UNIVERSITA' DEGLI STUDI DI TORINO
Facoltà di Economia
Corso di laurea in economia e commercio

TESI DI LAUREA

IL DILEMMA DEL PRIGIONIERO RIPETUTO: SIMULAZIONI MEDIANTI AGENTI INDIPENDENTI

Relatore: prof. **PIETRO TERNA**
Correlatore:

CANDIDATO

Piergiuseppe Merlo

ANNO ACCADEMICO 1998/99

Introduzione

L'aumentata capacità e velocità di calcolo degli elaboratori elettronici, fino ad oggi utilizzati per lo più solamente per l'esame dei risultati delle sperimentazioni, offre opportunità d'impiego più ampie, legate alle tecniche di simulazione. Questo lavoro ha così lo scopo di presentare alcuni strumenti utili per la simulazione e di proporre come concreta applicazione lo studio del Dilemma del Prigioniero ripetuto. Il testo si articola nei seguenti capitoli.

Capitolo 1: vengono introdotti alcuni concetti base della simulazione socio – economica, focalizzando l'attenzione specialmente sui modelli ad agenti indipendenti; infatti solamente modelli basati su entità fortemente autonome possono consentire l'emergere di comportamenti nuovi o imprevisti.
La descrizione dello strumento di simulazione SWARM, del modello costruito basandosi su questo strumento e della struttura interna ERA concludono il capitolo.

Capitolo 2: dopo aver definito e descritto il Dilemma del Prigioniero ed averlo opportunamente messo in relazione con la teoria economica, vengono presentate le basi teoriche che sosterranno lo sviluppo degli esperimenti successivamente presentati. Verranno studiati sia gli effetti di errori casuali sia la possibilità di evolvere autonomamente nuove strategie. Sono presentate due definizioni alternative di "rumore", cioè di errori casuali che possono essere inseriti nel modello, nonché alcuni strumenti specifici utili per studiare la possibilità di evoluzione quali algoritmi genetici e classifier systems. Queste definizioni teoriche saranno la base su cui vengono costruiti gli esperimenti descritti successivamente.

Capitolo 3: la struttura dei Classifier System è approfonditamente descritta con particolare riferimento alle proprietà ed alla dinamica del sistema. In questo capitolo viene anche fornita una breve descrizione dei più importanti parametri interni del sistema; la conoscenza di questi parametri è indispensabile per poter valutare i risultati ottenuti dal modello contenente il Classifier System.

Capitolo 4: descrizione dell'esperimento originale di Axelrod da cui inizia la sperimentazione. La ristrutturazione dell'esperimento in un modello ad agenti indipendenti fornisce gli stessi risultati ottenuti di Axelrod, garantendo, in questo modo, una solida base di partenza per le sperimentazioni successive. L'introduzione nel modello dei due diversi tipi di rumore consente di concludere che, in condizioni di scarsa chiarezza comunicativa, emergono delle strategie vincitrici diverse, più adatte ad affrontare la presenza del rumore.

Capitolo 5: descrizione del modello che ingloba un Classifier System e che permette di condurre esperimenti per valutarne la capacità di evolvere strategie di gioco autonome. I risultati, in questo caso, sono contrastanti. Accanto ad una chiara capacità di trovare soluzioni in tornei più semplici, sembra mancare la capacità di elaborare soluzioni di livello strategico adatte ad un gioco come quello del Dilemma del Prigioniero.

Indice

Bibliografia

Capitolo 1

Ricerche economiche e simulazione

1.1 Introduzione

Gli incessanti progressi tecnologici compiuti nella realizzazione e costruzione di computer stanno modificato l'ambito di utilizzazione e il concetto stesso di modello economico. Oggi si ha una diffusione di capacità elaborative in grado di svolgere rapidamente grosse moli di lavoro, con un'elevata affidabilità e a bassi costi. Questa nuova situazione rende praticabile un concetto di modello economico non più vincolato ad equazioni; queste avevano, ed hanno, il grandissimo merito di consentire di descrivere e trattare problemi complessi con strumenti che utilizzano un numero relativamente piccolo di operazioni matematiche abbastanza complesse. Ora questo vincolo può essere facilmente superato e si possono anche sviluppare modelli basati su un numero grandissimo di operazioni, non necessariamente elementari.

Tre sono i settori degli studi sociali ed economici in cui la "rivoluzione informatica" si sta facendo pesantemente sentire: la possibilità di costruire modelli ad agenti indipendenti che operano autonomamente in ambienti definiti, la rappresentazione della conoscenza, la facilità di scambio e di accessibilità dei risultati delle ricerche scientifiche.

Dopo una prima fase in cui si affidavano al computer i meri calcoli e le grandi elaborazioni di dati, da alcuni anni si è in una fase diversa; si è reso possibile un forte sviluppo di ricerche nell'ambito della creazione di modelli ad agenti indipendenti il cui comportamento è simulato con l'uso di computer. Diffusione e capacità elaborativa consentono di progettare modelli in grado di simulare delle situazioni reali, senza dover descrivere analiticamente la situazione, ma semplicemente riproducendo l'ambiente in cui avvengono interazioni (anche elementari) tra componenti del sistema.

La differenza tra un modello basato su equazioni ed un altro basato sull'interazione (in qualche modo "spontanea") tra agenti semplici ed indipendenti è, in essenza, che dalle equazioni non si possono che estrarre le soluzioni (comunque

un gran risultato) insite nelle equazioni stesse. Da un modello ad agenti indipendenti è possibile che emergano comportamenti o situazioni non previste alla costruzione del modello. Costruzione che, va ripetuto, riguarda ambiente di interazione, agenti interagenti, regole (possibilmente semplici) di interazione.

Perché è importante concentrarsi su regole di interazione semplici? Più sono semplici le regole, meno sono importanti le ipotesi su cui si basano. E meno numerose sono le ipotesi, più robusti sono i risultati. I modelli più interessanti si basano su regole elementari che escludono qualsiasi altra influenza che non siano quelle in esperimento. Si cerca così, attraverso il modello, di realizzare delle condizioni sperimentali tipiche della fisica, simili a quelle ottenute da Galileo quando studiava il moto lungo un piano inclinato.

Ma mentre Galileo poteva cercare condizioni sperimentali reali in cui fosse possibile isolare il fenomeno allo studio (ad esempio, ridurre l'attrito il più possibile vicino a zero per studiare le leggi del moto) nelle scienze sociali questo è impossibile. Riuscire a costruire un modello ad agenti indipendenti per indagare, in ambiente isolato, alcuni aspetti essenziali e i cui effetti non siano immersi in innumerevoli interazioni reciproche di altro genere, può essere l'unico esperimento possibile.

L'idea, niente affatto nuova, di poter "generare conoscenza" attraverso gli elaboratori ha ritrovato nuovi impulsi negli ultimi anni. Ad esempio, le prime esperienze di rappresentazione della conoscenza, basate sul tentativo di riprodurre il funzionamento del cervello umano riproducendo collegamenti sinaptici tipici dei neuroni del cervello umano, hanno condotto alla elaborazioni delle reti neuronali. Bisogna ammettere che la riproduzione del "ragionamento" è un obiettivo ancora lontano; però si ha ora a disposizione uno strumento computazionale teoricamente saldo e sufficientemente robusto da prestarsi a diversi campi di applicazione.

E' possibile che, nel campo della rappresentazione della conoscenza, si sia in una situazione analoga ad un uomo del rinascimento alle prese con i primi studi sul volo. Se quest'uomo avesse affermato allora che il genere umano non sarebbe mai riuscito a volare con macchine di tipo leonardesco (a simulazione del volo degli uccelli) avrebbe avuto sicuramente ragione. Ma se avesse affermato che il genere umano non sarebbe mai riuscito a volare (questa volta senza qualificazioni) avrebbe avuto torto. La disponibilità elaborativa cresce con ritmi esponenziali e a grandissima velocità ed è possibile che gli studi sulla rappresentazione della conoscenza si sviluppino lungo percorsi ed arrivino a risultati assolutamente imprevedibili.

Per ora, la macchina continua ad essere uno strumento, potente e veloce, di cui è possibile valersi per tutti quei compiti che risultino, per loro natura, inadatti o sgraditi ad un intelletto umano, ansioso di evitare la ripetizione e lento nel trattare, in modo conscio, grandi quantità di informazioni.

Il lavoro di ricerca è essenzialmente attività coordinata di gruppi diversi: i risultati ottenuti da un gruppo di ricerca vengono qualificati ed arricchiti dagli apporti ottenuti nel confronto con quelli di altri studiosi, sia esperti della materia, sia impegnati in altre discipline.

La possibilità di condividere dati ed esperienze è esigenza fortemente sentita e componente importante dell'evoluzione della conoscenza e del pensiero scientifico. L'elettronica ha permesso, oltre a quanto visto prima, l'enorme potenziamento dei

mezzi di comunicazione rendendo possibile la condivisione, in tempi sempre più brevi, di informazioni fra gruppi di lavoro sparsi per tutto il mondo: la rete Internet, molto utilizzata in ambito universitario e scientifico, collega quotidianamente studiosi siti in località remote e, soprattutto, permette di mantenere in linea un magazzino di informazioni dalle proporzioni mai conosciute in precedenza.

Per i modelli al computer si amplifica la possibilità di condividere i risultati delle proprie ricerche: oltre ai risultati è possibile trasmettere l'oggetto stesso dell'osservazione, l'intero modello, ossia l'insieme di programmi che producono la simulazione del sistema reale, oggetto di studio. Ciò equivale alla possibilità di condividere, per via Internet, l'intero esperimento e non solamente le deduzioni a cui si è giunti.

In questo modo altri studiosi possono rieseguire la simulazione, migliorare il modello, apportare le variazioni suggerite dalla loro esperienza personale. Ma soprattutto si garantisce un elemento essenziale per considerare scientifica un'attività: la riproducibiltà dei risultati.

Quando l'attività sperimentale dovesse estendersi ad un elevato numero di casi, o potesse basarsi su molteplici metodologie, più esperti potrebbero collaborare alla sua conduzione, apportando ciascuno, il contributo relativo alle parti in cui più forti fossero le loro conoscenze. Si apre, perciò, una opportunità di collaborazione attiva, già a partire dalla fase di predisposizione del modello normalmente superiore a quella offerta dalle metodologie tradizionali.

Come esempio dell'enorme potenzialità di simili collaborazioni si consideri lo sviluppo che ha avuto negli ultimi tempi il cosiddetto "freeware", cioè il software gratuitamente disponibile sulla rete; per solito è un software immesso in rete da studiosi operanti in tutto il mondo, con lo scopo di aumentarne l'utilizzazione e, di conseguenza, migliorarne lo sviluppo. Recenti studi (Meo 1998), relativi ad un programma di ricerca del Politecnico di Torino, stimano il valore di questo software come circa 1/6 del valore complessivo totale.

