
23	23	6	3	3	0	afad	edef	afad
24	24	10	3	3	4	afad	edef	afad
25	25	10	3	3	4	afad	edef	afad
26	26	10	3	3	4	afad	edef	afad
27	27	6	3	3	0	afad	edef	afad
28	28	10	3	3	4	afad	edef	afad
29	29	10	3	3	4	afad	edef	afad
30	30	6	3	3	0	afad	edef	afad

GIOCO=arun2

OBS	*TURNO*	*SOMMA*	*SCELTA1*	*SCELTA2*	*SCELTA3*	*APPR1*	*APPR2*	*APPR3*
31	1		4	4	4	afad	edef	afad
32	2		4	3	3	afad	edef	afad
33	3		4	3	4	afad	edef	afad
34	4		4	10	10	afad	edef	afad
35	5		3	3	3	afad	edef	afad
36	6		3	10	4	afad	edef	afad
37	7		3	0	4	afad	edef	afad
38	8		3	10	4	afad	edef	afad
39	9		3	0	4	afad	edef	afad
40	10		3	0	4	afad	edef	afad
41	11		3	10	4	afad	edef	afad
42	12		3	0	4	afad	edef	afad
43	13		3	10	4	afad	edef	afad
44	14		3	10	4	afad	edef	afad
45	15		3	0	4	afad	edef	afad
46	16	9	3	3	3	afad	edef	afad
47	17	10	3	3	4	afad	edef	afad
48	18	10	3	3	4	afad	edef	afad
49	19	10	3	3	4	afad	edef	afad
50	20	6	3	3	0	afad	edef	afad
51	21	4	3	0	1	afad	edef	afad
52	22	7	0	4	3	afad	edef	afad
53	23	6	3	3	0	afad	edef	afad
54	24	4	0	4	0	afad	edef	afad
55	25	3	0	3	0	afad	edef	afad
56	26	0	0	0	0	afad	edef	afad
57	27	1	0	0	1	afad	edef	afad
58	28	11	0	10	1	afad	edef	afad
59	29	3	0	0	3	afad	edef	afad
60	30	4	0	4	0	afad	edef	afad

GIOCO=arun3

OBS	*TURNO*	*SOMMA*	*SCELTA1*	*SCELTA2*	*SCELTA3*	*APPR1*	*APPR2*	*APPR3*
61	1		4	3	10	edef	afad	edef
62	2		4	4	4	edef	afad	edef
63	3		4	3	10	edef	afad	edef
64	4		4	3	10	edef	afad	edef
65	5		4	3	3	edef	afad	edef
66	6		4	3	10	edef	afad	edef
67	7		4	3	10	edef	afad	edef
68	8		4	3	10	edef	afad	edef
69	9		4	3	0	edef	afad	edef
70	10		4	3	10	edef	afad	edef
71	11		4	3	10	edef	afad	edef
72	12		4	3	10	edef	afad	edef
73	13		4	3	10	edef	afad	edef
74	14		4	3	1	edef	afad	edef
75	15		4	3	1	edef	afad	edef
76	16	16	3	3	10	edef	afad	edef
77	17	10	0	0	10	edef	afad	edef
78	18	10	0	0	10	edef	afad	edef
79	19	10	0	0	10	edef	afad	edef
80	20	10	0	0	10	edef	afad	edef
81	21	10	0	0	10	edef	afad	edef
82	22	10	0	0	10	edef	afad	edef
83	23	10	0	0	10	edef	afad	edef
84	24	10	0	0	10	edef	afad	edef
85	25	0	0	0	0	edef	afad	edef
86	26	21	10	10	1	edef	afad	edef
87	27	1	0	0	1	edef	afad	edef
88	28	3	0	0	3	edef	afad	edef
89	29	24	4	10	10	edef	afad	edef
90	30	5	4	0	1	edef	afad	edef

GIOCO=arun4

OBS	*TURNO*	*SOMMA*	*SCELTA1*	*SCELTA2*	*SCELTA3*	*APPR1*	*APPR2*	*APPR3*
91	1		3	3	10	edef	afad	edef
92	2		10	4	3	edef	afad	edef
93	3		10	3	10	edef	afad	edef
94	4		10	3	10	edef	afad	edef
95	5		4	3	10	edef	afad	edef
96	6		10	3	3	edef	afad	edef
97	7		10	3	0	edef	afad	edef
98	8		10	3	0	edef	afad	edef
99	9		10	3	0	edef	afad	edef
100	10		10	3	0	edef	afad	edef
101	11		10	3	0	edef	afad	edef
102	12		10	3	0	edef	afad	edef
103	13		10	3	0	edef	afad	edef
104	14		4	3	0	edef	afad	edef
105	15		10	3	0	edef	afad	edef
106	16	8	4	4	0	edef	afad	edef
107	17	9	3	3	3	edef	afad	edef
108	18	18	4	4	10	edef	afad	edef
109	19	9	3	3	3	edef	afad	edef
110	20	11	4	3	4	edef	afad	edef
111	21	9	3	3	3	edef	afad	edef
112	22	11	4	3	4	edef	afad	edef
113	23	11	3	4	4	edef	afad	edef
114	24	9	1	4	4	edef	afad	edef
115	25	17	3	4	10	edef	afad	edef
116	26	9	1	4	4	edef	afad	edef
117	27	11	4	4	3	edef	afad	edef
118	28	8	0	4	4	edef	afad	edef
119	29	17	10	4	3	edef	afad	edef
120	30	18	4	4	10	edef	afad	edef

GIOCO=brun1

OBS	*TURNO*	*SOMMA*	*SCELTA1*	*SCELTA2*	*SCELTA3*	*APPR1*	*APPR2*	*APPR3*
121	1		10	3	4	afad	edef	afad
122	2		3	3	3	afad	edef	afad
123	3		3	10	3	afad	edef	afad
124	4		3	0	3	afad	edef	afad
125	5		4	4	3	afad	edef	afad
126	6		4	0	3	afad	edef	afad
127	7		10	4	3	afad	edef	afad
128	8		1	3	3	afad	edef	afad
129	9		3	4	3	afad	edef	afad
130	10		4	4	3	afad	edef	afad
131	11		4	4	3	afad	edef	afad
132	12		10	4	3	afad	edef	afad
133	13		3	4	3	afad	edef	afad
134	14		4	4	3	afad	edef	afad
135	15		3	4	3	afad	edef	afad
136	16	4	1	0	3	afad	edef	afad
137	17	13	3	0	10	afad	edef	afad
138	18	3	3	0	0	afad	edef	afad
139	19	6	3	0	3	afad	edef	afad
140	20	8	4	4	0	afad	edef	afad
141	21	11	4	4	3	afad	edef	afad
142	22	10	4	3	3	afad	edef	afad
143	23	10	4	3	3	afad	edef	afad
144	24	10	4	3	3	afad	edef	afad
145	25	10	4	3	3	afad	edef	afad
146	26	7	4	3	0	afad	edef	afad
147	27	16	10	3	3	afad	edef	afad
148	28	10	4	3	3	afad	edef	afad
149	29	10	4	3	3	afad	edef	afad
150	30	10	4	3	3	afad	edef	afad

GIOCO=brun2

OBS	*TURNO*	*SOMMA*	*SCELTA1*	*SCELTA2*	*SCELTA3*	*APPR1*	*APPR2*	*APPR3*
151	1		3	4	4	afad	edef	afad
152	2		3	3	3	afad	edef	afad
153	3		3	3	3	afad	edef	afad
154	4		3	4	4	afad	edef	afad
155	5		3	3	3	afad	edef	afad
156	6		3	3	3	afad	edef	afad
157	7		3	4	3	afad	edef	afad
158	8		3	3	1	afad	edef	afad
159	9		3	3	3	afad	edef	afad
160	10		3	4	4	afad	edef	afad
161	11		3	10	3	afad	edef	afad
162	12		3	4	3	afad	edef	afad
163	13		3	4	3	afad	edef	afad
164	14		3	3	3	afad	edef	afad
165	15		3	3	3	afad	edef	afad
166	16	10	3	4	3	afad	edef	afad
167	17	9	3	3	3	afad	edef	afad
168	18	11	3	4	4	afad	edef	afad
169	19	9	3	3	3	afad	edef	afad
170	20	9	3	3	3	afad	edef	afad
171	21	11	3	4	4	afad	edef	afad
172	22	11	3	4	4	afad	edef	afad
173	23	9	3	3	3	afad	edef	afad
174	24	10	3	3	4	afad	edef	afad
175	25	10	3	3	4	afad	edef	afad
176	26	10	3	3	4	afad	edef	afad
177	27	10	3	3	4	afad	edef	afad
178	28	10	3	3	4	afad	edef	afad
179	29	10	3	3	4	afad	edef	afad
180	30	10	3	3	4	afad	edef	afad

GIOCO=brun3

OBS	*TURNO*	*SOMMA*	*SCELTA1*	*SCELTA2*	*SCELTA3*	*APPR1*	*APPR2*	*APPR3*
181	1		4	3	3	edef	afad	edef
182	2		3	0	0	edef	afad	edef
183	3		3	1	0	edef	afad	edef
184	4		4	10	3	edef	afad	edef
185	5		10	3	0	edef	afad	edef
186	6		10	0	3	edef	afad	edef
187	7		10	0	3	edef	afad	edef
188	8		10	0	3	edef	afad	edef
189	9		4	0	0	edef	afad	edef
190	10		10	0	3	edef	afad	edef
191	11		4	0	3	edef	afad	edef
192	12		10	0	3	edef	afad	edef
193	13		4	0	3	edef	afad	edef
194	14		0	0	3	edef	afad	edef
195	15		0	0	3	edef	afad	edef
196	16	10	10	0	0	edef	afad	edef
197	17	4	4	0	0	edef	afad	edef
198	18	10	10	0	0	edef	afad	edef
199	19	4	4	0	0	edef	afad	edef
200	20	10	10	0	0	edef	afad	edef
201	21	7	4	3	0	edef	afad	edef
202	22	16	10	3	3	edef	afad	edef
203	23	10	4	3	3	edef	afad	edef
204	24	6	0	3	3	edef	afad	edef
205	25	3	0	3	0	edef	afad	edef
206	26	10	10	0	0	edef	afad	edef
207	27	4	4	0	0	edef	afad	edef
208	28	6	0	3	3	edef	afad	edef
209	29	0	0	0	0	edef	afad	edef
210	30	7	4	0	3	edef	afad	edef

GIOCO=brun4

OBS	*TURNO*	*SOMMA*	*SCELTA1*	*SCELTA2*	*SCELTA3*	*APPR1*	*APPR2*	*APPR3*
211	1		4	3	4	edef	afad	edef
212	2		4	1	3	edef	afad	edef
213	3		4	3	4	edef	afad	edef
214	4		4	4	3	edef	afad	edef
215	5		4	1	3	edef	afad	edef
216	6		4	1	3	edef	afad	edef
217	7		4	3	4	edef	afad	edef
218	8		4	1	3	edef	afad	edef
219	9		4	1	3	edef	afad	edef
220	10		4	3	4	edef	afad	edef
221	11		4	0	3	edef	afad	edef
222	12		4	3	3	edef	afad	edef
223	13		4	3	3	edef	afad	edef
224	14		4	3	3	edef	afad	edef
225	15		4	3	3	edef	afad	edef
226	16	9	3	3	3	edef	afad	edef
227	17	8	3	1	4	edef	afad	edef
228	18	11	3	4	4	edef	afad	edef
229	19	9	3	3	3	edef	afad	edef
230	20	7	3	1	3	edef	afad	edef
231	21	8	3	1	4	edef	afad	edef
232	22	16	3	3	10	edef	afad	edef
233	23	11	3	4	4	edef	afad	edef
234	24	10	3	3	4	edef	afad	edef
235	25	10	3	3	4	edef	afad	edef
236	26	9	3	3	3	edef	afad	edef
237	27	10	3	4	3	edef	afad	edef
238	28	11	3	4	4	edef	afad	edef
239	29	9	3	3	3	edef	afad	edef
240	30	9	3	3	3	edef	afad	edef

GIOCO=brun5

OBS	*TURNO*	*SOMMA*	*SCELTA1*	*SCELTA2*	*SCELTA3*	*APPR1*	*APPR2*	*APPR3*
241	1		1	3	4	edef	afad	edef
242	2		3	4	3	edef	afad	edef
243	3		1	10	3	edef	afad	edef
244	4		1	1	3	edef	afad	edef
245	5		0	4	3	edef	afad	edef
246	6		0	3	3	edef	afad	edef
247	7		0	1	3	edef	afad	edef
248	8		10	3	3	edef	afad	edef
249	9		3	0	3	edef	afad	edef
250	10		3	3	3	edef	afad	edef
251	11		0	0	3	edef	afad	edef
252	12		0	3	3	edef	afad	edef
253	13		0	0	3	edef	afad	edef
254	14		0	3	3	edef	afad	edef
255	15		0	0	3	edef	afad	edef
256	16	13	0	10	3	edef	afad	edef
257	17	13	0	10	3	edef	afad	edef
258	18	10	0	10	0	edef	afad	edef
259	19	0	0	0	0	edef	afad	edef
260	20	13	0	10	3	edef	afad	edef
261	21	5	0	4	1	edef	afad	edef
262	22	13	0	10	3	edef	afad	edef
263	23	2	0	1	1	edef	afad	edef
264	24	0	0	0	0	edef	afad	edef
265	25	13	0	10	3	edef	afad	edef
266	26	3	0	0	3	edef	afad	edef
267	27	7	0	4	3	edef	afad	edef
268	28	7	0	3	4	edef	afad	edef
269	29	13	0	3	10	edef	afad	edef
270	30	4	0	0	4	edef	afad	edef

6.2.7 Analisi dei dati

I dati sono poi stati analizzati. I risultati dell'analisi sono stati raccolti nelle seguenti tabelle: riguardano le medie e le standard deviation dei valori scelti, dei punti ottenuti per turno o totali (punticum) e della somma ottenuta dalle terne. Una delle tabelle è dedicata ai dati degli agenti umani che hanno effettuato l'apprendimento con agenti altruisti (*afad*), l'altra con agenti

egoisti (*edef*). Ogni tabella è composta da una parte riguardane la fase di apprendimento ed una di gioco tra umani.