1.2 Evoluzione dell'applicazione del calcolo automatico all'economia

Nell'ambito dell'indagine economica, l'importanza del calcolo automatico è tradizionalmente riconosciuta in riferimento alle analisi econometriche e statistiche; più di recente, però, si è andata affermando la sua utilizzazione diretta nello studio dei fenomeni economici, come dimostrano i buoni risultati ottenuti da un nutrito numero di lavori.

Per chiarire ulteriormente quali differenze esistano tra l'utilizzo tradizionale dell'informatica e la costruzione di modelli è indispensabile richiamare la definizione di sistemi lineari e non lineari. Una possibile definizione di sistema lineare è quella che definisce il comportamento dell'intero sistema come somma dei comportamenti delle sue parti, mentre il comportamento dei sistemi non lineari non è semplicemente riconducibile a quello delle parti componenti.

E' evidente che il valere del principio di sovrapposizione nei sistemi lineari consente di scindere un sistema complicato in parti costitutive più semplici,

analizzare ciascuna componente separatamente e, infine, "ricostruire" il comportamento del sistema riaggregando le componenti. Questa è la caratteristica più importante dei sistemi lineari: studiando le parti separatamente, si può analizzare il sistema completo in ogni suo dettaglio.

Non si ha questa possibilità di analisi quando si studia un sistema non lineare: anche se si analizzano tutte le parti costitutive più semplici e si perviene ad una comprensione completa ed esauriente del loro funzionamento, non si è in grado di comprendere il sistema nel suo complesso. Il sistema è diverso dalla somma delle parti. I comportamenti dei sistemi complessi non lineari dipendono dalla interazione (sovente a carattere retroattivo) tra le parti e non tanto (o non solo) dalle caratteristiche delle parti stesse; queste proprietà vengono immediatamente meno quando le singole componenti vengono studiate separatamente.

La recente, grande disponibilità di mezzi informatici, rende possibile una metodologia di studio diversa, di sintesi piuttosto che di analisi: si affronta il sistema complesso non dall'alto per scomporlo, ma dal basso per ricomporlo.

Lo scopo è quindi di arrivare a far emergere comportamenti complessi nel sistema, partendo da numerose interazioni semplici: la complessità non è infatti nelle parti ma nelle loro relazioni. Da dove nasce allora la difficoltà nel trattare questi sistemi? Dal fatto che un sistema sociale, anche semplice, ha un numero di stati possibili (dovuto alla combinazione degli stati tra gli agenti) elevatissimo, tanto grande da non consentirne una trattazione di tipo enumerativo. Per fornire regole "globali" non si possono semplicemente considerare tutti i possibili stati e fornire una regola differente per ciascuno di essi: è necessario un approccio diverso.

Sottostante alle considerazioni prima viste, vi è anche la convinzione che buona parte (se non tutti) i comportamenti di notevole complessità che si possano osservare in natura o in sistemi sociali, siano fondati sull'applicazione di regole semplici attraverso interazioni complesse. Hofstadter (1988) cita l'esempio di formiche guerriere che anche se seguono comportamenti individuali semplici (quali possono essere le risposte agli stimoli olfattivi – ormonali) riescono a mettere in atto un comportamento complesso come quello di costruire un "ponte vivente" per consentire il superamento di ostacoli altrimenti non superabili.

Come si è visto, è molto difficile risalire da un comportamento complesso a che cosa l'ha generato: può essere parecchio più semplice creare un modello, che basandosi su comportamenti semplici presenti caratteristiche "globali" simili a quelle che si vogliono studiare. Caratteristiche che *emergono* dall'interazione di singoli agenti.

1.3 I modelli di simulazione basati su agenti

Vi sono diversi tipi di impieghi della simulazione fatta con l'uso di computer e non tutte hanno prioritariamente l'obiettivo di simulare il comportamento di popolazioni di agenti.

Previsione: l’obiettivo è, appunto, prevedere quali potranno essere gli esiti di interventi di politica economica o fiscale o anche modifiche istituzionali o, talvolta, dare indicazioni di massima sul prevedibile andamento di alcune variabili macroeconomiche di base. Questo tipo di simulazione è però quasi sempre basato su modelli definiti “di struttura”, indagano fenomeni molto complessi e si basano su dati reali raccolti con tecniche statistiche. In queste simulazioni quindi, poiché l’analisi è centrata sulle transazioni tra stati diversi di un sistema e non sul funzionamento del sistema stesso, non vi è necessità di ipotizzare il comportamento dei singoli agenti che vi operano, mentre è essenziale la precisione dei risultati ottenuti.

Sperimentazione: lo scopo in questo caso è di riprodurre una condizione sperimentale ben controllata che consenta di ricreare, in laboratorio, fenomeni della realtà sociale o economica che si vogliono studiare. In questo caso, si può dire, con una sorta di slogan, che *si vuole capire attraverso il riprodurre*. La precisione dei risultati perde importanza, ne acquista la capacità di spiegazione dei fenomeni osservati. La duttilità dello strumento consente di andare ben oltre la mera verifica di un processo interpretativo poiché consegna allo sperimentatore possibilità che altrimenti non esisterebbero.

- Si possono cambiare i valori delle variabili che definiscono il comportamento del modello, rendendo molto più semplice il ragionamento di tipo “what if”. Questa possibilità è ovviamente negata quando il sistema in osservazione è un reale sistema economico, ma diventa elemento essenziale per ottenere una specie di “laboratorio sociale”.
- Si possono collegare eventi distanti nel tempo o nello spazio e creare condizioni non facilmente ottenibili, osservare catene di eventi interessanti che portano a condizioni molto negative e che quindi, nella realtà, si fa di tutto per evitare.
- Inoltre, la combinazione di modelli e di tecniche esplorative (quali le reti neurali, gli algoritmi genetici e i classifier system che vedremo più in dettaglio nel prossimo capitolo) permettono di scandagliare, esplorare lo “spazio degli stati del sistema” e di trovare soluzioni o condizioni altrimenti ben difficilmente determinabili.

I modelli comportamentali hanno quindi come oggetto primario d'indagine i meccanismi attraverso i quali l'interazione dell'operato dei singoli agenti può generare conseguenze aggregate complesse, quali il sorgere e trasformarsi di istituzioni o la convergenza dei comportamenti verso direttrici comuni. Questa impostazione concretizza un enorme cambiamento nell'ambito delle scienze economiche: Tesfatsion (1998) mette in evidenza il fatto che ciò implichi l'adozione di un metodo sperimentale che porti a concepire il fenomeno economico come riproducibile in un qualche tipo di laboratorio, distaccandosi dalla concezione classica che considerava la storia l'unica possibile fonte di dati per le scienze sociali. L'economia sperimentale ha quindi la necessità di sviluppare uno stile di ricerca del tutto diverso: non più solamente analizzare per capire ma anche riprodurre per comprendere.

Importanti lavori, in questa direzione, riguardano lo sviluppo di mondi artificiali dove strutture sociali e comportamenti di gruppo emergono dall'interazione degli individui, rappresentati da programmi, operanti essi stessi nell'ambiente

artificiale, in base a regole autonomamente sviluppate. Promuovere lo sviluppo, spontaneo, di istituzioni e strutture in un mondo artificiale, ha lo scopo di indagare quali sono le condizioni essenziali che portano alla formazione delle medesime e, di conseguenza, chiarire quali meccanismi, a livello individuale necessita ipotizzare affinché si generino i comportamenti collettivi oggetto dell'esperimento. In campo economico, l'utilizzazione di queste metodologie è adottata dalla "*Agent based computational economics*" (ACE), di cui si ha una definizione in Tesfatsion (1997):

> Agent-based computational economics (ACE) is roughly characterized as the computational study of economies modelled as evolving decentralized systems of autonomous interacting agents. A central concern of ACE reasearchers is to understand the apparently spontaneous appearence of global regularities in economic processes, such as the unplanned coordination of trade

Questo ambito di ricerca (la ACE) collega, direttamente e senza ricorrere a costruzioni teoriche artificiose, i comportamenti micro e gli aspetti macroeconomici indagando come la somma di interazione semplici di singoli individui sia capace di generare nel sistema considerato nella sua interezza, comportamenti complessi. La posizione di ricerca è chiaramente antiriduzionista poiché manifestamente si ipotizza l'impossibilità di ricondurre la spiegazione delle proprietà di un sistema alla conoscenza delle caratteristiche dei suoi componenti elementari e l'importanza della dinamica del sistema stesso, giustificando l'interesse per le leggi che ne descrivono l'evoluzione nel tempo.

Il tipo di modello ad agenti indipendenti è essenziale Terna (1998):

> (…) per sperimentare la non linearità degli effetti aggregati dei loro comportamenti (cioè: il tutto che non necessariamente corrisponde alla somma delle parti) in presenza di interazione, con l'emergenza della complessità: è in ciò che consiste l'originalità della sperimentazione artificiale con modelli di simulazione fondati su agenti.

In conclusione si può dire che i modelli basati su agenti risultano particolarmente adatti a studiare "in laboratorio" sistemi complessi; è la dinamica delle autonome azioni individuali che diviene l'elemento per inferire il comportamento dell'aggregato. Il ricercatore non deve affatto sforzarsi di prevedere tutte le azioni possibili con quindi un rischio decisamente inferiore di influire sui risultati del modello attraverso la codificazione del medesimo.

Un ulteriore beneficio dei modelli ad agenti indipendenti è la possibilità di "complicare" il modello in modo incrementale: un tipico sviluppo del modello potrebbe essere descritto con i seguenti passi.

- eseguire, inizialmente, una versione minimale del modello
- usare i primi risultati per definire le linee di sviluppo del modello stesso
- modificare le capacità degli agenti o le caratteristiche dell'ambiente
- rielaborare una seconda versione del modello.

Questo metodo di definizione del modello per passi reiterati permette di procedere a verificazione continua del medesimo e della teoria su cui si basa, e garantisce di operare ogni aggiunta su una base solida e funzionante. Consente, insomma, di costruire il modello mediante un processo quanto mai "naturale" e senza separazione netta tra analisi, progetto e costruzione.

Va anche messa in evidenza l'"economicità" di questo modo di procedere, che deriva dal poter agevolmente riutilizzare parti del medesimo per la conduzione di altre sperimentazioni. Se i comportamenti degli agenti sono semplici ed i risultati derivano dalla interazione dei singoli individui è evidente che è possibile sia modificare efficacemente il modello sia riutilizzarne molte parti.

Inoltre diventa tecnicamente molto più semplice innestare nei modelli l'utilizzo di programmi "intelligenti". Questo inserimento genera dei modelli in cui l'interazione fra gli operatori non deve essere prevista a priori, ma che può scaturire dalla elaborazione autonoma di regole comportamentali da parte dei singoli, impegnati nel tentativo di migliorare i risultati ottenuti dall'interagire con l'ambiente.

Questo metodo di costruzione dei modelli richiede però strumenti adatti: non tutti gli ambienti di programmazione si prestano ad un simile utilizzo. La descrizione di uno dei software disponibili e adatti allo scopo è argomento del prossimo paragrafo.

1.4 SWARM

SWARM è un software package, vale a dire un insieme di programmi e di librerie specifiche originariamente sviluppato presso il Santa Fe Institute, per effettuare simulazioni mediante sistemi multi-agente in ambienti complessi. L'obiettivo di SWARM è di essere un utile "attrezzo" per ricercatori di varie discipline. La struttura di base è la simulazione di collezioni (sciami, swarm in inglese) di agenti interagenti tutti insieme in modo da poter essere usato per costruire una grande varietà di modelli diversi. Per consentire un maturo sviluppo della ricerche condotte mediante utilizzo di modello sviluppati a computer, c'è bisogno di un certo numero di strumenti e di software standard, ben progettati e che possano essere usati in molte discipline. La presentazione rintracciabile sul sito di SWARM è la seguente:

> Swarm is a multi-agent software platform for the simulation of complex adaptive systems. In the Swarm system the basic unit of simulation is the swarm, a collection of agents executing a schedule of actions. Swarm supports hierarchical modeling approaches whereby agents can be composed of swarms of other agents in nested structures. Swarm provides object oriented libraries of reusable components for building models and analyzing, displaying, and controlling experiments on those models. Swarm is currently available as a beta version in full, free source code form. It requires the GNU C Compiler, Unix, and X Windows. More information about Swarm can be obtained from our web pages, *http://www.santafe.edu/projects/swarm/*.

Come si evince dalla precedente citazione, il modello adottato da SWARM è quello di agenti indipendenti che possono però anche essere riuniti in gerarchie, nel senso che gli agenti di un livello del modello possono essere rappresentati a loro volta da sciami di agenti indipendenti. Non ci sono specifici requisiti di dominio quali possono essere quelli spaziali, di schemi di interazione o fisici. L'unità di base

di simulazione è semplicemente l'agente (o uno sciame di agenti), che può generare eventi che influenzano se stesso, gli altri agenti o l'ambiente in cui operano: il modello di simulazione è perciò null'altro che le interazioni tra gli agenti.

Perché mettere in atto tanti sforzi per ottenere la diffusione di standard per il software e gli ambienti in cui condurre gli esperimenti? Sempre dal sito di SWARM arriva la risposta:

> In the sciences, especially in the study of complex systems, computer programs have come to play an important role as scientific equipment. Computer simulations --- experimental devices built in software --- have taken a place as a companion to physical experimental devices. Computer models provide many advantages over traditional experimental methods, but also have several problems. In particular, the actual process of writing software is a complicated technical task with much room for error.
> Early in the development of a scientific field scientists typically construct their own experimental equipment: grinding their own lenses, wiring-up their own particle detectors, even building their own computers. Researchers in new fields have to be adept engineers, machinists, and electricians in addition to being scientists. Once a field begins to mature, collaborations between scientists and engineers lead to the development of standardized, reliable equipment (e.g., commercially produced microscopes or centrifuges), thereby allowing scientists to focus on research rather than on tool building. The use of standardized scientific apparatus is not only a convenience: it allows one to ``divide through'' by the common equipment, thereby aiding the production of repeatable, comparable research results.
> In complexity research, at the Santa Fe Institute and elsewhere, we rely heavily on computers in the course of our investigations. We spend a lot of time constructing our own experimental apparatus in software, the computational equivalent to blowing our own glassware. Unfortunately, computer modeling frequently turns good scientists into bad programmers. Most scientists are not trained as software engineers. As a consequence, many home-grown computational experimental tools are (from a software engineering perspective) poorly designed. The results gained from the use of such tools can be difficult to compare with other research data and difficult for others to reproduce because of the quirks and unknown design decisions in the specific software apparatus. Furthermore, writing software is typically not a good use of a highly specialized scientist's time. In many cases, the same functional capacities are being rebuilt time and time again by different research groups, a tremendous duplication of effort.
> A subtler problem with custom-built computer models is that the final software tends to be very specific, a dense tangle of code that is understandable only to the people who wrote it. Typical simulation software contains a large number of implicit assumptions, accidents of the way the particular code was written that have nothing to do with the actual model. And with only low-level source code it is very difficult to understand the high-level design and essential components of the model itself. Such software is useful to the people who built it, but makes it difficult for other scientists to evaluate and reproduce results.
> In order for computer modeling to mature there is a need for a standardized set of well-engineered software tools usable on a wide variety of systems. The Swarm project aims to produce such tools through a collaboration between scientists and software engineers. Swarm is an efficient, reliable, reusable software apparatus for experimentation. If successful, Swarm will help scientists focus on research rather than on tool building by giving them a standardized suite of software tools that provide a well equipped software laboratory.

Prima di passare ad esaminare la struttura di un modello sviluppato con SWARM, occorre mettere in evidenza alcuni elementi essenziali di questo strumento.

Tempo. Sicuramente intercorre una grande differenza fra lo scorrere del tempo in un esperimento condotto nel mondo reale ed in uno simulato al computer. Nella realtà il tempo passa senza bisogno di interventi e gli agenti si sincronizzano su di esso in base alla misurazione del medesimo; in un mondo artificiale occorre disporre di una componente deputata a gestire lo scorrere del tempo, che si preoccupi anche di mantenere il sincronismo fra le varie altre componenti. In compenso si può accelerare o rallentare a piacimento lo scorrere del tempo nel modello; occorre, però, esplicitare in modo completo le assunzioni ipotizzate nel modello per quanto riguarda il tempo, onde permettere ad altri studiosi di rieseguire l'esperimento per riprodurne i risultati.

Eventi che accadono a caso. L'accadere di eventi casuali viene simulata facendo dipendere questi eventi dall'estrazione di un numero generato, di volta in volta, in modo pseudo-casuale. L'ottenimento di questi numeri è basato su apposite procedure, piuttosto complicate, che permettono anche di scegliere fra i diversi tipi di distribuzione di probabilità, relative alla generazione di ciascuno dei valori contenuti nell'intervallo voluto. Usualmente questi algoritmi si basano su di un valore iniziale, detto "seme"; quando si voglia esaminare uno scenario in cui gli eventi casuali siano diversi, il seme dovrà essere variato. Anche in questo caso, la necessità di predisporre la casualità (che nel mondo reale si manifesta da sola) è compensata da un solido vantaggio: nell'osservazione del mondo reale il caso viene "subito" e, molte volte si concretizza in un inquinamento dei dati che genera il bisogno di sottoporre i medesimi a lunghe elaborazioni statistiche; nei mondi artificiali il caso viene "gestito", al pari delle altre variabili del problema, con ciò permettendone l'utilizzazione più idonea.

Linguaggio di programmazione. SWARM implementa la struttura logica degli sciami di agenti utilizzando, in maniera diretta, l'Objective C che è un linguaggio orientato agli oggetti (Object Orieted, OO).

Nella programmazione OO il software serve essenzialmente per:

- generare le varie classi di oggetti
- generare gli oggetti, ovvero le istanze di ciascuna classe
- generare le variabili tipiche della classe o degli oggetti
- scrivere i metodi relativi alle classi generate (un metodo può essere pensato come un programma che elabora certe azioni agendo esclusivamente sulle variabili di un oggetto e che attua il comportamento degli oggetti).

Ogni oggetto "possiede" le proprie variabili di istanza, che rappresentano, in qualche modo, lo "stato" di quella istanza; ogni oggetto può modificare il contenuto delle proprie variabili, ma solamente delle proprie.

Scambio di messaggi. Tutti gli oggetti, e specialmente gli agenti indipendenti che sono la struttura portante del modello, devono scambiarsi messaggi. Questo per ottenere informazioni da altri, per modificarne lo stato, per, insomma, interagire. La

funzione di scambio messaggi è tipica dell'ambiente OO e consente di creare strutture particolarmente semplici da modificare. La sintassi di un messaggio è la seguente:

[oggettoTarget messaggio Argomento1: var1 Argomento2: var2]

Dove *oggettoTarget* è l'oggetto destinato a ricevere il messaggio, *messaggio Argomento1: Argomento2:* è il messaggio che viene inviato a quell'oggetto, *var1* e *var2* sono gli argomenti, ovvero dei valori che vengono passati attraverso il messaggio stesso. Gli argomenti passati sono identificati sia da una parola chiave sia dalla posizione nel messaggio. Il messaggio, in realtà, altro non è che il relativo metodo definito nella classe a cui appartiene l'oggetto a cui viene inviato il messaggio. L'oggetto che invia il messaggio ottiene, usualmente ma non necessariamente, una "risposta". Ad esempio, nel modello presentato nei capitoli successivi, viene richiesto a un istanza della classe Club di cercare nel mondo se c'è, nei pressi dell'agente in quel momento attivo, un altro agente con cui iniziare un ciclo di iterazioni del dilemma del prigioniero. Questo si ottiene appunto inviando un messaggio a club così codificato:

opponent = [club getOpponentFar: distance fromX: posX y: posY forNumber: number];

in cui

- club è l'oggetto a cui è inviato il messaggio
- getOpponentFar: fromX: y: forNumber: è il messaggio vero e proprio
- distance, posX, posY, number sono 4 parametri passati
- opponent= indica che il messaggio avrà risposta e che questa sarà memorizzata nella variabile opponent

Le comunicazioni fra gli individui del mondo artificiale avvengono quindi tramite scambi di messaggi, ovvero per interazioni fra i programmi che li rappresentano.

SWARM predispone tutte le funzioni appena viste in modo standard; si ha quindi a disposizione tutto il corredo strumentale descritto. Questo *software* "di base", inoltre:

- possiede caratteristiche di robustezza, sufficienti a permettere la ripetizione ciclica dell'elaborazione per un gran numero di volte
- consente l'immissione di parametri, onde rendere agevole l'esecuzione dell'esperimento con presupposti diversi
- ha funzioni di "sonda" che rende molto semplice controllare il valore delle variabili in ogni momento.

Swarm è totalmente gratuito; esso fa parte di quel patrimonio di *freeware* cui si è accennato nel primo paragrafo. *Swarm* permette di concentrare l'attenzione sulla codificazione dell'esperimento ad un alto livello di astrazione, sollevando l'utilizzatore dalla gran parte degli oneri di programmazione delle componenti infrastrutturali, non direttamente connesse con la specifica realizzazione.

L'inserimento di larghe parti del codice in librerie comuni consente di pervenire allo sviluppo di modelli maggiormente uniformi, più adatti a favorire la collaborazione fra studiosi e, soprattutto, più facilmente comprensibili. Questa caratteristica risulta utile nella divulgazione, non solo dei modelli, ma anche dei risultati conseguiti.