***APPR=afad* PARTE=1**

Variable	*N*	*Mean*	*Std Dev*	*Minimum*	*Maximum*
SCELTA	180	2,933	1,646	0	10
PUNTI	180	30,122	10,978	-3	37
PUNTICUM	180	233,922	144,568	0	555
SOMMA	180	10,900	2,561	5	24

APPR=afad PARTE=2

Variable	*N*	*Mean*	*Std Dev*	*Minimum*	*Maximum*
SCELTA	180	2,611	1,962	0	10
PUNTI	180	16,944	18,108	-4	40
PUNTICUM	180	124,728	99,036	-9	387
SOMMA	180	8,806	3,719	0	24

APPR=edef PARTE=1

Variable	*N*	*Mean*	*Std Dev*	*Minimum*	*Maximum*
SCELTA	180	4,533	3,200	0	10
PUNTI	180	20,967	16,000	-4	37
PUNTICUM	180	154,894	137,819	-6	540
SOMMA	180	10,361	3,379	3	24

APPR=edef PARTE=2

Variable	*N*	*Mean*	*Std Dev*	*Minimum*	*Maximum*
SCELTA	180	3,400	2,826	0	10
PUNTI	180	15,711	17,532	-4	40
PUNTICUM	180	125,372	98,370	-9	393
SOMMA	180	9,228	4,189	0	24

Innanzitutto notiamo che i punti cumulati sono sempre superiori nella fase di apprendimento rispetto alla fase di gioco tra umani. Questo aspetto

era stato previsto, infatti sembrava plausibile che il coordinamento fosse più semplice con gli agenti virtuali che con quelli umani. Tutti i partecipanti ai turni dall'1 al 4 non erano a conoscenza della fase di apprendimento con agenti virtuali, ed hanno trovato più semplice il gioco con la macchina rispetto a quello con gli umani. Infatti, al termine dell'esperimento molti dei giocatori ritenevano di aver giocato la prima fase con compagni di squadra più intelligenti e più disposti a cooperare; in realtà le strategie di gioco erano solo più semplici da intuire e da assecondare.

I dati delle tabelle indicano inoltre che ci sono differenze nel comportamento dei giocatori umani durante l'apprendimento con agenti virtuali altruisti od egoisti.

La fase di apprendimento dell'esperimento è stata introdotta per cercare di influenzare le strategie dei diversi giocatori; nelle nostre aspettative chi giocava con agenti virtuali altruisti avrebbe dovuto adottare una strategia egoista e viceversa per chi giocava con egoisti. Dai dati infatti notiamo che la media delle scelte effettuate nella prima fase è decisamente più elevata (4,5) per chi ha giocato con gli egoisti rispetto a quella (2.9) di chi ha giocato con gli altruisti. I punti totali rispecchiano il comportamento tenuto, sono infatti più elevati quelli ottenuti con una strategia egoista.

Nella fase di gioco tra umani i valori si avvicinano molto riducendo la forbice che si era creata nella fase di apprendimento.

Rimane invariata però la tendenza a giocare numeri più bassi se si è prima cooperato con agenti virtuali altruisti, anche se non sembra una strategia decisamente vincente dai punti totali ottenuti (125 contro 124).

Le tabelle seguenti invece riportano la frequenza e la percentuale dei singoli valori delle somme, delle scelte e dei punti ottenuti. Anch'esse sono divise secondo il tipo di apprendimento e la parte (1^a o 2^a) dell'esperimento

APPR=afad parte=1

SOMMA	Frequency	Percent	Cumulative Frequency	Cumulative Percent
5	1	0,6	1	0,6
7	11	6,1	12	6,7
8	4	2,2	16	8,9
10	103	57,2	119	66,1
11	30	16,7	149	82,8
14	12	6,7	161	89,4
15	5	2,8	166	92,2
17	13	7,2	179	99,4
24	1	0,6	180	100

SCELTA	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	19	10,6	19	10,6
1	10	5,6	29	16,1
3	116	64,4	145	80,6
4	30	16,7	175	97,2
10	5	2,8	180	100

PUNTI	Frequency	Percent	Cumulative Frequency	Cumulative Percent
-3	4	2,2	4	2,2
-1	5	2,8	9	5
0	7	3,9	16	8,9
20	5	2,8	21	11,7
26	30	16,7	51	28,3
27	9	5	60	33,3
29	5	2,8	65	36,1
30	12	6,7	77	42,8
37	103	57,2	180	100

APPR=afad parte=2

SOMMA	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	4	2,2	4	2,2
1	3	1,7	7	3,9
3	8	4,4	15	8,3
4	11	6,1	26	14,4
5	1	0,6	27	15
6	20	11,1	47	26,1
7	9	5	56	31,1
8	6	3,3	62	34,4
9	22	12,2	84	46,7
10	64	35,6	148	82,2
11	17	9,4	165	91,7
13	2	1,1	167	92,8
16	7	3,9	174	96,7
17	2	1,1	176	97,8
18	2	1,1	178	98,9
21	1	0,6	179	99,4
24	1	0,6	180	100

SCELTA	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	44	24,4	44	24,4
1	8	4,4	52	28,9
3	80	44,4	132	73,3
4	43	23,9	175	97,2
10	5	2,8	180	100

PUNTI	Frequency	Percent	Cumulative Frequency	Cumulative Percent
-4	6	3,3	6	3,3
-3	40	22,2	46	25,6
-1	7	3,9	53	29,4
0	31	17,2	84	46,7
20	5	2,8	89	49,4
26	13	7,2	102	56,7
27	12	6,7	114	63,3
29	1	0,6	115	63,9
30	1	0,6	116	64,4
36	24	13,3	140	77,8
37	28	15,6	168	93,3
40	12	6,7	180	100

APPR=edef parte=1

SOMMA	Frequency	Percent	Cumulative Frequency	Cumulative Percent
3	7	3,9	7	3,9
4	1	0,6	8	4,4
5	1	0,6	9	5
6	11	6,1	20	11,1
7	21	11,7	41	22,8
8	1	0,6	42	23,3
9	9	5	51	28,3
10	72	40	123	68,3
11	4	2,2	127	70,6
13	20	11,1	147	81,7
14	17	9,4	164	91,1
15	1	0,6	165	91,7
16	6	3,3	171	95
17	8	4,4	179	99,4
24	1	0,6	180	100

SCELTA	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	24	13,3	24	13,3
1	2	1,1	26	14,4
3	42	23,3	68	37,8
4	72	40	140	77,8
10	40	22,2	180	100

PUNTI	Frequency	Percent	Cumulative Frequency	Cumulative Percent
-4	15	8,3	15	8,3
-3	20	11,1	35	19,4
-1	1	0,6	36	20
0	15	8,3	51	28,3
20	40	22,2	91	50,6
26	4	2,2	95	52,8
27	3	1,7	98	54,4
29	1	0,6	99	55
30	9	5	108	60
36	53	29,4	161	89,4
37	19	10,6	180	100

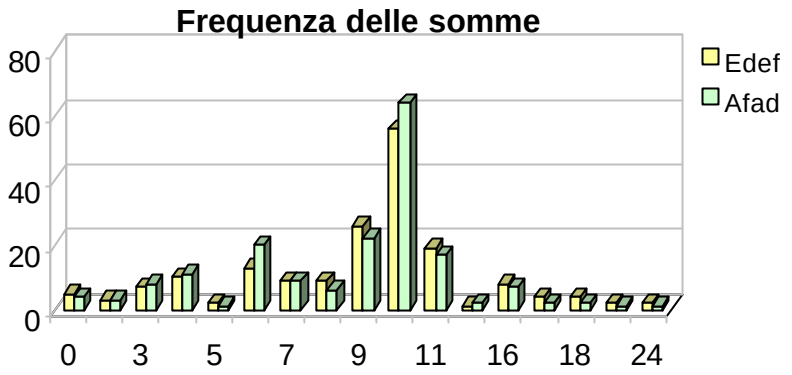
APPR=edef parte=2

SOMMA	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	5	2,8	5	2,8
1	3	1,7	8	4,4
3	7	3,9	15	8,3
4	10	5,6	25	13,9
5	2	1,1	27	15
6	13	7,2	40	22,2
7	9	5	49	27,2
8	9	5	58	32,2
9	26	14,4	84	46,7
10	56	31,1	140	77,8
11	19	10,6	159	88,3
13	1	0,6	160	88,9
16	8	4,4	168	93,3
17	4	2,2	172	95,6
18	4	2,2	176	97,8
21	2	1,1	178	98,9
24	2	1,1	180	100

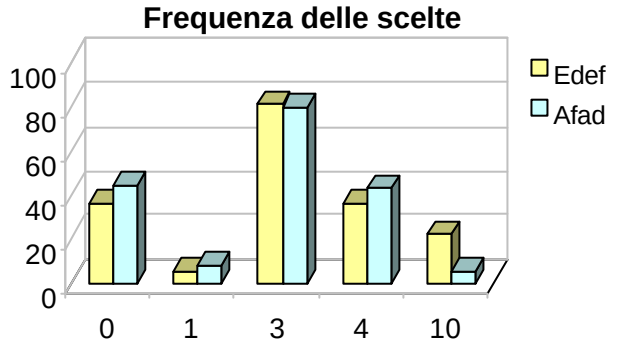
SCELTA	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	36	20	36	20
1	5	2,8	41	22,8
3	81	45	122	67,8
4	36	20	158	87,8
10	22	12,2	180	100

PUNTI	Frequency	Percent	Cumulative Frequency	Cumulative Percent
-4	16	8,9	16	8,9
-3	41	22,8	57	31,7
-1	4	2,2	61	33,9
0	23	12,8	84	46,7
20	10	5,6	94	52,2
26	16	8,9	110	61,1
27	12	6,7	122	67,8
29	1	0,6	123	68,3
30	13	7,2	136	75,6
36	4	2,2	140	77,8
37	28	15,6	168	93,3
40	12	6,7	180	100

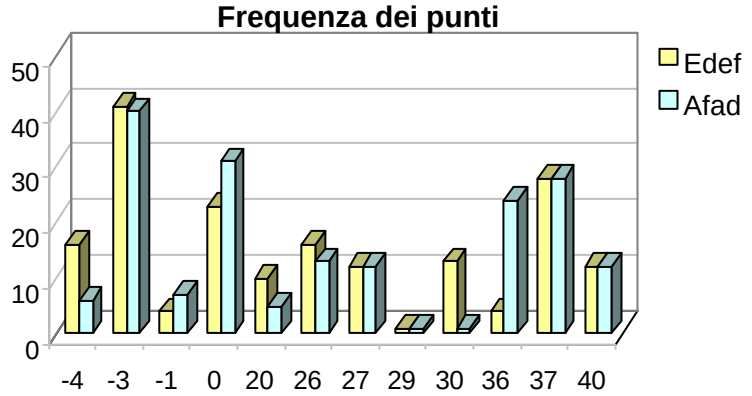
L'analisi delle frequenze delle somme appare smentire un'eventuale differenza tra i due gruppi di giocatori umani. Come risulta chiaro dal grafico seguente, la distribuzione delle somme è pressoché identica.



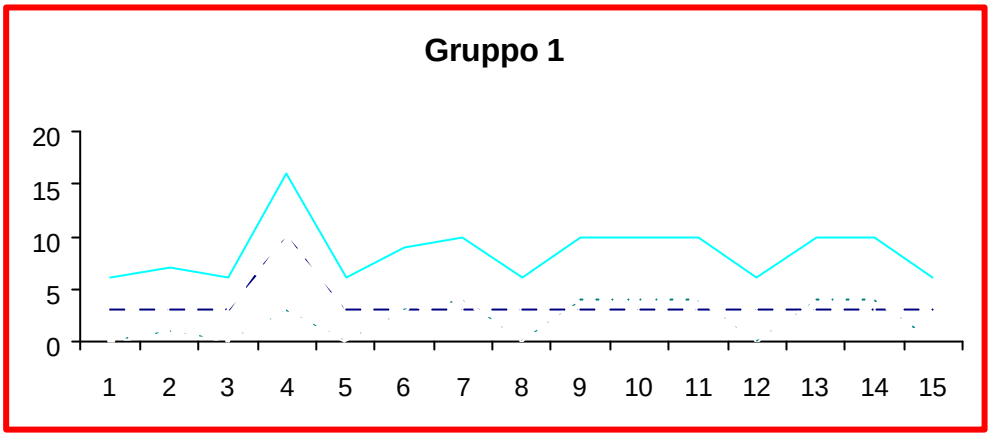
La frequenza delle scelte presenta la stessa situazione, si può notare però che il numero dei 10 è sensibilmente più elevato. Ciò dipende, senza dubbio, dall'apprendimento fatto (con agenti egoisti), ma è soprattutto espressione della strategia molto altruista di alcuni giocatori.