Infine, e da non trascurare, il mondo *Swarm* conta diverse centinaia di migliaia di utilizzatori, che possono interagire fra loro, e con gli esperti del prodotto, grazie ad una *mailing-list* gestita direttamente dal laboratorio che ha sviluppato il prodotto e distribuita via Internet; in questo modo vengono divulgate sia le questioni relative all'utilizzo del prodotto, sia gli annunci dei nuovi rilasci. Il software è fornito sotto i termini della licenza "GNU", ovvero è consentita la modificazione, la duplicazione e la divulgazione, con l'unico obbligo di non apporre termini maggiormente restrittivi, nei confronti di ulteriori fruitori. *Swarm* può essere installato su svariate piattaforme, fra cui *MS/Windows 95/98/NT, LINUX, SOLARIS* ed altri ancora.

1.5 Struttura di un modello sviluppato in SWARM

Nel contesto stabilito da SWARM, la procedure che porta alla costruzione del modello di base per un esperimento potrebbe seguire, a grandi linee, il seguente schema:

1. creare un mondo artificiale con spazio, tempo e oggetti che vengono collocati (a caso o seguendo certi criteri) in questa struttura generale; far si che questi oggetti definiscano da soli il proprio comportamento seguendo le loro proprie regole e i propri stati interni; di tanto in tanto controllare lo stato del mondo

2. creare un certo numero di oggetti che servono per osservare, registrare e analizzare i dati prodotti dalle interazioni degli agenti di cui al punto 1

3. far “girare” il modello, attivando sia gli agenti di simulazione che gli oggetti creati per l’osservazione in accordo ad alcune esplicite scelte di simultaneità

4. interagire con l’esperimento in corso attraverso i dati prodotti dagli oggetti creati per l’osservazione, in modo da ottenere una serie di dati sperimentali dal modello stesso

5. a seconda dei risultati al punto 4, modificare l’apparato sperimentale ed, eventualmente, ritornare al punto 3

6. nella pubblicazione dei risultati, fornire tutti i dettagli necessari per permettere ad altri ricercatori di riprodurre i risultati ottenuti.

L'esperimento in Swarm è dunque caratterizzato da una struttura gerarchica su due livelli: quello dell'osservatore (observer), più esterno, e quello del modello

vero e proprio (model); ciascuno dei due contiene oggetti, elenchi di azioni ed orari. Qui di seguito è presentata una breve descrizione della struttura ricavata dal sito SWARM stesso.

Structure of a Swarm Simulation

The core of a Swarm simulation is the modeled world itself. In the simplest case, a model consists of one swarm inhabited by a group of agents and a schedule of activity for those agents. The agents themselves are implemented as objects. Agents are created by taking a class from the Swarm libraries, specializing it for the particular modeling domain, and then instantiating it, one object per agent. For example, an agent that is a neural network could start by taking a general purpose neural network class from the `neuro` library, adding extra methods needed for the specific type of network, and then creating an instance of it to be the actual neural network.

In modeling it is common (but not universal) to speak of agents as living in an environment. Many simulation platforms fix the environment as a particular type; two dimensional grids are common. A distinguishing feature of Swarm is that there is no design requirement for a particular kind of environment. For instance, a coyote/rabbit model might have the rabbits living in the environment of a garden. In Swarm, this environment is itself just another agent. The garden is simply an instance of a user-defined garden agent, perhaps based on a cellular automata to simulate growth of carrots. The garden agent might have a special status in the model, but in the underlying software it is treated no differently than any other agent. In the general case, the environment for the agents is the agents themselves: some agents might have a larger influence than others, but in Swarm they are considered fundamentally equivalent.

Once a user has defined the agents and established their relationships, the last step in building the model itself is to put the agents together into a swarm. The user writes a schedule of activity for the agents, defining how time is simulated in the system by creating a set of actions in a specified ordering. Schedules are built by creating instances of data structures from the `activity` library, filling them in with ordered object/message pairs. Once the schedule is completed the model swarm is ready to be executed.

A model running by itself is not very interesting: data collection tools are needed to observe the model and record what is happening. In Swarm measurement happens by the actions of observer agents, special objects whose purpose it is to observe other objects via the probe interface For example, one observer agent might be watching the number of rabbits and producing a time series graph of population dynamics. Another observer agent could track the spatial distribution of the coyotes, storing data to a file for later analysis.

The observer agents themselves are a swarm, a group of agents and a schedule of activity. By combining this swarm with a model swarm running as a subswarm of the observer, a full experimental apparatus is created. By using hierarchical swarms to separate data collection from the model, the model itself remains pure and self-contained, a simulated world under glass. Different observer swarms can be used to implement different data collection and experimental control protocols, but the model itself remains unchanged.

La fase di creazione degli oggetti rispetta la gerarchia evidenziata: il main, crea l'observer; questi crea, a sua volta: il model, gli oggetti grafici (per visualizzare i dati rilevati), una mappa o "finestra" (per l'immissione dei parametri) ed una serie di "pulsanti", tramite i quali l'utilizzatore può intraprendere azioni sul modello. Il model crea: la propria mappa per l'immissione dei parametri, gli agenti e gli eventuali oggetti di servizio.

Terminata la creazione degli oggetti, il main ordina all'observer di creare le "azioni", e lo stesso ordine è impartito, dall'observer, al model. Vengono quindi create due tabelle corredate degli opportuni orari, scanditi dal proprio gestore del tempo. Il sincronismo fra model ed observer è automaticamente gestito da componenti del pacchetto Swarm.

Da ultimo, il main ordina all'observer di operare; questi trasmette l'ordine al model che comincia ad eseguire le azioni contenute nella propria tabella, inviando gli ordini (ovvero i messaggi) agli agenti ed agli altri oggetti che popolano il modello. Le azioni vengono eseguite ciclicamente dall'observer e dal model. Mediante i pulsanti di “stop”, “start”, “next” e “quit” si può controllare l’esecuzione del modello. I grafici, eventualmente previsti, vengono aggiornati dall'observer durante l'elaborazione. La posizione degli agenti nel mondo artificiale può essere visualizzata in una apposita finestra grafica, dove ogni agente è rappresentato da un punto; agendo, con il pulsante destro del mouse, su uno di questi punti è possibile accedere alle variabili interne dell'agente mediante la funzione di “sonda”. Una volta selezionato, nel modo descritto, un agente, tenendo premuto il pulsante del mouse e spostando il medesimo su una finestra contenente un grafico, i dati dell'agente vengono automaticamente visualizzati anche sul grafico.

Nonostante le notevoli possibilità, il relativo onore di programmazione è ridotto veramente al minimo: le variazioni da apportare al codice di main, observer e model, sono poche e per la gran parte delle realizzazioni, la loro struttura può essere mutuata direttamente dagli esempi forniti col pacchetto. Ovviamente le necessità di intervento sul modello vero e proprio è maggiore ma risulta, comunque, abbastanza contenuta.

1.6 La struttura interna dei modelli: lo schema ERA

L'adozione di *Swarm* e la scelta, per altro conseguente, di *Objective-C*, come linguaggio di programmazione, permettono di ottenere gli sperati vantaggi in termini di possibilità di spiegare con chiarezza i modelli elaborati ed i risultati ottenuti. Non viene però stabilito nessuno standard per la parte che costituisce il modello vero e proprio.

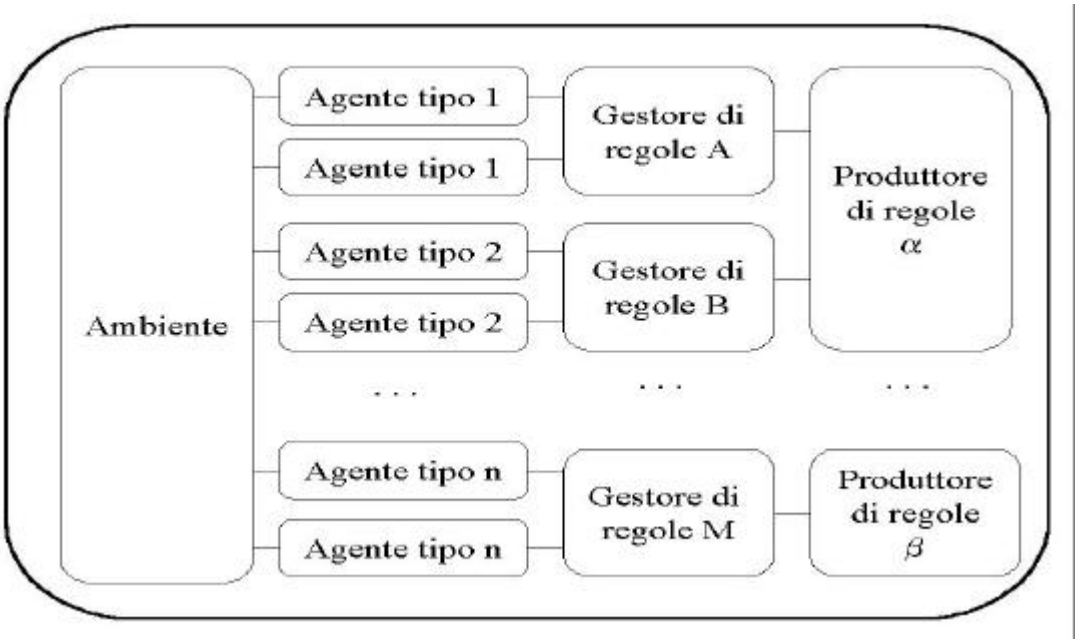
In Terna (1998) si trova una proposta volta a uniformare la struttura dei modelli basati su agenti, in particolare quelli sviluppati in SWARM. Nel testo viene suggerita l'adozione di uno schema denominato *Environment-Rule-Agent* (ERA): schema che si propone di agevolare sia l'utilizzazione del modello da parte di altri ricercatori, sia la semplificazione di modificazioni successive.

Lo schema, riprodotto in figura prevede una netta separazione fra l'ambiente e gli agenti che, per la gestione delle proprie regole di comportamento, si avvalgono dei servizi forniti da oggetti specializzati, appartenenti ai due livelli successivi.

Gli agenti non comunicano direttamente fra loro ma per il tramite dell'ambiente che, ad esempio, potrebbe fornire a ciascuno le informazioni sufficienti a formare una lista dei propri vicini o corrispondenti.

Il gestore di regole si occupa di selezionare la norma di comportamento più adatta alla specifica situazione; questa viene descritta dalle informazioni, relative all'ambiente, comunicate dall'agente fruitore dei servizi del gestore di regole.

Il produttore di regole è incaricato della componente di ricerca di nuove regole, possibilmente migliori, in modo da realizzare una sorta di "apprendimento".