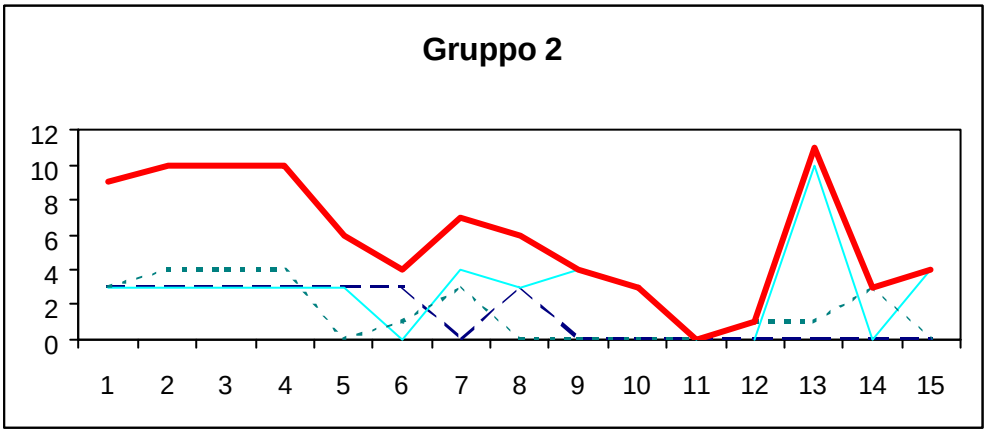


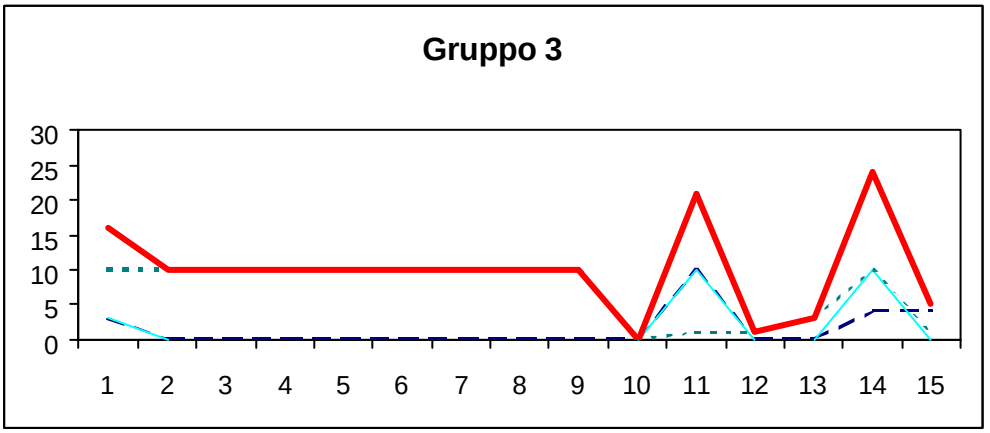
Infine dalla distribuzione dei punti per turno si può notare ancora una volta che le differenze sono veramente minime.

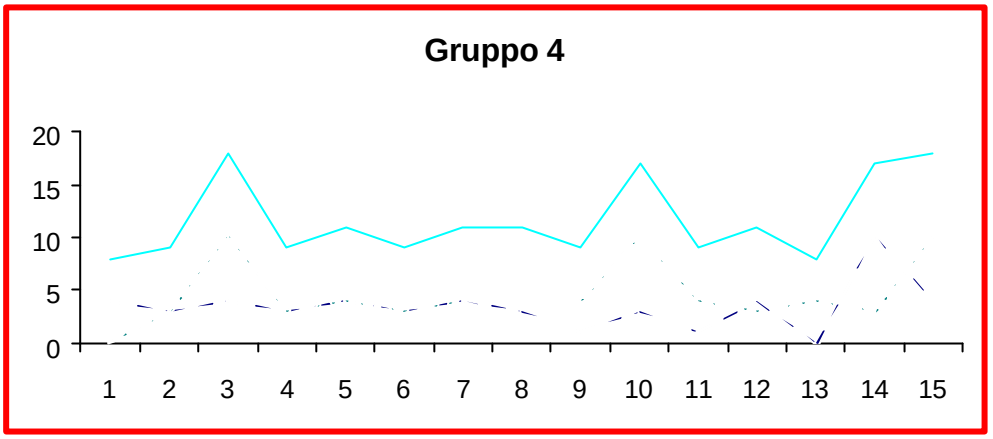


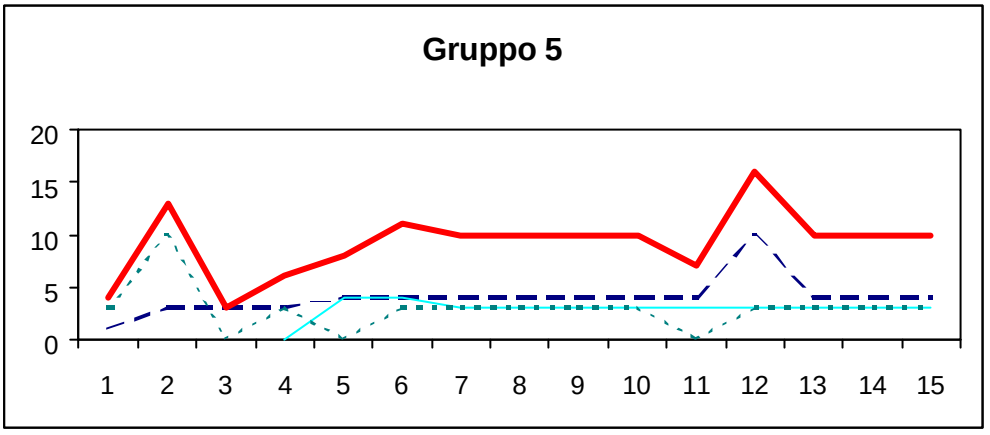
Riporto di seguito i grafici più significativi, riguardanti i singoli andamenti delle terne, per permettere una visione d'insieme delle strategie effettivamente riscontrate.

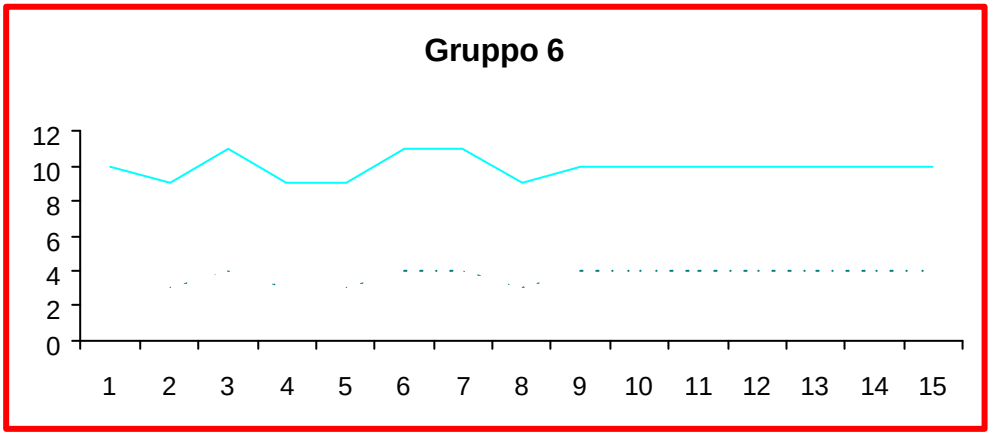


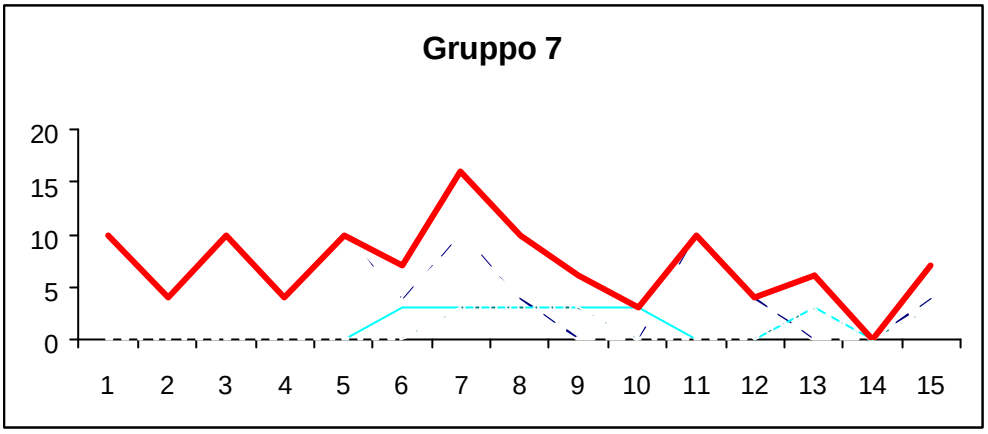


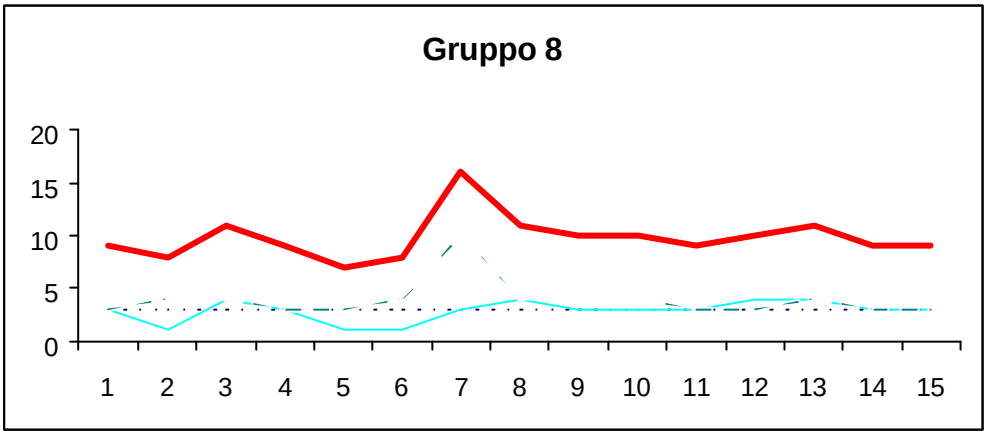


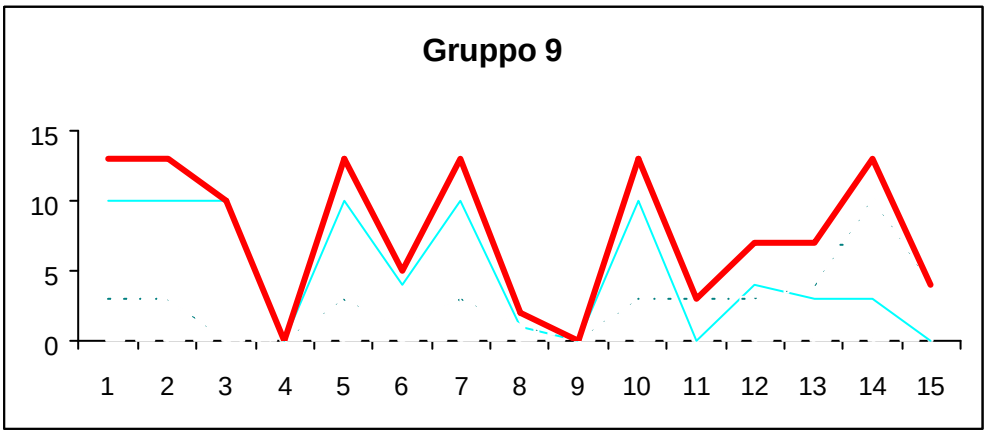

Gruppo 4
20
15
10
5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15


Gruppo 5
20
15
10
5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15


Gruppo 6
12
10
8
6
4
2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15


Gruppo 7
20
15
10
5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15


Gruppo 8
20
15
10
5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15


Gruppo 9
15
10
5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

CONCLUSIONI

All'interno di questo lavoro sono stati delineati i tratti essenziali dell'economia sperimentale e, attraverso l'analisi dei suoi pregi e dei suoi difetti, le potenzialità a disposizione degli sperimentatori.

Sono state illustrate le radici della sperimentazione in economia e i filoni di ricerca che si sono sviluppati nel tempo; abbiamo cercato di dimostrare l'utilità di studiare in modo accurato il comportamento umano per meglio determinare la figura dell'agente rappresentativo.

In particolare abbiamo ricercato eventuali comportamenti cooperativi ed altruistici in un ambiente economico attraverso un esperimento svolto presso la Facoltà di Economia. L'idea alla base della nostra sperimentazione nasce per verificare se il comportamento umano è influenzabile dall'apprendimento effettuato con agenti artificiali. Tale fase di apprendimento con agenti virtuali egoisti (altruisti) guidati da un algoritmo poteva:

- non avere nessun effetto sul comportamento umano nella fase seguente;
- portare gli agenti umani ad un comportamento altruistico (egoistico), nato per assecondare il comportamento della squadra;
- portare ad un atteggiamento egoistico (altruistico) per reazione al comportamento della squadra.

Il programma utilizzato ci ha permesso di raggiungere gli obbiettivi che ci eravamo preposti e di verificare le ipotesi, adattandosi perfettamente alle nostre necessità.

SWIEE consente di utilizzare Swarm per effettuare esperimenti in modo da far interagire due modi diversi di creare, analizzare e verificare teorie economiche:

- "le simulazioni", dove si cerca di riprodurre parte della realtà che ci circonda per meglio comprenderne il funzionamento. Si crea il mondo, se ne osserva l'evoluzione artificiale e si cerca di trarne conclusioni applicabili alla realtà;
- "gli esperimenti", dove un campione di agenti reali viene coinvolto in situazioni predeterminate al fine di studiarne le azioni e reazioni, cercando di comprenderne i ragionamenti e le strategie.

SWIEE unisce questi due aspetti facendo interagire un mondo simulato, dove il trascorre del tempo viene gestito internamente da Swarm, ed uno reale dove il tempo scorre e viene percepito in maniera diversa da ciascuno dei partecipanti.

I risultati dell'esperimento ci hanno permesso di constatare che, dopo aver effettuato la fase di apprendimento con agenti artificiali, gli agenti umani hanno dimostrato una marcata tendenza all'adattamento al gioco della squadra, mentre nella fase in cui erano presenti solo giocatori umani, questi hanno mostrato reazioni meno "lineari".

Questo può dipendere dal fatto che le scelte "suggerite" dagli algoritmi, coerenti con una strategia semplice e precisa, inducevano il giocatore umano ad assecondare il comportamento degli agenti artificiali, mentre in una squadra di soli agenti umani era più difficile individuare le strategie dei compagni.

Tra le situazioni estreme, sono stati riscontrati atteggiamenti puramente egoistici che hanno creato notevoli difficoltà nella ricerca di un equilibrio e che hanno impedito alla squadra di ottenere un punteggio elevato.

L'analisi dei dati ha evidenziato che si sono create strategie di comportamento precise: con atteggiamenti altruisti, nei confronti di giocatori che dimostravano di seguire strategie coerenti o cooperative, ed egoistici, verso chi adottava strategie di difficile comprensione od eccessivamente egoistiche.

La tendenza generale, riscontrabile nei grafici, mostra infine che la fase di apprendimento influenza in maniera significativa il successivo comportamento del giocatore umano: la maggior parte dei partecipanti all'esperimento infatti ha dimostrato di accettare il comportamento imposto dagli agenti artificiali, proiettando anche nella seconda fase la strategia "suggerita" dal gioco con gli artificiali.

Il lavoro sperimentale svolto a Torino ha portato buoni frutti, sia in termini di partecipazione ed entusiasmo, sia in termini di coordinamento interno e di utilizzo degli strumenti a disposizione. L'esperimento da noi svolto vuole essere soprattutto un punto di partenza, uno stimolo per approfondire lo studio di quest'aspetto dell'economia, per ampliare

l’utilizzo di queste metodologie di analisi, e per dimostrare l’utilità, la versatilità e la potenza degli strumenti utilizzati

APPENDICE

Listato dei codici

StarSum10.java

```
// Game Theory - Model 1 Copyright © 2000 Riccardo Boero
// This library is distributed without any warranty; without
even the
// implied warranty of merchantability or fitness for a
particular purpose.
// See file LICENSE for details and terms of copying.

import swarm.Globals;

public class StartSum10 {
    // Main function
    public static void main (String[] args) {
        Globals.env.initSwarm ("sum10", "2.1.1", "bug-
swarm@santafe.edu", args);

      Sum10ObserverSwarm topLevelSwarm =
          new Sum10ObserverSwarm (Globals.env.globalZone);
      Globals.env.setWindowGeometryRecordName
(topLevelSwarm, "topLevelSwarm");

      topLevelSwarm.buildObjects ();
      topLevelSwarm.buildActions ();
      topLevelSwarm.activateIn (null);
      topLevelSwarm.go ();
      topLevelSwarm.drop ();
    }
}
```

Player.java

```
import swarm.Globals;
import java.io.*;

// The virtual representation of the player

public class Player {

    // paramethers
    public int points, choice, choice2, payoffArray[], id,
cumPoints, accInstructions;
    public int id1, id2, sum3, id3, id4;
    public Player playerArray[];
    public Timer timer;
    public PrintWriter buff;
    public String data;
    public RuleMaster ruleMaster;
    public int numPlayers, changed;

    // constructor
    public Player (int b, int a[], int c, Player[] pl, Timer
t, RuleMaster r) {
      choice = 0;
      points = 0;
      cumPoints = 0;
      accInstructions = 0;
```

```
      numPlayers = b;
      payoffArray = a;
      id = c;
      playerArray = pl;
      timer = t;
      ruleMaster = r;
      buff = timer.getBuffer();
      changed = 0;
      int j;
      //choice2 = 1;
      if (id == 0 || id == 3 || id == 6 || id == 9 || id ==
12 || id == 15 || id == 18 || id == 21 || id == 24 || id ==
27 || id == 30 || id == 33 || id == 36 || id == 39 || id ==
42 || id == 45 || id == 48 || id == 51 || id == 54 || id ==
57 || id == 60 || id == 63 || id == 66 || id == 69 || id ==
72 || id == 75 || id == 78 || id == 81 || id == 84 || id ==
87 || id == 90 || id == 93 || id == 96 || id == 99 || id ==
102 || id == 105 || id == 108 || id == 111 || id == 114 ||
id == 117 || id == 120 || id == 123) {
          id1 = id + 1;
          id2 = id + 2;
      }
      if (id == 1 || id == 4 || id == 7 || id == 10 || id ==
13 || id == 16 || id == 19 || id == 22 || id == 25 || id ==
28 || id == 31 || id == 34 || id == 37 || id == 40 || id ==
43 || id == 46 || id == 49 || id == 52 || id == 55 || id ==
58 || id == 61 || id == 64 || id == 67 || id == 70 || id ==
73 || id == 76 || id == 79 || id == 82 || id == 85 || id ==
88 || id == 91 || id == 94 || id == 97 || id == 100 || id ==
103 || id == 106 || id == 109 || id == 112 || id == 115 ||
id == 118 || id == 121 || id == 124) {
          id1 = id - 1;
          id2 = id + 1;
      }
      if (id == 2 || id == 5 || id == 8 || id == 11 || id ==
14 || id == 17 || id == 20 || id == 23 || id == 26 || id ==
29 || id == 32 || id == 35 || id == 38 || id == 41 || id ==
44 || id == 47 || id == 50 || id == 53 || id == 56 || id ==
59 || id == 62 || id == 65 || id == 68 || id == 71 || id ==
74 || id == 77 || id == 80 || id == 83 || id == 86 || id ==
89 || id == 92 || id == 95 || id == 98 || id == 101 || id ==
104 || id == 107 || id == 110 || id == 113 || id == 116 ||
id == 119 || id == 122 || id == 125) {
          id1 = id - 2;
          id2 = id - 1;
      }
      if ((id >= 0 && id <=2) || (id >= 9 && id <=11) || (id
>= 18 && id <=20) || (id >= 27 && id <=29) || (id >= 36 &&
id <=38) || (id >= 45 && id <=47) || (id >= 54 && id <=56)
|| (id >= 63 && id <=65) || (id >= 72 && id <=74) || (id >=
81 && id <=83) || (id >= 90 && id <=92) || (id >= 99 && id
<=101) || (id >= 108 && id <=110) || (id >= 117 && id
<=119)) {
          id3 = id + 3;
          id4 = id + 6;
      }
      if ((id >= 3 && id <=5) || (id >= 12 && id <=14) ||
(id >= 21 && id <=23) || (id >= 30 && id <=32) || (id >= 39
&& id <=41) || (id >= 48 && id <=50) || (id >= 57 && id
<=59) || (id >= 66 && id <=68) || (id >= 75 && id <=77) ||
(id >= 84 && id <=86) || (id >= 93 && id <=95) || (id >= 102
&& id <=104) || (id >= 111 && id <=113) || (id >= 120 && id
<=122)) {
```

```
          id3 = id - 3;
          id4 = id + 3;
      }
      if ((id >= 6 && id <=8) || (id >= 15 && id <=17) ||
(id >= 24 && id <=26) || (id >= 33 && id <=35) || (id >= 42
&& id <=44) || (id >= 51 && id <=53) || (id >= 60 && id
<=62) || (id >= 69 && id <=71) || (id >= 78 && id <=80) ||
(id >= 87 && id <=89) || (id >= 96 && id <=98) || (id >= 105
&& id <=107) || (id >= 114 && id <=116) || (id >= 123 && id
<=125)) {
          id3 = id - 6;
          id4 = id - 3;
      }
    }

    // methods
    public int getPoints () {
      return points;
    }

    public int[] getOtherChoice () {
      int i[];
      i = new int[2];
      i[0] = playerArray[id1].getChoice();
      i[1] = playerArray[id2].getChoice();
      return i;
    }

    public Object setChoice (int i) {
      choice = i;
      timer.setBlock(id);
      return this;
    }

    public int getChoice () {
      return choice;
    }

    public int getChoice2 () {
      return choice2;
    }

    public Object setChoice2 () {
      choice2 = choice;
      return this;
    }

    public int getCumPoints () {
      return cumPoints;
    }

    public Object setInstructions () {
      accInstructions++;
      return this;
    }

    public Object nullInstructions () {
      accInstructions = 0;
      return this;
    }

    public int getSum3 () {
      int sum3b, i[];
```

```
      i = getOtherChoice();
      sum3b = choice + i[0] + i[1];
      return sum3b;
    }

    // Schedule's method
    public Object step () {
      if (id == 0 || id == 3 || id == 6 || id == 9 || id ==
12 || id == 15 || id == 18 || id == 21 || id == 24 || id ==
27 || id == 30 || id == 33 || id == 36 || id == 39 || id ==
42 || id == 45 || id == 48 || id == 51 || id == 54 || id ==
57 || id == 60 || id == 63 || id == 66 || id == 69 || id ==
72 || id == 75 || id == 78 || id == 81 || id == 84 || id ==
87 || id == 90 || id == 93 || id == 96 || id == 99 || id ==
102 || id == 105 || id == 108 || id == 111 || id == 114 ||
id == 117 || id == 120 || id == 123) {
      }
      else {
      setChoice(ruleMaster.getChoice(this));
      }
      return this;
    }

    public Object step2 () {
      while (!(timer.askBlock())) {
          continue;
      }


      //Upgrade payoffs
      //int otherChoice[];
      //otherChoice = getOtherChoice ();

      sum3 = getSum3();
      if (sum3 == 10)
          points = 40 - choice;
      if (sum3 > 10)
          points = 30 - choice;
      if (sum3 < 10)
          points = -choice;

      //if (otherChoice == expectation)
      //  payoff ++;
      if (changed == 1) {
          changed = 0;
          cumPoints = 0;
      }
      if (timer.askChange()) {
         changed = 1;
         id1 = id3;
         id2 = id4;
      }
      cumPoints += points;
      timer.setBlock2();
      setChoice2();
      data = (String) (id + " " + choice + " " + points + "
" + cumPoints + " " + sum3);
      //System.out.println(data);
      buff.println(data);
      return this;
    }
}
```

RemoteServer.java

```
import java.rmi.Naming;
import java.rmi.*;
import java.net.*;
import java.io.*;
import java.rmi.server.*;
import java.rmi.registry.LocateRegistry;
// Remote Server's class easily manages in Swarm simulation
// RMI Java technology

public class RemoteServer {
    Player[] playerArray;
    Timer timer;
    private static RMIRegistry  rmi;
    public RemoteServer(Player[] pl, Timer t, int[]
payoffArray) {
      playerArray = pl;
      timer = t;
      try {
          activateRegistry();
          //System.setSecurityManager(new
MySecurityManager());
          RemoteSum10 c = new RemoteSum10Impl(playerArray,
timer, payoffArray);
          // Pay attention to the IP of your host and to the
port:
          // default port of RMIRegistry is 1099
          Naming.rebind("//130.192.140.49:1098/RemoteSum10",
c);
          System.out.println("RemoteServer is registered.");
      } catch (Exception e) {
          System.out.println("Trouble in get RemoteSum10: "
+ e);
      }
    }
    public void activateRegistry() {

      try
          {
            rmi = new RMIRegistry();
          }
      catch ( java.rmi.UnknownHostException uhe )
          {
            System.out.println( "The host computer name you
have specified, does not match your real computer name." );
          }
      catch ( RemoteException re )
          {
            System.out.println( "Error starting service" );
            System.out.println( "" + re );
          }
      catch ( MalformedURLException mURLe )
          {
            System.out.println( "Internal error" + mURLe );
          }
      catch ( NotBoundException nbe )
          {
            System.out.println( "Not Bound" );
            System.out.println( "" + nbe );
          }
```

```
    }
}
```

RemoteSum10.java

```
import java.rmi.*;

// Interface to Server Methods

public interface RemoteSum10 extends Remote {
    public void setChoice(int i, int v) throws
RemoteException;
    public int getCumPoints(int i) throws RemoteException;
    public boolean askRound(int i) throws RemoteException;
    public boolean askBlock2() throws RemoteException;
    public int getPoints(int i) throws RemoteException;
    public int[] getPayoffArray() throws RemoteException;
    public int getChoice(int i) throws RemoteException;
    public boolean askEnd() throws RemoteException;
    public int askNumRound() throws RemoteException;
    public void setBlock3() throws RemoteException;
    public boolean askChange() throws RemoteException;
}
```

RemoteSum10Impl.java

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;

// Implementation of Server's Methods.

public class RemoteSum10Impl extends UnicastRemoteObject
    implements RemoteSum10 {
    public Player[] playerArray;
    public Timer timer;
    public int[] payoffArray;
    public RemoteSum10Impl(Player[] pl, Timer t, int[] p)
throws RemoteException {
      super();
      playerArray = pl;
      timer = t;
      payoffArray = p;
    }
    public void setChoice(int i, int v) throws
RemoteException {
      int id = i, value = v;
      playerArray[id].setChoice(value);
    }
    public int getCumPoints(int i) throws RemoteException  {
      int id = i;
      int value = playerArray[id].getCumPoints();
      return value;
      }
    public boolean askRound(int i) throws RemoteException  {
      int id = i;
      boolean ok = timer.getRound(id);
      return ok;
    }
    public void setBlock3() throws RemoteException  {
```