Questa architettura iniziale può essere ulteriormente sviluppata poiché è possibile inserire altri elementi, posizionandoli, ad esempio, fra il livello dell'agente e quello del gestore di regole. Il loro compito è facilitare le comunicazioni fra agenti e gestori di regole o, nel caso di attività particolari che non possono essere demandate direttamente all'ambiente, il loro compito può essere di organizzare l'incontro tra agenti.

Nel modello che verrà dettagliatamente illustrato nel cap.4 e che si basa su questa struttura, sono infatti previsti altri due oggetti, diversi dagli agenti:

- un gestore di incontri (*club*) che ha appunto il compito di organizzare l'incontro tra due agenti, raccogliere le loro decisioni, assegnare il punteggio relativo,
- ed un interprete (*interface*) capace di tradurre dati espressi nella metrica e nomenclatura propria dell'agente in altri, aderenti alla metrica e denominazione utilizzata dal gestore di regole.

Questa struttura (ERA) introduce, come abbiamo visto, una precisa diversificazione degli oggetti che consente di ottenere alcuni vantaggi:

- semplifica il lavoro di individuare dove posizionare le singole componenti; si può così procedere sulla base di un robusto impianto strutturale, evitando la redazione di codici confusi e di difficile comprensione e gestione

- permette di specializzare gli oggetti del modello; se ogni oggetto svolge una determinata funzione: si evitano duplicazioni e, nuovamente, il codice risulta più comprensibile e la sua manutenzione più agevole
- ammette anche l'utilizzazione di componenti diverse, confezionate in precedenza o da altri ricercatori. Il fatto che ciascuna componente abbia una sua funzione ben definita, permette di sostituire una sola componente (ad esempio il gestore di regole) praticamente senza variazioni nel resto del modello. In questo modo sono facilitati gli scambi di esperienze: se si volesse sperimentare la metodologia di gestione delle regole utilizzata da un altro studioso, in mancanza di una così precisa impostazione, si dovrebbe, probabilmente, intervenire su gran parte del codice; se anche questi adottasse lo schema ERA, invece, l'intervento comporterebbe poco più della ricompilazione del programma.

Capitolo 2

Il dilemma del prigioniero

2.1 Definizione e inquadramento nella teoria dei giochi

La teoria dei giochi studia i processi decisionali in situazioni in cui il risultato finale è prodotto dalle azioni di numerosi agenti indipendenti; usualmente, come accade nella maggior parte dei modelli che descrivono situazioni di tipo economico, la decisione del singolo agente dipende dalle decisioni di alcuni o tutti gli altri. Si dice allora che il gioco descritto è di tipo *interattivo.* In queste situazioni, ogni agente cerca costantemente di prevedere le probabili mosse degli altri, in modo da elaborare la migliore risposta o "contromossa" possibile. Il compito della teoria dei giochi è appunto quello di elaborare ed offrire uno schema concettuale per la corretta definizione di tale risposta.

2.1.1 Alcune definizioni importanti

Nella teoria dei giochi si distingue tra giochi *cooperativi* e giochi *non-cooperativi.* La distinzione è spesso imprecisa, però possiamo dire che:

- nei primi, i giocatori stringono accordi vincolanti prima e durante lo svolgimento del gioco e la comunicazione tra giocatori è permessa
- nei secondi, non possono esistere accordi vincolanti e la comunicazione tra giocatori può essere o non essere permessa.

L'imprecisione deriva, in realtà, non dalla definizione, ma dal fatto che alcune situazioni, definite come "cooperative" nel linguaggio comune, sono invece definite "non cooperative" nel linguaggio della teoria. I giochi non cooperativi sono perciò la descrizione di situazioni in cui gli individui scelgono un comportamento cooperativo solamente perché tale atteggiamento riflette l'interesse del singolo individuo (tipicamente perché l'individuo teme la ritorsione degli altri qualora la sua cooperazione si interrompesse). In campo economico la non – cooperazione è la regola e la cooperazione l'eccezione. D'altro canto solamente postulando una situazione non–cooperativa si rende interessante lo studio delle condizioni necessarie

affinché coordinamento e cooperazione sorgano spontaneamente tra giocatori "economici".

Un *single shot game* è un tipo di gioco che viene effettuato una volta sola e in cui l'esperienza passata viene perduta ogni volta che il gioco viene nuovamente iniziato; viceversa in un *gioco ripetuto* tutta o parte della storia passata del gioco viene a far parte delle informazioni a disposizione dei giocatori che sono perciò in grado di ricordare le mosse e le strategie più usate, i risultati migliori e le sconfitte più nette. Se il gioco viene ripetuto all'infinito si parla di *supergame.*

Il modello elaborato e qui presentato simula un caso di gioco ripetuto non–cooperativo; un'interazione tra 63 giocatori che giocano il Dilemma del Prigioniero.

Le informazioni sono indispensabili per giocare, qualunque sia la forma del gioco, single shot o ripetuto, cooperativo o non. Per vincere al gioco bisogna perciò risolvere due problemi: acquisire le informazioni rilevanti ed utilizzarle nel migliore dei modi.

La teoria dei giochi nella sua veste classica postula che questi due problemi non esistano: tutti gli agenti condividono le informazioni rilevanti e le utilizzano nel migliore dei modi.

La teoria dei giochi si occupa, come detto prima, di situazioni di tipo interattivo strategico. Nel linguaggio comune si confonde spesso l'interazione tra soggetti con il loro comportamento strategico, ma è importante evitare una simile confusione.

Un gioco è interattivo quando il risultato finale dipende dalle decisioni di ciascun giocatore, decisioni che il singolo prende tenendo conto del comportamento di tutti (o parte) degli altri. Interazione vuol dire quindi che le decisioni sono legate tra di loro in qualche modo.

Un singolo agente si comporta in modo strategico (vale a dire che l'interazione è di tipo strategico) quando sfrutta la conoscenza delle mosse altrui per formulare le proprie risposte, ma anche quando sfrutta le proprie mosse per controllare le azioni altrui. Ogni individuo, in un gioco strategico, pianifica il proprio comportamento globale (quello che seguirà dall'inizio alla fine del gioco) cercando di prevedere, utilizzare e indurre i comportamenti degli altri giocatori.

Il problema è che i giochi studiati dalla teoria sono caratterizzati dalla presenza contemporanea di numerosi "strateghi", ognuno dei quali cerca di controllare gli altri.

Il comportamento strategico dei giocatori può anche essere differenziato in semplice e difficile. Quello semplice si ha nei giochi ad interazione sequenziale, quando i giocatori decidono a turno (come nel gioco degli scacchi) e dunque conoscono la mossa immediatamente precedente del loro avversario; il loro set di informazioni è molto vasto, spesso completo. Quello difficile è tipico dei giochi ad interazione simultanea, dove tutti i giocatori agiscono nello stesso momento o comunque senza conoscere le mosse altrui; sono dunque costretti ad ipotizzare le scelte del rivale per calcolare il risultato ottimale del gioco e della strategia. Il Dilemma del Prigioniero ne è un esempio importante.

2.1.2 Forma estesa e forma strategica

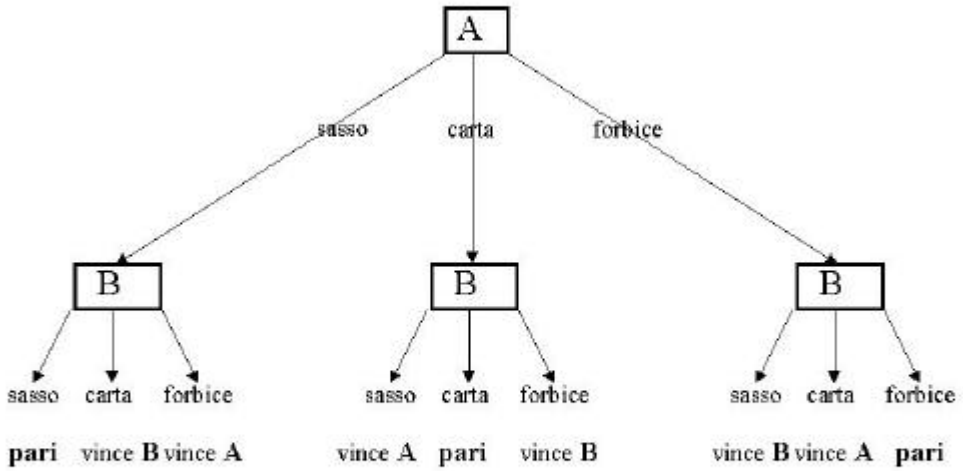
Per riuscire a seguire la complessa trama delle interazioni strategiche occorre una qualche struttura formale per specificare i rapporti intercorrenti tra le decisioni dei singoli giocatori.

I modelli impiegati per rappresentare un gioco sono essenzialmente due: la *forma estesa* e la *forma normale* o *strategica*.

L'esempio qui riportato é tratto da Schianchi (1997) e descrive il gioco della morra cinese. E' questo un gioco tra due persone, simultaneo. Ogni giocatore può effettuare tre mosse: carta, forbici, sasso. I risultati del gioco nascono dalle diverse combinazioni delle scelte dei due giocatori e possono essere di vittoria di uno dei due o di parità se entrambe scelgono la stessa opzione. Negli altri casi le regole sono:

- carta vince su sasso perché lo copre
- forbici vince su carta perché la tagliano
- sasso vince su forbici perché le rompe.

Una comune rappresentazione in forma estesa del gioco descritto è la seguente.



L'esempio qui rappresentato deve essere letto in questo modo:

- il giocatore A ha tre possibili opzioni che sono rappresentate dalle tre frecce; ogni quadratino con un nome dentro è la rappresentazione di un *nodo*, ovvero di un momento di possibile scelta per quel giocatore;
- per ogni freccia che fuoriesce dal nodo, l'avversario ha a sua volta le stesse tre possibili scelte;
- le nove combinazioni danno luogo ai relativi possibili risultati.

E’ immediatamente evidente che questo tipo di rappresentazione, la forma estesa, è molto efficiente in caso di gioco a scelte sequenziali. Questo schema non è molto utile invece quando il gioco è a interazione simultanea; è infatti difficile rappresentare la simultaneità perché occorre introdurre nel grafico altre linee di non immediata comprensione.

La forma strategica del gioco della morra cinese è invece la seguente.

		Giocatore B		
		Sasso	Carta	Forbici
Giocatore A	Sasso	0,0	-1,1	1,-1
	Carta	1,-1	0,0	-1,1
	Forbici	-1,1	1,-1	0,0

Questa tabella va letta in questo modo. Se, ad esempio, A sceglie “carta” (seconda riga) e B “forbici” (terza colonna), A perde (-1) e B vince (1). Se invece A sceglie “forbici” (terza riga) e B sceglie “forbici” il gioco si conclude con una patta (0,0).