```
      timer.setBlock3();
    }
    public boolean askBlock2() throws RemoteException  {
      boolean ok = (timer.askBlock2());
      return ok;
      }
    public int getPoints(int i) throws RemoteException  {
      int id = i;
      int value = playerArray[id].getPoints();
      return value;
    }
    public int[] getPayoffArray() throws RemoteException  {
      return payoffArray;
    }

    public int getChoice(int i) throws RemoteException  {
      int value = playerArray[i].getChoice2();
      return value;
    }
    public boolean askEnd() throws RemoteException  {
      boolean ok;
      if (timer.cycles >= (timer.numCycles - 1))
          ok = true;
      else
          ok = false;
      return ok;
    }
    public boolean askChange() throws RemoteException  {
      boolean ok;
      if (timer.askChange())
          ok = true;
      else
          ok = false;
      return ok;
    }
    public int askNumRound() throws RemoteException  {
      int i = timer.cycles;
      return i;
    }
}
```

RmiRegistry.java

```
import java.rmi.Naming;
import java.rmi.*;
import java.net.*;
import java.io.*;
import java.rmi.server.*;
import java.rmi.registry.LocateRegistry;

public class RMIRegistry {
    public RMIRegistry() throws RemoteException,
MalformedURLException, NotBoundException {
      LocateRegistry.createRegistry( 1098 );
      System.out.println( "Registry created on host computer
on port 1098.");
    }
}
```

RuleMaster.java

```
import swarm.Globals;
//import java.io.*;

// Rule Master

public class RuleMaster {

    // paramethers
    public Timer timer;
    public int numPlayers;
    // constructor
    public RuleMaster (Timer t, int r) {
      timer = t;
      numPlayers = r;
      //int i, j, z, x;
    }

    // methods
    public int getChoice (Player a) {
      int b = 0;
      int j;

      /*if (a.id == 0 || a.id == 3 || a.id == 6 || a.id == 9
|| a.id == 12 || a.id == 15
              || a.id == 18 || a.id == 21 || a.id == 24 ||
a.id == 27 || a.id == 30 || a.id == 33
              || a.id == 36 || a.id == 39 || a.id == 42 ||
a.id == 45 || a.id == 48 || a.id == 51) {
        if (timer.cycles <= 1) {
            b = 3;
        }
        else {
               int temp =
Globals.env.uniformIntRand.getIntegerWithMin$withMax (0, 4);
            if (temp == 0)
            b = 0;
            if (temp == 1)
            b = 1;
            if (temp == 2)
            b = 3;
                  if (temp == 3)
                  b = 4;
            if (temp == 4)
            b = 10;
        }
      }*/

// agenti altruisti

      if (a.id == 4 || a.id == 7 || a.id == 13 || a.id == 19
|| a.id == 25 || a.id == 31 || a.id == 37 || a.id == 43 ||
a.id == 49 || a.id == 55 || a.id == 61 || a.id == 67 || a.id
== 73 || a.id == 79 || a.id == 85 || a.id == 91 || a.id ==
97 || a.id == 103 || a.id == 109 || a.id == 115 || a.id ==
121) {
            b = 4;
      }

      if (a.id == 5 || a.id == 8 || a.id == 14 || a.id == 20
|| a.id == 26 || a.id == 32 || a.id == 38 || a.id == 38 ||
a.id == 44 || a.id == 50 || a.id == 56 || a.id == 62 || a.id
== 68 || a.id == 74 || a.id == 80 || a.id == 86 || a.id ==
```

```
92 || a.id == 98 || a.id == 104 || a.id == 110 || a.id ==
116 || a.id == 122) {
        if (timer.cycles == 0) {
          b = 3;
        }
        else {
            if ((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2) < 10) {
               if ((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2 + 1) < 10){
                 if ((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2 + 3) < 10){
                   if ((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2 + 4) < 10){
                     b = 10;
                   }
                   else
                     b = 4;
                 }
                 else
                   b = 3;
               }
               else
                 b = 1;
            }
            else
               b = 0;

        }
      }

// agenti egoisti

      if (a.id == 1 || a.id == 10 || a.id == 16 || a.id ==
22 || a.id == 28 || a.id == 34 || a.id == 40 || a.id == 46
|| a.id == 52 || a.id == 58 || a.id == 64 || a.id == 70 ||
a.id == 76 || a.id == 82 || a.id == 88 || a.id == 94 || a.id
== 100 || a.id == 106 || a.id == 112 || a.id == 118 || a.id
== 124) {
            b = 3;
      }

      if (a.id == 2 || a.id == 11 || a.id == 17 || a.id ==
23 || a.id == 29 || a.id == 35 || a.id == 41 || a.id == 47
|| a.id == 53 || a.id == 59 || a.id == 65 || a.id == 71 ||
a.id == 77 || a.id == 83 || a.id == 89 || a.id == 95 || a.id
== 101 || a.id == 107 || a.id == 113 || a.id == 119 || a.id
== 125) {
        if (timer.cycles == 0) {
          b = 3;
        }
        else {
            if (((a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2) < 7 ||
(a.playerArray[a.id1].choice2 +
a.playerArray[a.id2].choice2) > 10)){
                 b = 0;
            }
            else
               b = 3;
        }
      }
      return b;
```

```
    }
}
```

Sum10Client.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.rmi.*;
import java.rmi.Naming;

public class Sum10Client extends Applet  {
    int value, value2, id, id1, id2, id3, id4,
payoffArray[];
    String identification, valueS;
    boolean updated, activated;
    //Object parentFrame;
    RemoteSum10 server;

    java.awt.Label label1 = new java.awt.Label();
    java.awt.Checkbox radioButton1 = new
java.awt.Checkbox();
    java.awt.CheckboxGroup Group1 = new
java.awt.CheckboxGroup();
    java.awt.Checkbox radioButton2 = new
java.awt.Checkbox();
    java.awt.Checkbox radioButton3 = new
java.awt.Checkbox();
    java.awt.Checkbox radioButton4 = new
java.awt.Checkbox();
    java.awt.Checkbox radioButton5 = new
java.awt.Checkbox();
    java.awt.Label label2 = new java.awt.Label();
    java.awt.Label label11 = new java.awt.Label();
    java.awt.Label label12 = new java.awt.Label();
    java.awt.Label label13 = new java.awt.Label();
    java.awt.Label label14 = new java.awt.Label();
    java.awt.TextField textField7 = new
java.awt.TextField();
    java.awt.TextField textField8 = new
java.awt.TextField();
    java.awt.TextField textField9 = new
java.awt.TextField();
    java.awt.TextField textField10 = new
java.awt.TextField();
    java.awt.TextField textField11 = new
java.awt.TextField();
    java.awt.TextField textField12 = new
java.awt.TextField();
    java.awt.TextField textField13 = new
java.awt.TextField();
    java.awt.Label label3 = new java.awt.Label();
    java.awt.Button button1 = new java.awt.Button();
    java.awt.Label label4 = new java.awt.Label();
    java.awt.TextField textField1 = new
java.awt.TextField();
    java.awt.Label label5 = new java.awt.Label();
    java.awt.TextField textField2 = new
java.awt.TextField();
    java.awt.Label label6 = new java.awt.Label();
```

```
    java.awt.TextField textField3 = new
java.awt.TextField();
    java.awt.Label label7 = new java.awt.Label();
    java.awt.TextField textField4 = new
java.awt.TextField();
    java.awt.Label label8 = new java.awt.Label();
    java.awt.Label label9 = new java.awt.Label();
    java.awt.TextField textField5 = new
java.awt.TextField();
    java.awt.Label label15 = new java.awt.Label();
    java.awt.Label label16 = new java.awt.Label();
    java.awt.Label label10 = new java.awt.Label();
    java.awt.TextField textField6 = new
java.awt.TextField();
    java.awt.TextArea textArea1 = new java.awt.TextArea();
    java.awt.Button button2 = new java.awt.Button();
    //aggiunte
    java.awt.Label label17 = new java.awt.Label();
    java.awt.TextField textField14 = new
java.awt.TextField();

    public void init() {
      identification = getParameter("identification");
      id = Integer.parseInt(identification);
      setLayout(null);
      setSize(364,398);
      activated = false;
      label1.setText("Scelta:");
      add(label1);
      label1.setFont(new Font("Dialog", Font.BOLD, 12));
      label1.setBounds(12,12,50,12);
      radioButton1.setCheckboxGroup(Group1);
      radioButton1.setLabel("A");
      add(radioButton1);
      radioButton1.setFont(new Font("Dialog", Font.BOLD,
12));
      radioButton1.setBounds(70,12,28,12);
      radioButton2.setCheckboxGroup(Group1);
      radioButton2.setLabel("B");
      add(radioButton2);
      radioButton2.setFont(new Font("Dialog", Font.BOLD,
12));
      radioButton2.setBounds(126,12,28,12);
      radioButton3.setCheckboxGroup(Group1);
      radioButton3.setLabel("C");
      add(radioButton3);
      radioButton3.setFont(new Font("Dialog", Font.BOLD,
12));
      radioButton3.setBounds(182,12,28,12);
      radioButton4.setCheckboxGroup(Group1);
      radioButton4.setLabel("D");
      add(radioButton4);
      radioButton4.setFont(new Font("Dialog", Font.BOLD,
12));
      radioButton4.setBounds(238,12,28,12);
      radioButton5.setCheckboxGroup(Group1);
      radioButton5.setLabel("E");
      add(radioButton5);
      radioButton5.setFont(new Font("Dialog", Font.BOLD,
12));
      radioButton5.setBounds(294,12,28,12);
      label2.setText("A:");
      label2.setAlignment(java.awt.Label.RIGHT);
```

```
      add(label2);
      label2.setBounds(50,36,20,24);
      add(textField7);
      textField7.setEditable(false);
      textField7.setBounds(70,36,30,24);
      label11.setText("B:");
      label11.setAlignment(java.awt.Label.RIGHT);
      add(label11);
      label11.setBounds(106,36,20,24);
      add(textField8);
      textField8.setEditable(false);
      textField8.setBounds(126,36,30,24);
      label12.setText("C:");
      label12.setAlignment(java.awt.Label.RIGHT);
      add(label12);
      label12.setBounds(162,36,20,24);
      add(textField9);
      textField9.setEditable(false);
      textField9.setBounds(182,36,30,24);
      label13.setText("D:");
      label13.setAlignment(java.awt.Label.RIGHT);
      add(label13);
      label13.setBounds(218,36,20,24);
      add(textField10);
      textField10.setEditable(false);
      textField10.setBounds(238,36,30,24);
      label14.setText("E:");
      label14.setAlignment(java.awt.Label.RIGHT);
      add(label14);
      label14.setBounds(274,36,20,24);
      add(textField11);
      textField11.setEditable(false);
      textField11.setBounds(294,36,30,24);
      //modificate
      label15.setText("Somma della tua terna:");
      label15.setAlignment(java.awt.Label.RIGHT);
      add(label15);
      label15.setBounds(12,218,130,24);
      add(textField12);
      textField12.setEditable(false);
      textField12.setBounds(150,218,40,24);
      label16.setText("Somma tuoi punti:");
      label16.setAlignment(java.awt.Label.RIGHT);
      add(label16);
      label16.setBounds(200,218,100,24);
      add(textField13);
      textField13.setEditable(false);
      textField13.setBounds(310,218,40,24);
      //fine
      //aggiunte
      label17.setText("Turno n.");
      label17.setAlignment(java.awt.Label.RIGHT);
      add(label17);
      label17.setBounds(120,184,50,24);
      add(textField14);
      textField14.setEditable(false);
      textField14.setBounds(170,184,40,24);

      label3.setText("Risultati dell\'ultimo incontro:");
      add(label3);
      label3.setFont(new Font("Dialog",
Font.BOLD|Font.ITALIC, 12));
      label3.setBounds(12,96,180,24);
```

```
      button1.setLabel("Invia i dati");
      add(button1);
      button1.setBackground(java.awt.Color.lightGray);
      button1.setBounds(12,70,340,24);
      button1.addActionListener(new ButtonHandler());
      label4.setText("Tua scelta:");
      label4.setAlignment(java.awt.Label.RIGHT);
      add(label4);
      label4.setBounds(12,120,83,24);
      add(textField1);
      textField1.setEditable(false);
      textField1.setBounds(95,120,40,24);
      label5.setText("Scelta di 1:");
      label5.setAlignment(java.awt.Label.RIGHT);
      add(label5);
      label5.setBounds(135,120,65,24);
      add(textField2);
      textField2.setEditable(false);
      textField2.setBounds(200,120,40,24);
      label6.setText("Scelta di 2:");
      label6.setAlignment(java.awt.Label.RIGHT);
      add(label6);
      label6.setBounds(240,120,72,24);
      add(textField3);
      textField3.setEditable(false);
      textField3.setBounds(312,120,40,24);

      label7.setText("Tuo punteggio:");
      label7.setAlignment(java.awt.Label.RIGHT);
      add(label7);
      label7.setBounds(12,156,83,24);
      add(textField4);
      textField4.setEditable(false);
      textField4.setBounds(95,156,40,24);
      label9.setText("Punti di 1:");
      label9.setAlignment(java.awt.Label.RIGHT);
      add(label9);
      label9.setBounds(135,156,65,24);
      add(textField5);
      textField5.setEditable(false);
      textField5.setBounds(200,156,40,24);
      label10.setText("Punti di 2:");
      label10.setAlignment(java.awt.Label.RIGHT);
      add(label10);
      label10.setBounds(240,156,72,24);
      add(textField6);
      textField6.setEditable(false);
      textField6.setBounds(312,156,40,24);

      button2.setLabel("Attiva");
      add(button2);
      button2.setBackground(java.awt.Color.lightGray);
      button2.setBounds(12,244,340,24);
      button2.addActionListener(new ButtonHandler());

      label8.setText("Messaggi:");
      add(label8);
      label8.setFont(new Font("Dialog",
Font.BOLD|Font.ITALIC, 12));
      label8.setBounds(12,268,132,24);
      add(textArea1);
      textArea1.setBounds(12,292,340,84);
```

```
      updated = true;


     if (id == 0 || id == 3 || id == 6 || id == 9 || id ==
12 || id == 15 || id == 18 || id == 21 || id == 24 || id ==
27 || id == 30 || id == 33 || id == 36 || id == 39 || id ==
42 || id == 45 || id == 48 || id == 51 || id == 54 || id ==
57 || id == 60 || id == 63 || id == 66 || id == 69 || id ==
72 || id == 75 || id == 78 || id == 81 || id == 84 || id ==
87 || id == 90 || id == 93 || id == 96 || id == 99 || id ==
102 || id == 105 || id == 108 || id == 111 || id == 114 ||
id == 117 || id == 120 || id == 123) {
          id1 = id + 1;
          id2 = id + 2;
      }
      if (id == 1 || id == 4 || id == 7 || id == 10 || id ==
13 || id == 16 || id == 19 || id == 22 || id == 25 || id ==
28 || id == 31 || id == 34 || id == 37 || id == 40 || id ==
43 || id == 46 || id == 49 || id == 52 || id == 55 || id ==
58 || id == 61 || id == 64 || id == 67 || id == 70 || id ==
73 || id == 76 || id == 79 || id == 82 || id == 85 || id ==
88 || id == 91 || id == 94 || id == 97 || id == 100 || id ==
103 || id == 106 || id == 109 || id == 112 || id == 115 ||
id == 118 || id == 121 || id == 124) {
          id1 = id - 1;
          id2 = id + 1;
      }
      if (id == 2 || id == 5 || id == 8 || id == 11 || id ==
14 || id == 17 || id == 20 || id == 23 || id == 26 || id ==
29 || id == 32 || id == 35 || id == 38 || id == 41 || id ==
44 || id == 47 || id == 50 || id == 53 || id == 56 || id ==
59 || id == 62 || id == 65 || id == 68 || id == 71 || id ==
74 || id == 77 || id == 80 || id == 83 || id == 86 || id ==
89 || id == 92 || id == 95 || id == 98 || id == 101 || id ==
104 || id == 107 || id == 110 || id == 113 || id == 116 ||
id == 119 || id == 122 || id == 125) {
          id1 = id - 2;
          id2 = id - 1;
      }
      if ((id >= 0 && id <=2) || (id >= 9 && id <=11) || (id
>= 18 && id <=20) || (id >= 27 && id <=29) || (id >= 36 &&
id <=38) || (id >= 45 && id <=47) || (id >= 54 && id <=56)
|| (id >= 63 && id <=65) || (id >= 72 && id <=74) || (id >=
81 && id <=83) || (id >= 90 && id <=92) || (id >= 99 && id
<=101) || (id >= 108 && id <=110) || (id >= 117 && id
<=119)) {
          id3 = id + 3;
          id4 = id + 6;
      }
      if ((id >= 3 && id <=5) || (id >= 12 && id <=14) ||
(id >= 21 && id <=23) || (id >= 30 && id <=32) || (id >= 39
&& id <=41) || (id >= 48 && id <=50) || (id >= 57 && id
<=59) || (id >= 66 && id <=68) || (id >= 75 && id <=77) ||
(id >= 84 && id <=86) || (id >= 93 && id <=95) || (id >= 102
&& id <=104) || (id >= 111 && id <=113) || (id >= 120 && id
<=122)) {
          id3 = id - 3;
          id4 = id + 3;
      }
      if ((id >= 6 && id <=8) || (id >= 15 && id <=17) ||
(id >= 24 && id <=26) || (id >= 33 && id <=35) || (id >= 42
&& id <=44) || (id >= 51 && id <=53) || (id >= 60 && id
<=62) || (id >= 69 && id <=71) || (id >= 78 && id <=80) ||
```

```
(id >= 87 && id <=89) || (id >= 96 && id <=98) || (id >= 105
&& id <=107) || (id >= 114 && id <=116) || (id >= 123 && id
<=125)) {
          id3 = id - 6;
          id4 = id - 3;
      }
    }
    public boolean askRound(int i){
      boolean aa = false;
      try  {
          aa = server.askRound(i);
      } catch (Exception ex) {
          textArea1.setText(ex.toString());
      }
      return aa;
      }
    public boolean askBlock2(){
      boolean aa = true;
      try  {
          aa = server.askBlock2();
      } catch (Exception ex) {
          textArea1.setText(ex.toString());
      }
      return aa;
    }

    public void update () {
      try  {
                        //Update information
          textField1.setText(Integer.toString(value));
          int newChoice1 = server.getChoice(id1);
          textField2.setText(Integer.toString(newChoice1));
          int newChoice2 = server.getChoice(id2);
          textField3.setText(Integer.toString(newChoice2));
          int newPoints = server.getPoints(id);
          textField4.setText(Integer.toString(newPoints));
          int newPoints1 = server.getPoints(id1);
          textField5.setText(Integer.toString(newPoints1));
          int newPoints2 = server.getPoints(id2);
          textField6.setText(Integer.toString(newPoints2));

textField12.setText(Integer.toString(value+newChoice1+newCho
ice2));
          int newCumPoints = server.getCumPoints(id);

textField13.setText(Integer.toString(newCumPoints));
                        //aggiunte
          int newNumRound = server.askNumRound();
          textField14.setText(Integer.toString(newNumRound +
1));

          updated = true;
          if (server.askEnd())
            textArea1.setText(" -- L'ESPERIMENTO E'
TERMINATO!!! --\n -- L'ESPERIMENTO E' TERMINATO!!! --\n --
L'ESPERIMENTO E' TERMINATO!!! --");
          else {
            if (server.askChange()) {
                textArea1.setText(" -- ATTENZIONE! Da ora
affronti nuovi giocatori! --\n -- ATTENZIONE! Da ora
affronti nuovi giocatori! --\n -- ATTENZIONE! Da ora
affronti nuovi giocatori! --");
                id1 = id3;
```

```
                id2 = id4;
             }else
                textArea1.setText("Risultati aggiornati, ora
puoi effettuare una nuova scelta.");
          }
      } catch (Exception ex) {
          textArea1.setText(ex.toString());
      }
    }

    public void activate () {
      try {
          URL hostURL = getCodeBase();
          String host = hostURL.getHost();
          // Change the port number if necessary
          server =
(RemoteSum10)Naming.lookup("//"+host+":1098/RemoteSum10");
          } catch (Exception ex) {
          textArea1.setText(ex.toString());
          }
      try  {
          payoffArray = server.getPayoffArray();
      } catch (Exception ex) {
          textArea1.setText(ex.toString());
      }

      // A message to tell players that the first round is
an example
      //textArea1.setText("ATTENZIONE:\nil primo turno e' di
prova!!!");
      textField7.setText(Integer.toString(payoffArray[0]));
      textField8.setText(Integer.toString(payoffArray[1]));
      textField9.setText(Integer.toString(payoffArray[2]));
      textField10.setText(Integer.toString(payoffArray[3]));
      textField11.setText(Integer.toString(payoffArray[4]));
      activated = true;
    }

    class ButtonHandler implements ActionListener {
      public void actionPerformed(ActionEvent ev) {
          String s=ev.getActionCommand();

          if ("Invia i dati".equals(s)) {
            if (Group1.getSelectedCheckbox() == null) {
                textArea1.setText("Occorre selezionare la
propria scelta.");
            }
            else {
                if (!(askRound(id))) {
                  if (updated) {
                      try  {
                        //Submit values
                        valueS =
Group1.getSelectedCheckbox().getLabel();
                      if (valueS == "A")
                          value = payoffArray[0];
                      if (valueS == "B")
                          value = payoffArray[1];
                      if (valueS == "C")
                          value = payoffArray[2];
                      if (valueS == "D")
                          value = payoffArray[3];
                      if (valueS == "E")
```

```
                          value = payoffArray[4];
                        server.setChoice(id, value);
                        updated = false;
                        textArea1.setText("Ok, adesso puoi
provare ad aggiornare i risultati.");
                      } catch (Exception ex) {
                        textArea1.setText(ex.toString());
                      }
                  }
                  else {
                      textArea1.setText("Prima devi
aggiornare i risultati!!!");
                  }
                }
                else {
                  textArea1.setText("Prima devi aggiornare i
risultati!!!");
                }
              }
            }


            if ("Attiva".equals(s)) {
              activate();
              textArea1.setText("Ora puoi iniziare a
giocare.");
              button2.setLabel("Aggiorna i risultati");
            }
            if ("Aggiorna i risultati".equals(s)) {
              if ((askRound(id))) {
                  if ((askBlock2())) {
                    if (!(updated)) {
                        try  {
                          //Update information

      textField1.setText(Integer.toString(value));
                          int newChoice1 =
server.getChoice(id1);

      textField2.setText(Integer.toString(newChoice1));
                          int newChoice2 =
server.getChoice(id2);

      textField3.setText(Integer.toString(newChoice2));
                          int newPoints = server.getPoints(id);

      textField4.setText(Integer.toString(newPoints));
                          int newPoints1 =
server.getPoints(id1);

      textField5.setText(Integer.toString(newPoints1));
                          int newPoints2 =
server.getPoints(id2);

      textField6.setText(Integer.toString(newPoints2));

      textField12.setText(Integer.toString(value+newChoice1+
newChoice2));
                          int newCumPoints =
server.getCumPoints(id);

      textField13.setText(Integer.toString(newCumPoints));
                          //aggiunte
```

```
                        int newNumRound =
server.askNumRound() + 1;

      textField14.setText(Integer.toString(newNumRound));

                        updated = true;
                        if (server.askEnd())
                            textArea1.setText(" --
L'ESPERIMENTO E' TERMINATO!!! --\n -- L'ESPERIMENTO E'
TERMINATO!!! --\n -- L'ESPERIMENTO E' TERMINATO!!! --");
                        else
                            {
                              if (server.askChange()) {
                                  textArea1.setText(" --
ATTENZIONE! Da ora affronti nuovi giocatori! --\n --
ATTENZIONE! Da ora affronti nuovi giocatori! --\n --
ATTENZIONE! Da ora affronti nuovi giocatori! --");
                                  id1 = id3;
                                  id2 = id4;
                              }
                              else

textArea1.setText("Risultati aggiornati, ora puoi effettuare
una nuova scelta.");
                            }
                        try  {
                            server.setBlock3();
                        } catch (Exception ex) {
                            textArea1.setText(ex.toString());
                        }
                    } catch (Exception ex) {
                      textArea1.setText(ex.toString());
                    }
                }
                else {
                    textArea1.setText("Risultati gia\'
aggiornati.");
                }
              }
              else {
                textArea1.setText("Perfavore aspetta la
scelta degli altri\ne quindi invia nuovamente la tua
scelta.");
              }
            }
            else {
                textArea1.setText("Perfavore aspetta alcuni
secondi la scelta\ndegli altri giocatori e riprova\nad
aggiornare i risultati");
            }

          }
      }
    }
}
```