Ciascun giocatore non conosce la scelta che farà il suo avversario (il gioco è a scelte simultanee) ed è costretto a inferirne il possibile e probabile comportamento in base alle informazioni in suo possesso. E’ chiaro che se il gioco non viene ripetuto, le informazioni in suo possesso sono nulle.

Un altro esempio di gioco strategico simultaneo è la decisione di quale notizia pubblicare in copertina che devono prendere settimanalmente “Time” e “Newsweek” (Dixit,Nalebuff 1991). Mentre nei giochi a decisioni sequenziali (come gli scacchi) il ragionamento è lineare (se io faccio così il mio avversario farà quest’altra azione), nei giochi a decisioni simultanee il ragionamento è circolare (se io penso che lui pensa che io penso…).

Per “quadrare il cerchio” ogni giocatore non può osservare la strategia dell’avversario, ma è costretto ad ipotizzarla. In altri termini nei giochi non cooperativi con decisioni simultanee ogni persona deve porsi simultaneamente nei propri e nei panni altrui e quindi scegliere la mossa migliore per entrambe le parti.

Questo fatto è di importanza fondamentale e va messo bene in evidenza: nei giochi non cooperativi a decisioni simultanee (che sono la maggioranza dei giochi reali) ogni individuo non può scegliere solo per sé ma è costretto a scegliere per tutti. E’ per studiare a quali condizioni può sorgere una cooperazione di fatto, non voluta individualmente ma ottenuta collettivamente che si studiano i modelli ad agenti indipendenti. Questi permettono appunto di “studiare mediante il costruire” ed esaminare anche situazioni descrivibili con giochi a decisioni simultanee.

Per ritornare all’esempio, si può supporre, per semplicità, che due siano le principali notizie della settimana: Guerra o Politica. Da ricerche di mercato si sa che, su 100 lettori, 30 sono interessati alla Guerra e 70 alla Politica.

L’editore di “Time” sa che se sceglierà per la copertina le vicende politiche venderà il 70% delle copie, viceversa se pubblicherà Guerra ne venderà solo il 30%.

Sa anche che, se il suo concorrente ("Newsweek") pubblicherà la stessa vicenda il pubblico si dividerà esattamente a metà (35 e 35 o 15 e 15).

Ecco come la situazione può essere rappresentata come un gioco in forma strategica.

		NEWSWEEK	
		Politica	Guerra
TIME	Politica	35,35	70,30
	Guerra	30,70	15,15

La teoria dei giochi affronta e tenta di risolvere due problemi distinti:

- cercare di determinare la strategia migliore per un giocatore, definire se è tale anche per l'altro, cercare di definire qual è il procedimento per arrivare a determinare la strategia ottimale;
- studiare sotto quali condizioni emergono comportamenti collettivi (proprietà emergenti del sistema) a partire da comportamenti elementari dei singoli agenti (tipicamente come si può sviluppare la cooperazione in un gioco non cooperativo interattivo).

In questo lavoro, viene essenzialmente affrontato il secondo problema e la costruzione del modello ad agenti indipendenti (più dettagliatamente descritta nei capitoli successivi) è finalizzata a studiare come si modifichi il comportamento di cooperazione al mutare delle condizioni ambientali.

2.2 Il Dilemma del Prigioniero

L'ipotesi su cui si fonda il modello qui presentato è che i soggetti agenti all'interno del modello, perseguendo il proprio interesse in assenza di un'autorità centrale che in qualche modo li costringa a collaborare tra loro, sviluppano una forma di cooperazione. L'ipotesi di comportamenti guidati dal tornaconto personale e non, ad esempio, dall'amore per il prossimo o dal bene del gruppo di appartenenza, permette di condurre esperimenti che riguardano il caso più difficile; occorre però sottolineare come questa ipotesi non sia molto restrittiva.

Un buon esempio che permette di illustrare il problema fondamentale della cooperazione è quello di due paesi industrializzati, ciascuno dei quali ha eretto barriere doganali contro le esportazioni dell'altra nazione. Se entrambe le nazioni abbattessero le barriere doganali ne ricaverebbero reciproci vantaggi derivanti dal libero scambio. Però se l'iniziativa di ridurre le barriere viene presa unilateralmente, l'economia di quel paese si troverebbe di fronte ragioni di scambio certo non favorevoli. In realtà, qualunque cosa faccia uno dei due paesi, l'altro ha interesse a mantenere le proprie tariffe protezionistiche. Nasce di qui il problema: ciascun paese ha l'incentivo ha mantenere le barriere all'interscambio, determinando un esito peggiore di quello che si avrebbe in caso di cooperazione tra i paesi.

Il nocciolo del dilemma risiede nel fatto che il perseguimento del proprio interesse da parte di ciascuno determina un risultato meno positivo per tutti, anche

per se stessi. Per cercare di districarsi nelle innumerevoli possibili situazioni specifiche, è necessario formalizzare il problema in modo da evitare di essere fuorviati dai dettagli di ogni singolo caso. Il gioco del Dilemma del Prigioniero è il risultato di questa formalizzazione.

Il Dilemma del Prigioniero venne elaborato da Merril Flood e Melvin Dresher negli anni cinquanta e può essere formalizzato nel modo seguente.

Due criminali vengono indagati, ma la polizia non ha prove sufficienti per arrestarli. Sarebbe necessario che almeno uno dei due confessasse.

Vengono perciò interrogati e ad ognuno separatamente, la polizia offre il seguente patto:

- se uno dei due denuncia l'altro lo rilasceremo mentre l'altro verrà condannato al massimo della pena;
- se nessuno dei due denuncia l'altro sarete entrambi trattenuti per un periodo di carcerazione preventiva;
- se vi coinvolgete a vicenda sarete entrambe condannati, ma il fatto di aver cooperato vi permetterà di ottenere uno sconto sulla pena.

Supponiamo di dare il punteggio –1 al massimo della pena, 0 per la pena scontata, 3 per la carcerazione preventiva e 6 per la libertà, si può rappresentare la situazione come un gioco in forma strategica.

		2° corruttore	
		Denuncia	Non denuncia
1° corruttore	Denuncia	0,0 (P,P)	6,-1 (T,S)
	Non denuncia	-1,6 (S,T)	3,3 (R,R)

Per il primo criminale "Denuncia" è la strategia dominante poiché i payoff associati sono più grandi dei rispettivi payoff relativi all'altra strategia (0>-3, 6>3). Analogamente per il secondo criminale. Il risultato della scelta razionale è che entrambi sceglieranno di coinvolgere il complice con il risultato di essere entrambi condannati, seppur con una pena più lieve.

Il fatto è che la coppia di risultati (0,0) non è l'ottimo del gioco. Infatti la coppia di risultati (5,5) è collettivamente migliore perché garantisce che tutti e due i giocatori guadagnino passando ad essa. Anche supponendo che i due criminali si accordino, il Dilemma del Prigioniero non perde di significato; entrambi hanno infatti ancora interesse a denunciare l'altro nella certezza che il complice mantenga la parola data.

Che cosa è dunque necessario per essere nella situazione del "Dilemma del Prigioniero"? Due giocatori ciascuno dei quali ha due possibilità di scelta, cooperare o defezionare (nell'esempio precedente denunciare o non denunciare), dovendo però effettuare la scelta senza sapere che cosa farà l'altro. Il dilemma sta nel fatto che se operano razionalmente, ovvero entrambe defezionano, entrambe ottengono un risultato inferiore rispetto a quello che avrebbero avuto se entrambi avessero cooperato. Questo gioco costituisce la base dell'intero modello qui presentato.

Per la definizione del gioco si impone che valgano certi rapporti tra i punteggi ottenibili nelle varie situazioni. La vincita più alta possibile è quella che più avanti

(capitolo 4) viene indicata con T, il premio alla tentazione di tradire quando l'altro ha deciso di cooperare. Il risultato peggiore è S (Sucker's pay, la paga del babbeo) che riceve chi ha deciso di cooperare quando l'altro giocatore ha deciso di defezionare.
R, il punteggio di reciproca cooperazione deve essere superiore a P, la penalità per la reciproca defezione.

L'ordine dei punteggi deve perciò essere T >R > P > S.

Per definire correttamente il dilemma occorre ancora stabilire che i giocatori non possano sottrarsi al dilemma sfruttandosi alternativamente l'un l'altro. Il punteggio ottenibile con un'alternanza di T e di S deve essere inferiore a quello ottenibile con reciproche cooperazioni. Ovvero T + S < 2*R.

Il "Dilemma del prigioniero" resta così definito dalle due condizioni viste prima:

T >R > P > S
R > (T + S) / 2

Se il "Dilemma del prigioniero" viene giocato una volta sola (single shot game) è logico aspettarsi che emergano le strategie dominanti del gioco, ovvero la defezione per entrambi i giocatori. Ma anche se i due giocatori avessero un numero noto e prefissato di incontri, si instaurerebbe una situazione simile. Infatti, all'ultima mossa non ci sarebbe nessun incentivo a cooperare; ma neanche alla penultima, poiché entrambi penserebbero che all'ultima ci sarà una defezione da parte dell'avversario. Questo ragionamento sarebbe applicato "all'indietro" e renderebbe impossibile la cooperazione reciproca.

Il ragionamento non regge più se i giocatori si incontrano ripetutamente, ma un numero non definito di volte. Oltretutto nelle condizioni normali accade appunto che sia praticamente impossibile determinare a priori quanti incontri ci saranno.

La possibilità di cooperazione viene offerta se il gioco viene ripetuto per un numero indefinito di incontri.

Le altre ipotesi inserite nel modello qui proposto vengono qui brevemente elencate:

- l'interazione è tra due soli giocatori alla volta; un giocatore interagisce con tutti gli altri, ma sempre con un solo giocatore alla volta;
- ciascun giocatore è in grado di ricordare la storia precedente dei suoi incontri;
- non esiste alcun meccanismo a disposizione dei giocatori che consenta minacce o impegni realizzabili con certezza
- non c'è modo di essere sicuri di che cosa farà l'altro giocatore
- non si possono basare le proprie scelte sui risultati della precedente interazione del proprio avversario con un terzo giocatore; la sola informazione disponibile riguarda la storia dell'interazione pregressa tra i due giocatori impegnati in quel momento;
- non c'è modo di sottrarsi all'interazione
- non c'è modo di modificare i risultati conseguibili.

I giocatori comunicano tra di loro solamente attraverso la sequenza delle azioni decise e comunicate; questa è l'ipotesi fondamentale del "Dilemma del

Prigioniero". Ciò che rende possibile la nascita della cooperazione è, in questo modello, esclusivamente l'eventualità che i due agenti possano incontrarsi in una seconda occasione. Questa possibilità significa che la scelta effettuata oggi non determina solamente l'esito dell'incontro attuale ma influisce sulle scelte future di entrambe i contendenti; il futuro incombe sul presente e incide sulla strategia attuale.