Sum10ModelSwarm.java

```
import swarm.Globals;
import swarm.Selector;
import swarm.defobj.Zone;
```

```
import swarm.defobj.SymbolImpl;

import swarm.activity.Activity;
import swarm.activity.ActionGroup;
import swarm.activity.ActionGroupImpl;
import swarm.activity.Schedule;
import swarm.activity.ScheduleImpl;
import swarm.activity.ActionForEach;

import swarm.collections.List;
import swarm.collections.ListImpl;

import swarm.objectbase.Swarm;
import swarm.objectbase.SwarmImpl;
import swarm.objectbase.VarProbe;
import swarm.objectbase.MessageProbe;
import swarm.objectbase.EmptyProbeMapImpl;

import java.io.*;

/** The Sum10ModelSwarm encapsulates all the objects used in
the
  simulated world itself (but not the user interface
objects) */

public class Sum10ModelSwarm extends SwarmImpl {

    // simulation parameters
    public int numPlayers, numCycles, payoffArray[];

    // ActionGroup for holding an ordered sequence of action
   public ActionGroup modelActions;

    // the single Schedule
    public Schedule modelSchedule;

    // list of producers
    public List playerList;

    // the world of the economic simulation
    //public Market market;
    public Player playerArray[];
    //public Player player2;
    public Timer timer;
    public RuleMaster ruleMaster;
    public RemoteServer remoteServer;
    public Sum10ObserverSwarm sum10ObserverSwarm;
    public int a, b, c, d, e;
    //public int aA1, aA2, aA3, aA4, bA1, bA2, bA3, bA4,
cA1, cA2, cA3, cA4, dA1, dA2, dA3, dA4;
    //public int aB1, aB2, aB3, aB4, bB1, bB2, bB3, bB4,
cB1, cB2, cB3, cB4, dB1, dB2, dB3, dB4;
    // data file
    public String dataFile;

    // file for output datas
    public FileWriter file;

    // buffered stream for output datas
    public PrintWriter buff;
```

```
    // methods to provide access to the objects inside the
model
    public Player[] getPlayerArray () {
      return playerArray;
    }

    // Constructor of Model Swarm
    public Sum10ModelSwarm (Zone aZone, Sum10ObserverSwarm
o) {
      super (aZone);
      sum10ObserverSwarm = o;

      // simulation parameters
      numPlayers = 27;
      numCycles = 30;

      //payoffs

      a = 0;
      b = 1;
      c = 3;
      d = 4;
      e = 10;

      //data storage file
      dataFile = "data/run1.dat";

      //  customized probe map
      class Sum10ModelProbeMap extends EmptyProbeMapImpl {
          private VarProbe probeVariable (String name) {
            return

Globals.env.probeLibrary.getProbeForVariable$inClass
                (name, Sum10ModelSwarm.this.getClass ());
          }
          private MessageProbe probeMessage (String name) {
            return

Globals.env.probeLibrary.getProbeForMessage$inClass
                (name, Sum10ModelSwarm.this.getClass ());
          }
          private void addVar (String name) {
            addProbe (probeVariable (name));
          }
          private void addMessage (String name) {
            addProbe (probeMessage (name));
          }
          public Sum10ModelProbeMap (Zone _aZone, Class
aClass) {
            super (_aZone, aClass);
            addVar ("numPlayers");
            addVar ("numCycles");
            addVar ("a");
            addVar ("b");
            addVar ("c");
            addVar ("d");
            addVar ("e");
            addVar ("dataFile");
            //addVar ("payoffArray[1]");
            //addMessage ("addProducer");
            }
          }
```

```
      // installing of custom probeMap class directly into
the
      // probeLibrary
      Globals.env.probeLibrary.setProbeMap$For
      (new Sum10ModelProbeMap (aZone, getClass ()), getClass
());
    }

    // building of model objects
    public Object buildObjects () {
      int i;


      // allow our parent class to build anything
      super.buildObjects();

      payoffArray = new int[5];
      payoffArray[0] = a;
      payoffArray[1] = b;
      payoffArray[2] = c;
      payoffArray[3] = d;
      payoffArray[4] = e;

      // saving datas
      try {

      // file for output datas
      file = new FileWriter (dataFile);

      // build buffered stream for output datas
      buff = new PrintWriter (file);
      }
      catch (IOException e){
          System.err.println ("Exception in building data-
saving objects: "
                       + e.toString ());
      }
      buff.println ("ID Scelta Punti PuntiTotale
SommaTerna");

      // setting up objects used to represent human players
      playerList = new ListImpl (getZone ());
      timer = new Timer(numPlayers, sum10ObserverSwarm,
numCycles, buff);
      ruleMaster = new RuleMaster(timer, numPlayers);
      playerArray = new Player[numPlayers];

      // a loop to create players and put them in the list
      for (i = 0; i < playerArray.length; i++) {
            playerArray[i] = new Player (numPlayers,
payoffArray, i, playerArray, timer, ruleMaster);
          System.out.println("Built Player" + i);
          playerList.addLast (playerArray[i]);
      }

      // server class to manage RMI - Remote Method
Invocation
      remoteServer = new RemoteServer(playerArray, timer,
payoffArray);

      return this;
    }
```

```
    // Model Schedule
    public Object buildActions () {
      super.buildActions();

      // create the list of simulation actions. We put these
in
      // an action group, because we want these actions to
be
      // executed in a specific order, but these steps
should
      // take no (simulated) timebuild.
      modelActions = new ActionGroupImpl (getZone ());

      // a step to check number of cycles
      try {
          modelActions.createActionTo$message
            (timer, new Selector (timer.getClass (),
"checkCycle", false));
      } catch (Exception e) {
          System.err.println("Exception in timer.checkCycle:
" + e.getMessage ());
      }

      // main step
      try {
          ActionForEach actionForEach;
          Selector sel =
            new Selector (Class.forName ("Player"), "step",
false);

          actionForEach =
            modelActions.createActionForEach$message
(playerList, sel);
      }
      catch (Exception e) {
          System.err.println("Exception in player.step: " +
e.getMessage ());
      }

      try {
          ActionForEach actionForEach;
          Selector sel =
            new Selector (Class.forName ("Player"), "step2",
false);

          actionForEach =
            modelActions.createActionForEach$message
(playerList, sel);
      }
      catch (Exception e) {
          System.err.println("Exception in player.step: " +
e.getMessage ());
      }

      // a step to make counters for applet-player
synchronization = 0
      try {
          modelActions.createActionTo$message
            (timer, new Selector (timer.getClass (),
"nullBlock", false));
      } catch (Exception e) {
          System.err.println("Exception in timer.nullBlock:
" + e.getMessage ());
```

```
      }

      // a step to make the counter of access to
instructions page = 0
      try {
          ActionForEach actionForEach;
          Selector sel =
            new Selector (Class.forName ("Player"),
"nullInstructions", false);

          actionForEach =
            modelActions.createActionForEach$message
(playerList, sel);
      }
      catch (Exception e) {
          System.err.println("Exception in player.step: " +
e.getMessage ());
      }

      //This schedule has a repeat interval of 1, it will
loop every
      // time step.  The action is executed at time 0
relative to
      // the beginning of the loop.
      // This is a simple schedule, with only three action
that are
      // just repeated every time.
          modelSchedule = new ScheduleImpl (getZone (), 1);
    modelSchedule.at$createAction (0, modelActions);

          return this;
    }

    // this model is run as a subswarm of an observer swarm
    public Activity activateIn (Swarm swarmContext) {

      // activation of ourselves via the
      // superclass activateIn: method
      super.activateIn (swarmContext);

      // activation of our own schedule
      modelSchedule.activateIn (this);

      // return our activity
      return getActivity ();
    }
}
```

Sum10ObserverSwarm.java

```
import swarm.Globals;
import swarm.Selector;
import swarm.defobj.Zone;
import swarm.defobj.SymbolImpl;

import swarm.activity.Activity;
import swarm.activity.ActionGroup;
import swarm.activity.ActionGroupImpl;
import swarm.activity.Schedule;
import swarm.activity.ScheduleImpl;
import swarm.activity.ActionForEach;
```

```
import swarm.collections.List;
import swarm.collections.ListImpl;

import swarm.objectbase.Swarm;
import swarm.objectbase.SwarmImpl;
import swarm.objectbase.VarProbe;
import swarm.objectbase.MessageProbe;
import swarm.objectbase.EmptyProbeMapImpl;

import java.io.*;

/** The Sum10ModelSwarm encapsulates all the objects used in
the
  simulated world itself (but not the user interface
objects) */

public class Sum10ModelSwarm extends SwarmImpl {

    // simulation parameters
    public int numPlayers, numCycles, payoffArray[];

    // ActionGroup for holding an ordered sequence of action
   public ActionGroup modelActions;

    // the single Schedule
    public Schedule modelSchedule;

    // list of producers
    public List playerList;

    // the world of the economic simulation
    public Player playerArray[];
    public Timer timer;
    public RuleMaster ruleMaster;
    public RemoteServer remoteServer;
    public Sum10ObserverSwarm sum10ObserverSwarm;
    public int a, b, c, d, e;
    // data file
    public String dataFile;

    // file for output datas
    public FileWriter file;

    // buffered stream for output datas
    public PrintWriter buff;


    // methods to provide access to the objects inside the
model
    public Player[] getPlayerArray () {
      return playerArray;
    }

    // Constructor of Model Swarm
    public Sum10ModelSwarm (Zone aZone, Sum10ObserverSwarm
o) {
      super (aZone);
      sum10ObserverSwarm = o;

      // simulation parameters
      numPlayers = 27;
      numCycles = 30;
```

```
      //payoffs
      a = 0;
      b = 1;
      c = 3;
      d = 4;
      e = 10;

      //data storage file
      dataFile = "data/run1.dat";

      //  customized probe map
      class Sum10ModelProbeMap extends EmptyProbeMapImpl {
          private VarProbe probeVariable (String name) {
            return

Globals.env.probeLibrary.getProbeForVariable$inClass
                (name, Sum10ModelSwarm.this.getClass ());
          }
          private MessageProbe probeMessage (String name) {
            return

Globals.env.probeLibrary.getProbeForMessage$inClass
                (name, Sum10ModelSwarm.this.getClass ());
          }
          private void addVar (String name) {
            addProbe (probeVariable (name));
          }
          private void addMessage (String name) {
            addProbe (probeMessage (name));
          }
          public Sum10ModelProbeMap (Zone _aZone, Class
aClass) {
            super (_aZone, aClass);
            addVar ("numPlayers");
            addVar ("numCycles");

            addVar ("a");
            addVar ("b");
            addVar ("c");
            addVar ("d");
            addVar ("e");
            addVar ("dataFile");
            //addVar ("payoffArray[1]");
            //addMessage ("addProducer");
            }
          }

      // installing of custom probeMap class directly into
the
      // probeLibrary
      Globals.env.probeLibrary.setProbeMap$For
      (new Sum10ModelProbeMap (aZone, getClass ()), getClass
());
    }

    // building of model objects
    public Object buildObjects () {
      int i;


      // allow our parent class to build anything
      super.buildObjects();
```

```
      /**payoffArray = new int[4][8];

      payoffArray = new int[5];
      payoffArray[0] = a;
      payoffArray[1] = b;
      payoffArray[2] = c;
      payoffArray[3] = d;
      payoffArray[4] = e;

      // saving datas
      try {

      // file for output datas
      file = new FileWriter (dataFile);

      // build buffered stream for output datas
      buff = new PrintWriter (file);
      }
      catch (IOException e){
          System.err.println ("Exception in building data-
saving objects: "
                       + e.toString ());
      }
      buff.println ("ID Scelta Punti PuntiTotale
SommaTerna");

      // setting up objects used to represent human players
      playerList = new ListImpl (getZone ());
      timer = new Timer(numPlayers, sum10ObserverSwarm,
numCycles, buff);
      ruleMaster = new RuleMaster(timer, numPlayers);
      playerArray = new Player[numPlayers];

      // a loop to create players and put them in the list
      for (i = 0; i < playerArray.length; i++) {
            playerArray[i] = new Player (numPlayers,
payoffArray, i, playerArray, timer, ruleMaster);
          System.out.println("Built Player" + i);
          playerList.addLast (playerArray[i]);
      }

      // server class to manage RMI - Remote Method
Invocation
      remoteServer = new RemoteServer(playerArray, timer,
payoffArray);

      return this;
    }

    // Model Schedule
    public Object buildActions () {
      super.buildActions();

      // create the list of simulation actions. We put these
in
      // an action group, because we want these actions to
be
      // executed in a specific order, but these steps
should
      // take no (simulated) timebuild.
      modelActions = new ActionGroupImpl (getZone ());
```

```
      // a step to check number of cycles
      try {
          modelActions.createActionTo$message
            (timer, new Selector (timer.getClass (),
"checkCycle", false));
      } catch (Exception e) {
          System.err.println("Exception in timer.checkCycle:
" + e.getMessage ());
      }

      // main step
      try {
          ActionForEach actionForEach;
          Selector sel =
            new Selector (Class.forName ("Player"), "step",
false);

          actionForEach =
            modelActions.createActionForEach$message
(playerList, sel);
      }
      catch (Exception e) {
          System.err.println("Exception in player.step: " +
e.getMessage ());
      }

      try {
          ActionForEach actionForEach;
          Selector sel =
            new Selector (Class.forName ("Player"), "step2",
false);

          actionForEach =
            modelActions.createActionForEach$message
(playerList, sel);
      }
      catch (Exception e) {
          System.err.println("Exception in player.step: " +
e.getMessage ());
      }

      // a step to make counters for applet-player
synchronization = 0
      try {
          modelActions.createActionTo$message
            (timer, new Selector (timer.getClass (),
"nullBlock", false));
      } catch (Exception e) {
          System.err.println("Exception in timer.nullBlock:
" + e.getMessage ());
      }

      // a step to make the counter of access to
instructions page = 0
      try {
          ActionForEach actionForEach;
          Selector sel =
            new Selector (Class.forName ("Player"),
"nullInstructions", false);

          actionForEach =
            modelActions.createActionForEach$message
(playerList, sel);
```

```
      }
      catch (Exception e) {
          System.err.println("Exception in player.step: " +
e.getMessage ());
      }

      //This schedule has a repeat interval of 1, it will
loop every
      // time step.  The action is executed at time 0
relative to
      // the beginning of the loop.
      // This is a simple schedule, with only three action
that are
      // just repeated every time.
          modelSchedule = new ScheduleImpl (getZone (), 1);
    modelSchedule.at$createAction (0, modelActions);

          return this;
    }

    // this model is run as a subswarm of an observer swarm
    public Activity activateIn (Swarm swarmContext) {

      // activation of ourselves via the
      // superclass activateIn: method
      super.activateIn (swarmContext);

      // activation of our own schedule
      modelSchedule.activateIn (this);

      // return our activity
      return getActivity ();
    }
}
```

Timer.java

```
import swarm.Globals;
import swarm.Selector;
import swarm.defobj.Zone;
import swarm.defobj.SymbolImpl;

import swarm.activity.Activity;
import swarm.activity.ActionGroup;
import swarm.activity.ActionGroupImpl;
import swarm.activity.Schedule;
import swarm.activity.ScheduleImpl;
import swarm.activity.ActionForEach;

import swarm.collections.List;
import swarm.collections.ListImpl;

import swarm.objectbase.Swarm;
import swarm.objectbase.SwarmImpl;
import swarm.objectbase.VarProbe;
import swarm.objectbase.MessageProbe;
import swarm.objectbase.EmptyProbeMapImpl;

import java.io.*;
```

```
/** The Sum10ModelSwarm encapsulates all the objects used in
the
  simulated world itself (but not the user interface
objects) */

public class Sum10ModelSwarm extends SwarmImpl {

    // simulation parameters
    public int numPlayers, numCycles, payoffArray[];

    // ActionGroup for holding an ordered sequence of action
   public ActionGroup modelActions;

    // the single Schedule
    public Schedule modelSchedule;

    // list of producers
    public List playerList;

    // the world of the economic simulation
    public Player playerArray[];
    public Timer timer;
    public RuleMaster ruleMaster;
    public RemoteServer remoteServer;
    public Sum10ObserverSwarm sum10ObserverSwarm;
    public int a, b, c, d, e;
    // data file
    public String dataFile;

    // file for output datas
    public FileWriter file;

    // buffered stream for output datas
    public PrintWriter buff;


    // methods to provide access to the objects inside the
model
    public Player[] getPlayerArray () {
      return playerArray;
    }

    // Constructor of Model Swarm
    public Sum10ModelSwarm (Zone aZone, Sum10ObserverSwarm
o) {
      super (aZone);
      sum10ObserverSwarm = o;

      // simulation parameters
      numPlayers = 27;
      numCycles = 30;

      //payoffs
      a = 0;
      b = 1;
      c = 3;
      d = 4;
      e = 10;

      //data storage file
      dataFile = "data/run1.dat";

      //  customized probe map
```

```
      class Sum10ModelProbeMap extends EmptyProbeMapImpl {
          private VarProbe probeVariable (String name) {
            return

Globals.env.probeLibrary.getProbeForVariable$inClass
                (name, Sum10ModelSwarm.this.getClass ());
          }
          private MessageProbe probeMessage (String name) {
            return

Globals.env.probeLibrary.getProbeForMessage$inClass
                (name, Sum10ModelSwarm.this.getClass ());
          }
          private void addVar (String name) {
            addProbe (probeVariable (name));
          }
          private void addMessage (String name) {
            addProbe (probeMessage (name));
          }
          public Sum10ModelProbeMap (Zone _aZone, Class
aClass) {
            super (_aZone, aClass);
            addVar ("numPlayers");
            addVar ("numCycles");

            addVar ("a");
            addVar ("b");
            addVar ("c");
            addVar ("d");
            addVar ("e");
            addVar ("dataFile");
            //addVar ("payoffArray[1]");
            //addMessage ("addProducer");
            }
          }

      // installing of custom probeMap class directly into
the
      // probeLibrary
      Globals.env.probeLibrary.setProbeMap$For
      (new Sum10ModelProbeMap (aZone, getClass ()), getClass
());
    }

    // building of model objects
    public Object buildObjects () {
      int i;


      // allow our parent class to build anything
      super.buildObjects();


      payoffArray = new int[5];
      payoffArray[0] = a;
      payoffArray[1] = b;
      payoffArray[2] = c;
      payoffArray[3] = d;
      payoffArray[4] = e;

      // saving datas
      try {
```

```
      // file for output datas
      file = new FileWriter (dataFile);

      // build buffered stream for output datas
      buff = new PrintWriter (file);
      }
      catch (IOException e){
          System.err.println ("Exception in building data-
saving objects: "
                        + e.toString ());
      }
      buff.println ("ID Scelta Punti PuntiTotale
SommaTerna");

      // setting up objects used to represent human players
      playerList = new ListImpl (getZone ());
      timer = new Timer(numPlayers, sum10ObserverSwarm,
numCycles, buff);
      ruleMaster = new RuleMaster(timer, numPlayers);
      playerArray = new Player[numPlayers];

      // a loop to create players and put them in the list
      for (i = 0; i < playerArray.length; i++) {
            playerArray[i] = new Player (numPlayers,
payoffArray, i, playerArray, timer, ruleMaster);
          System.out.println("Built Player" + i);
          playerList.addLast (playerArray[i]);
      }

      // server class to manage RMI - Remote Method
Invocation
      remoteServer = new RemoteServer(playerArray, timer,
payoffArray);

      return this;
    }

    // Model Schedule
    public Object buildActions () {
      super.buildActions();

      // create the list of simulation actions. We put these
in
      // an action group, because we want these actions to
be
      // executed in a specific order, but these steps
should
      // take no (simulated) timebuild.
      modelActions = new ActionGroupImpl (getZone ());

      // a step to check number of cycles
      try {
          modelActions.createActionTo$message
            (timer, new Selector (timer.getClass (),
"checkCycle", false));
      } catch (Exception e) {
          System.err.println("Exception in timer.checkCycle:
" + e.getMessage ());
      }

      // main step
      try {
          ActionForEach actionForEach;
```

```
          Selector sel =
            new Selector (Class.forName ("Player"), "step",
false);

          actionForEach =
            modelActions.createActionForEach$message
(playerList, sel);
      }
      catch (Exception e) {
           System.err.println("Exception in player.step: " +
e.getMessage ());
      }

      try {
          ActionForEach actionForEach;
          Selector sel =
            new Selector (Class.forName ("Player"), "step2",
false);

          actionForEach =
            modelActions.createActionForEach$message
(playerList, sel);
      }
      catch (Exception e) {
          System.err.println("Exception in player.step: " +
e.getMessage ());
      }

      // a step to make counters for applet-player
synchronization = 0
      try {
          modelActions.createActionTo$message
            (timer, new Selector (timer.getClass (),
"nullBlock", false));
      } catch (Exception e) {
          System.err.println("Exception in timer.nullBlock:
" + e.getMessage ());
      }

      // a step to make the counter of access to
instructions page = 0
      try {
          ActionForEach actionForEach;
          Selector sel =
            new Selector (Class.forName ("Player"),
"nullInstructions", false);

          actionForEach =
            modelActions.createActionForEach$message
(playerList, sel);
      }
      catch (Exception e) {
          System.err.println("Exception in player.step: " +
e.getMessage ());
      }

      //This schedule has a repeat interval of 1, it will
loop every
      // time step.  The action is executed at time 0
relative to
      // the beginning of the loop.
      // This is a simple schedule, with only three action
that are
```

```
      // just repeated every time.
          modelSchedule = new ScheduleImpl (getZone (), 1);
    modelSchedule.at$createAction (0, modelActions);

          return this;
    }

    // this model is run as a subswarm of an observer swarm
    public Activity activateIn (Swarm swarmContext) {

      // activation of ourselves via the
      // superclass activateIn: method
      super.activateIn (swarmContext);

      // activation of our own schedule
      modelSchedule.activateIn (this);

      // return our activity
      return getActivity ();
    }
}
```

BIBLIOGRAFIA

- Akerlof G.A. (1984), *Gift Exchange and Efficiency-Wage Theory: Four Views*, in "The American Economic Review", Volume 74, Issue 2, Papers and Proceedings of the Ninety-Sixth Annual Meeting of the American Economic Association (May), pp 79-83

- Andreoni J. (1995), *Cooperation in Public-Goods Experiments: Kindness or Confusion?*, in "The American Economic Review", Volume 85, Issue 4 (Sep.), pp. 891-904

- Andreoni J. e Miller J.H. (1993), *Rational Cooperation in the Finitely Repeated Prisoner's Dilemma: Experimental Evidence*, in "The Economic Journal", Volume 103, Issue 418 (May), pp. 570-585

- Axelrod R. (1984), *The Evolution of Cooperation*, Basic Books, New York

- Axelrod R. (1981), *The Emergence of Cooperation Among Egoists*, in "American Political Science Review", 75, pp. 306-318

- Axtell R. L. ed Epstein J. M. (1994), *Agent-based modeling: understanding our creations*, in "The bulletin of the Santa Fe Institute", winter.

- Becker G.S. (1976), *Altruism, Egoism, and Genetic Fitness: Economics and Sociobiology*, in "Journal of Economic Literature", Volume 14, Issue 3 (sep), pp. 817-826

- Bergstrom T.C. e Stark O. (1993), *How altruism can prevail in a Evolutionary Environment*, in "The American Economics Rewievs", Volume. 83, Issue 2 (May), pp. 149-155

- Camerer Colin F. (2001), *Behavioral Economics*, Princeton University Press, Princeton

- Cosmides L. e Tooby J. (1992), *Cognitive Adaptations for Social Exchange*, in J.H. Barkow, L. Cosmides e J. Tooby (eds), in "The Adapted Mind", Oxford University Press, New York, pp. 163-228

- Eisenberg N. Miller P.A. (1987), *Empathy, Sympathy, and altruism: Empirical and Conceptual Links*, in "Empathy and its Development", Eisenberg N.e Strayer J. (eds), , Cambridge, Cambridge University Press, pp. 292-316

- Fehr Ernst e Gachter Simon (2000), *"Fairness and Retaliation: The Economics of Reciprocity"*, in "Journal of Economic Perspectives" 14, 3 pp 159-183

- Henrich Joseph, Boyd Robert, Bowles Samuel, Camerer Colin, Feht Ernst, Gintis Herbert e McElreath Richard (maggio 2001), "*Cooperation, Reciprocity and Punishment in Fifteen Small-scale Societies"*, American Economics Review

- Hey John D. (1991), *Experiments in Economics*, Blackwell, Oxford, trad. it. (1998), *Esperimenti in Economia*, G. Giappichelli, Torino

- Innocenti A. (1995), *Le origini della experimental economics (1948-1959): una valutazione critica sull'evoluzione delle metodologie sperimentali*, working papers

- Kagel John H. e Roth Alvin E. (1995), *Handbook of Experimental Economics*, Princeton University Press, Princeton

- Kahneman, Knetsch, e Thaler (1986), *Fairness as a Constraint on Profit Seeking: Entitlements in the Market*, in "The American Economic Review", volume 76, Issue 4, (Sep), pp. 728-741

- Kirman A. (1992), *Whom or what does the rapresentative individual represent?*, in "Journal of Economic Perspectives, volume 6, numero 2, pp. 117-136

- Krebs D. (1982), *Psychological Approaches to Altruism: An Evaluation*, Ethics, 92 (3), April pp. 447-58

- Langton C. G. (1992), *Vita artificiale*, in "Sistemi intelligenti", tradotto da Maurizio Riccucci volume 2, pp. 189-246,

- Mark R. Rosenzweig e Kenneth I. Wolpin (Dicembre 2000), *Natural "Natural Experiments" in Economics*, in "Journal of Economic Literature", vol. XXXVIII, pp. 827-874

- Morishima M. (1982), *Why Has Japan 'Succeede'? Western Technology and Japanese Ethos*, Cambridge, Cambridge University Press

- Novarese M. e Rizzello S. (2000), *Economia sperimentale e analisi empirica dei processi di apprendimento*, Laboratorio di Economia Cognitiva - Università del Piemonte Orientale, Alessandria, working paper

- Ochs J. (1995), *Coordination Problems*, in J.H. Kagel & A.E. Roth (eds) The Handbook of Experimental Economics, Princeton University Press, Princeton, pp. 195-251

- Parisi Domenico (2001), *Simulazioni, la realtà rifatta al computer*, il Mulino, Bologna

- Rabin M. (1993), *Incorporating Fairness into Game Theory and Economics*, in "The American Economic Review", Volume 83, Issue 5 (Dec), pp. 1281-1302

- Rotemberg J.J. (1994), *Human Relations in the Workplace*, in "The Journal of Political Economy", Volume 102, Issue 4 (Aug), pp. 684-717

- Roth Alvin, Prasnikan Vesna, Okuno-Fujiwara Masahiro, Zamir Shmuel (1991), *"Bargaining and Market Behavior in Jerusalem, Ljublliana, Pittsburgh and Tokyo: An Experimental Study"*, in "American Economic Rreview"

- Sacco P.L. e Zamagni S. (1994), *Un'approccio dinamico evolutivo all'altruismo*, in "Rivista Internazionale di Scienze Sociali", 4-6, CII:2, pp. 223-262

- Samuelson P. e Nordhaus W.D. (1985), *Principles of Economics*, XII ed., McGraw-Hill, New York

- Shafir E. e Tversky A. (1992), *Thinking through uncertainty: Nonconsequential reasoning and choiche*, Cognitive Psychology, 24, pp. 449-74

- Simon H.A. (1983), *Reason in Human Affairs*, Stanford, CA, Stanford University Press, trad. it. Simon (1984), *La ragione nelle vicende umane*, Il Mulino, Bologna

- Simon H.A. (1993), *Altruism and Economics*, in "The American Economics Rewievs", Volume 83, Issue 2 (May), pp. 156-161

- Skyrms B. (1996), *The Evolution of Social Contract*, Cambridge, Cambridge University Press

- Trivers L. (1971), *The evolution of reciprocal altruism*, Quarterly Review of Biology, 46, pp. 35-57

- Tullock G. (1985), *Adam Smith and the Prisoners' Dilemma*, in "The Quarterly Journal of Economics", Volume 100, Issue Supplement, pp. 1073-1081

- Tullock G. (1999), *Non-prisoner's dilemma*, in "Journal of Economic Behavior & Organization", Vol. 39, pp. 455-458

- Williams G.C. (1966), *Adaptation and Natural Selection: A Critique of some current evolutionary thought*, Princeton University Press, Princeton

Indice

Introduzione

Parte prima

Verificare l'economia con tecniche sperimentali

Parte seconda

Il modello canonico delle scelte individuali

Parte terza
Altruismo e cooperazione

Parte quarta
Swarm

Parte sesta

Esperimenti