2.3 Il problema del rumore

Un importante caratteristica delle interazioni nel mondo reale è che le scelte non possono essere implementate senza errori. Poiché l'altro giocatore non sa necessariamente se una certa scelta è un errore o una scelta deliberata, un singolo errore può portare a drammatiche complicazioni. Per esempio, nel settembre del 1983, un aereo della compagnia aerea Sud Coreana volò per errore sul territorio dell'Unione Sovietica. Fu abbattuto dai sovietici, uccidendo tutte le 269 persone che erano a bordo. Americani e sovietici manifestarono l'un l'altro la propria rabbia in una breve ma rapida escalation di tensioni tipiche della guerra fredda (Goldstein 1991).

Gli effetti dell'errore sono stati trattati sotto la dizione di "rumore", ovvero di quella situazione che impedisce di decifrare correttamente il comportamento del proprio avversario. Il modo migliore di trattare il rumore è una questione vitale nella teoria dei giochi, specialmente nel contesto del dilemma del prigioniero reiterato. Chiaramente, quando il rumore viene introdotto, può accadere che la volontà di cooperare di uno dei due giocatori venga "intesa" come defezione a causa degli effetti del "rumore"; vi possono essere perciò alcune defezioni non intenzionali. Questo può ridurre l'efficacia delle strategie di semplice reciprocità.

Che cosa intendiamo qui per rumore? Si trova in Axelrod (1997)

> When I set up the computer tournaments for the Prisoner's Dilemma, I allowed for the possibility of random errors in a very simple way. I did this by informing all the entrants that one of the rules in the tournament would be a purely random strategy. Afterward, I realized that this form of randomization did not really deal with the problems of possible misunderstanding of the coice made by the other player or possible misimplementation of one's own intended choice.

Rumore è quindi una generica possibilità di errori casuali, non voluti.

Sono stati introdotti nel modello e studiati due diverse forme di rumore, indicate per brevità come "rumore ambientale" (che concerne l'ambiente stesso, che in questo caso è costituito dall'insieme delle strategie in gioco) e come "rumore di canale" (che riguarda il canale di comunicazione attivo tra i giocatori).

Con il primo si intende che gli errori casuali fanno parte della definizione dell'ambiente stesso, ovvero che tra le strategie comprese nel modello ve ne sono alcune (comunque almeno una come peraltro già previsto nell'esperimento di Axelrod) che generano scelte casuali. L'assunzione che esistano strategie che scelgono a caso può apparire bizzarra poiché, presumibilmente, nessuno si immagina

che possano esistere persone reali, organizzazioni o stati che scelgono in modo casuale. Se si osserva però l'interazione "dalla parte dell'avversario" si può notare che questi non può distinguere tra una strategia che sceglie a caso e un'altra, che è sì coerente, ma a regole che non riesce a interpretare. RANDOM può essere vista non come una strategia che sceglie a caso, ma come una strategia che viene sistematicamente fraintesa dai suoi avversari. Certamente, per essere sicuri che, in un modello a calcolatore, una strategia non sia capita la soluzione tecnica più conveniente è generare azioni in modo casuale.

Il secondo tipo di rumore si riferisce a quegli errori che avvengono nella fase di comunicazione delle rispettive azioni decise dai due agenti che si incontrano. In questo caso la strategie è ben definita, ma la comunicazione all'agente avversario è soggetta ad errori.

Axelrod propone di seguire tre differenti orientamenti per cercare di trattare il problema del rumore.

1) Generosità. Permettere che, in una certa percentuale di casi, l'avversario defezioni senza effettuare ritorsioni è stata indicata da molti autori come una possibile soluzione al problema. Ad esempio si potrebbe introdurre nel torneo una versione "generosa" di TIT FOR TAT, chiamiamola GTFT, la quale, in tutti i casi in cui TIT FOR TAT deciderebbe di defezionare, innesca un scelta casuale a probabilità prefissata (diciamo 10% dei casi) che può viceversa condurre alla decisione di cooperare.
2) Pentimento. Una strategia basata sulla reciprocità come TIT FOR TAT può essere modificata affinché eviti di rispondere alla defezione che l'avversario ha fatto seguire alla propria defezione non intenzionale. Questo consente di trovare una strada rapida per ritornare alla reciproca cooperazione. E' basata sull'idea che non bisognerebbe essere provocati dall'azione di defezione del proprio avversario che segue una propria defezione non intenzionale. La strategia che si potrebbe introdurre nel torneo potrebbe essere una versione "contrita" di TIT FOR TAT, chiamiamola CTFT. Questa strategia ha tre stati: "pentita", "contenta", e "provocata". Inizia nello stato di "contenta" cooperando e rimane in questo stato a meno che sperimenti una defezione unilaterale. Se è stata vittima di una defezione mentre era in uno stato di "contenta", entra nello stato di "provocata" e defeziona fino a quando una cooperazione da parte dell'avversario causa il ritorno allo stato di "contenta". Se è stata lei a defezionare mentre era nello stato di "contenta" (defezione dovuta al "rumore") entra nello stato di "pentita" e coopera. Quando è in questo stato di pentimento, ritorna nello stato di contenta solamente in caso di cooperazione anche dell'avversario.
3) Se vinci stai, se perdi cambia. Può essere usata una strategia completamente differente, basata sul principio che se il payoff più recente è stato alto, andrà ripetuta la stessa scelta, ma la scelta andrebbe invertita in caso opposto. Questa strategia è emersa da un processo evolutivo simulato che includeva il rumore, ma che consentiva alle strategie di "ricordare" solamente la mossa precedente (Nowak and Sigmund 1993). Nuovi recenti studi (Kraines e Kraines 1995) considerano Win-Stay, Lose-Shift nel più ampio contesto delle strategie Pavlov, vale a dire strategie che imparano e modificano la loro risposta in base ai risultati conseguiti. Questo insieme di strategie sono sovente studiate introducendo

parametri per definire la “velocità” con cui effettuano lo shift di comportamento in base ai più recenti risultati.

Sia Pavlov sia la versione “pentita” di TIT FOR TAT hanno la desiderabile caratteristica di riuscire velocemente a ritornare a cooperare dopo un errore isolato quando il match è con la loro strategia gemella. Ad esempio, se si esamina l’incontro tra due giocatori che utilizzano entrambe la strategia Pavlov, si può osservare la seguente sequneza di azioni. Alla defezione isolata (causata, per errore, dal rumore) di uno dei giocatori seguirà una defezione da parte di entrambe. Questo però li porterà entrambe a collaborare nuovamente alla mossa ancora successiva.

Analogamente in un incontro tra due giocatori che utilizzano CTFT (versione “contrita” di TIT FOR TAT). Quando un giocatore risponde con una defezione non provocata (anche qui causata dal rumore) entra in uno stato di pentimento che lo porterà a collaborare alla mossa successiva anche se l’altro giocatore defezionerà. Entrambe ritorneranno a collaborare alla mossa ancora successiva, ricuperando completamente l’errore causato dal rumore.

La capacità di recuperare la reciproca collaborazione dopo un errore non intenzionale non è senza costi. Infatti, per un agente che usa la strategia Pavlov, la sua volontà di continuare a cooperare dopo una mutua defezione può dare all’avversario un incentivo per continuare semplicemente a defezionare.

2.4 L’evoluzione di nuove strategie

Una problematica domanda che sorge spontanea dall’esame del torneo originale di Axelrod (vedi capitolo 4) è se il successo della strategia Tit for Tat non dipenda dalle personali credenze dei vari autori delle strategie presentate e che generano un ambiente “inconsapevolmente” favorevole a Tit for Tat. In altre parole, i risultati del torneo sono influenzati dal fatto che tutti i partecipanti, avendo idee preconcette sulla necessità di strategie “buone” e che attuano azioni di reciprocità, fanno emergere Tit for Tat come vincitore oppure un atteggiamento di reciprocità come quello di Tit for Tat sorge spontaneamente senza preconcetti di sorta sulle credenze e sul tipo di risposte dato dagli altri agenti del modello?

Per rispondere a questa domanda occorre modificare il modello utilizzato per lo studio sugli effetti del rumore ed inserire almeno un agente che si possa definire evolutivo, ovvero che possa generare autonomamente strategie per il “Dilemma del Prigioniero”

I metodi di evoluzione, usati negli ambienti artificiali, agiscono, in modo iterativo, su una popolazione di individui con lo scopo di aumentare le prestazioni espresse dai singoli nella interazione con l'ambiente. La struttura degli individui ed il tipo di operazioni effettuate possono variare a seconda del tipo di algoritmo. La popolazione iniziale viene, solitamente, generata in modo casuale ed automatico dal programma e le operazioni compiute sui dati sono piuttosto semplici. Una misura del grado di adattabilità di ciascuna struttura permette di indirizzare le operazioni di

ricambio generazionale: assegnando maggior possibilità di riproduzione agli individui migliori gli algoritmi evolutivi sono in grado di esprimere una generale capacità di adattamento basato sull'esperienza. Tutte le selezioni vengono operate con criteri non strettamente deterministici allo scopo di mantenere un certo grado di eterogeneità della popolazione tale da garantire una esplorazione sufficientemente accurata dello spazio degli stati dell'ambiente simulato dal modello. In forza delle loro caratteristiche, gli algoritmi evolutivi permettono di evitare l'onere di prevedere e codificare a priori tutta una serie di dati relativi:

- alla popolazione iniziale,
- agli obiettivi, anche intermedi,
- alla conoscenza dell'ambiente,

presentando in ciò caratteristiche di elevata comodità di utilizzazione. I principali metodi utilizzati per studiare modelli di tipo evolutivo sono: gli algoritmi genetici ed i classifier system.

2.4.1 Algoritmi genetici

Gli algoritmi genetici stanno ricevendo una attenzione particolare da parte di studiosi delle più disparate discipline, interesse causato sia da fattori interni al metodo stesso sia da motivazioni esterne.

Fra i più importanti fattori interni si possono elencare:

- la possibilità di agire in situazioni di scarsa conoscenza della natura e struttura del problema da risolvere,
- la facilità di integrazione con tecniche di ottimizzazione tradizionali
- la possibilità di sfruttare vantaggi derivanti dall'agire in collaborazione con altri paradigmi dell'intelligenza artificiale come le reti neurali.

Le motivazioni che possiamo definire esterne riguardano:

- il superamento delle tendenze basate sull'aspetto strettamente numerico dei problemi
- la possibilità di evitare taluni limiti, insiti in altre tecniche di ottimizzazione, che possono costituire ostacolo all'utilizzazione in applicazioni non convenzionali.

Oppure, in Goldberg (1989):

> (…) nuovi utenti si avvicinano agli algoritmi genetici per via del fascino di una metodologia basata sulle procedure naturali di ricerca e di scelta.

Per quanto attiene alle scienze economiche (Margarita 1992), esistono punti specifici di interesse nelle caratteristiche "quasi ottimali" delle soluzioni offerte dagli algoritmi genetici, che risultano particolarmente coerenti con le teorie di razionalità limitata e con i modelli comportamentali degli agenti economici:

> (…) dotati di un sistema decisionale approssimativo ma rapido piuttosto che rigoroso ed esaustivo (…)

Il ricorso a processi genetici favorisce il ritorno ad una economia basata sull'uso di strumenti più naturali, dotata di maggiore affinità con l'economia

qualitativa, in contrapposizione con l'eccessiva infiltrazione matematica che caratterizza le trattazioni quantitative.

Per quanto attiene alla struttura ed alla definizione degli algoritmi genetici si trova in Goldberg (1989):

> Gli algoritmi genetici sono algoritmi di ricerca basati sui meccanismi di selezione naturale e sulla genetica. Integrano un principio di sopravvivenza della struttura maggiormente adattata all'ambiente con un meccanismo di scambio di informazioni, strutturato ma stocastico, per formare un algoritmo di ricerca che presenta un po' del fiuto innovativo insito nella ricerca umana. Ad ogni generazione un nuovo insieme di creature artificiali (sequenze) è generato sulla base degli individui migliori dell'insieme precedente.

Negli studi economici è ricorrente il problema della ottimizzazione vincolata o programmazione matematica, normalmente inteso come ricerca di valori di una (o più) variabili in modo che, rispettando condizioni di vincolo, sia massimi (o minimi) i valori corrispondenti della funzione obiettivo. La risoluzione di un simile problema non presenta, solitamente, difficoltà qualora le leggi che vincolano le varie grandezze siano note ed esprimibili in una funzione che presenti caratteristiche, tipicamente di continuità e derivabilità, tali da permettere di applicare ad essa gli strumenti del calcolo differenziale. Possono invece emergere complicazioni insormontabili quando le leggi suddette non siano perfettamente conosciute oppure quando la funzione non risulti esprimibile matematicamente in modo tale da rendere impossibile l'applicazione dei sistemi di calcolo prima accennati.

Quando si faccia ricorso agli algoritmi genetici (Holland 1975), la ricerca della soluzione di un problema, si caratterizza come il tentativo di esprimere una struttura che rappresenti azioni in grado di portare il sistema ad interagire al meglio con l'ambiente in cui opera; in pratica l'obiettivo è stabilire quali organismi meglio si adattino ad un dato contesto. Quando i vincoli ambientali non siano noti od emergenti, l'osservazione attenta dell'evolversi delle strutture permette di trarre utili conclusioni sulle regole di funzionamento del sistema ambientale in oggetto.

Immaginiamo di poter condurre sperimentazioni volte a misurare il grado di adattamento all'ambiente di ogni singolo individuo; è possibile definire "piano" la strategia di movimento all'interno dell'universo delle alternative possibili.

Il più semplice piano è quello detto "enumerativo": tutte le possibili alternative vengono valutate in un ordine non influenzato dai risultati degli esperimenti già effettuati; questo procedimento è sicuramente in grado di individuare la struttura migliore, memorizzando, ad ogni esperimento, quella con il grado di adattamento più alto. Il limite di applicabilità di una siffatta strategia sta nel fatto che le possibili alternative per un sistema non particolarmente complesso è, di solito, enormemente alto, tale da rendere non praticabile simile approccio.

Risulta evidente l'obbligo di garantire un'efficienza del procedimento, in termini di tempi di esecuzione, non disgiunta dalla salvaguardia dell'efficienza in termini di capacità di determinazione delle soluzioni. Questo obiettivo è materia della teoria riguardante i piani "adattivi": piani che si modificano in base ai risultati delle precedenti interazioni in modo da recepire gli elementi di conoscenza che

provengono dall'ambiente. Questi piani devono consolidare i progressi fatti e usare le conoscenze acquisite per incrementare la quantità di strutture maggiormente "adattate" da sperimentare. La loro azione riguarda un insieme iniziale di entità che vengono progressivamente modificate, tramite appositi operatori, per generarne di nuove, capaci di meglio operare nell'ambiente di riferimento.

Un sistema in fase di adattamento (Holland 1975) è caratterizzato dalla contemporanea presenza, in ogni passo evolutivo, di diversi operatori agenti su strutture; il piano determina quali strutture debbano crescere, cioè riprodursi, e quali scomparire, in funzione delle capacità di adattarsi alle caratteristiche dell'ambiente.

In questo processo agiscono tre componenti principali:

- il contesto ambientale (E),
- il piano di adattamento (P)
- un insieme di sistemi di misurazione dei risultati, cioè della adattabilità delle strutture (M), naturalmente dipendente dal contesto.

Non è richiesta una completa conoscenza delle caratteristiche dell'ambiente essendo sufficiente la cognizione di quelle caratteristiche in grado di determinare delle alternative nello sviluppo del piano.

In pratica, il funzionamento del processo, si basa sulla codificazione dei parametri del problema in strutture: nelle applicazioni che si avvalgono dell'uso di un elaboratore ciò può essere facilmente ottenuto utilizzando stringhe di caratteri, lunghe a piacere, solitamente definite su un alfabeto binario formato dai caratteri "0" e "1"; l'insieme di queste stringhe costituirà la popolazione iniziale dell'algoritmo genetico, mentre le singole stringhe ne saranno gli individui. Unicamente per semplicità nello sviluppo dei programmi di solito si opera con popolazioni formate da un numero fisso di stringhe della stessa lunghezza; ad ogni ciclo, in numero di individui destinati a soccombere è pari a quello dei nuovi nati. La creazione della generazione iniziale è di solito condotta in modo casuale a meno che non si voglia partire da una popolazione già selezionata in base ad altre elaborazioni.

Ad ogni ciclo ciascun individuo viene valutato in termini di idoneità ad operare nell'ambiente. Questa idoneità è nota nella letteratura come *fitness* ed è calcolata in base ai valori assunti dalla funzione di misurazione dei risultati; tale valore influenza la probabilità assegnata a ciascuna stringa di essere selezionata per la riproduzione e, di conseguenza, la possibilità di sopravvivere. La riproduzione viene effettuata copiando, una o più volte, una stringa genitore. La selezione può avvenire, come d'uso, con metodo stocastico basato sull'assegnazione di elevate probabilità di estinzione alle stringhe con più bassi valori di *fitness*, secondo una impostazione tipicamente darwiniana. Il ricambio generazionale può riguardare l'intera popolazione, tutti gli individui vengono rimpiazzati ad ogni ciclo (*generational replacement*), oppure essere limitata ad alcuni individui (*steady-state reproduction*).

Con la riproduzione si attua solo una scelta degli individui migliori all'interno della popolazione iniziale; per scoprire nuove strutture è necessario avvalersi di altri operatori. Ciascuno di questi operatori realizza una manipolazione delle stringhe in

base a metodi propri; e' possibile immaginare un numero pressoché infinito di operatori ma i due basilari sono:

- il crossover
- la mutazione.

Il crossover è articolato in due fasi: si scelgono, casualmente, due individui della nuova generazione ed una posizione all'interno della stringa (scelta a caso), le parti a sinistra della posizione scelta delle due stringhe vengono scambiate tra loro, in modo da ottenere due individui diversi da quelli di partenza. Ovviamente è possibile immaginare crossover più complicati ma questo semplice qui descritto sembra funzionare sufficientemente bene per la maggior parte delle applicazioni. Quando il crossover agisce in collaborazione con la riproduzione, partendo da due genitori che fondono il loro patrimonio genetico contribuendo ciascuno a quello dei nuovi nati, si parla di riproduzione "sessuata".

La mutazione simula gli errori nella trasmissione del patrimonio genetico; la si attua modificando il valore di una posizione scelta a caso. Di solito la probabilità che questo processo avvenga è molto bassa; ciò non di meno essa è indispensabile per effettuare una efficiente esplorazione dello spazio delle soluzioni.

La convergenza della popolazione su un certo schema rivela, solitamente, il raggiungimento di una configurazione di strutture che, pur potendo non essere quella ottimale, sicuramente risulterebbe migliore di quella di partenza. In dipendenza di questa osservazione l'azione dell'algoritmo viene, normalmente, interrotta al raggiungimento di un grado stabilito, piuttosto elevato, di similitudine o omogeneità della popolazione; l'interpretazione del genotipo prevalente fornisce la soluzione ricercata del problema. Può comunque essere utile osservare la popolazione in stadi diversi del processo onde evincere quale sia l'operato del meccanismo di selezione; dal procedere di esso è possibile ottenere informazioni sul funzionamento dell'ambiente del tutto nuove rispetto a quelle conosciute fino a quel momento.

2.4.2 Classifier systems

Vengono qui introdotti questi sistemi evolutivi in modo molto sintetico, essendo essi oggetto del prossimo capitolo. Per rendere semplice il confronto tra i due metodi presentati, possiamo dire che i classifier system sono un'applicazione degli algoritmi genetici al problema di conferire capacità di apprendimento ai tradizionali sistemi esperti. Un altro vantaggio deriva dal metodo di selezione delle regole: nei sistemi esperti la scelta della regola da applicare viene fatta ad ogni passo, ogni scelta vincola lo sviluppo della ricerca; nei *classifier system* la selezione viene posposta dopo la verificazione di tutti i *match*, così viene incrementato il parallelismo del metodo e resa possibile la gestione di problemi diversi nell'ambito di una medesima applicazione.

Il valore di ciascuna regola di un sistema di produzioni viene assegnato in modo fisso dall'esperto che si occupa della sua istruzione iniziale; nei sistemi evolutivi, al contrario, l'originale valore viene modificato in proporzione ai risultati ottenuti dalla regola, divenendo oggetto di apprendimento. L'importanza delle varie regole risente, in questo modo, della loro idoneità ad operare nell'ambiente e può modificarsi in seguito a variazioni intervenute nel contesto di applicazione; lo stesso

valore viene usato dall'algoritmo genetico per decidere quali regole dovranno riprodursi e quali scomparire. La gestione del valore delle regole costituisce l'elemento più importante di un *classifier system;* l'algoritmo che presiede a questa attività può assumere connotazioni estremamente variate, ma quello normalmente utilizzato è il *bucket brigade algorithm* (Holland 1975) basato sulla competizione, dei singoli *classifier*, per ottenere il diritto di inserire il proprio messaggio nell'insieme relativo, e su un meccanismo di ripartizione dei valori tra le varie regole che concorrono alla decisione finale.

2.4.3 Un confronto sulla leggibilità dei risultati

Presentiamo ora sia un'applicazione degli algoritmi genetici alla risoluzione del Dilemma del Prigioniero sia una breve descrizione dell'applicazione dei classifier system allo stesso problema (che sarà però più approfonditamente descritta nel capitolo 5); questo ci consentirà di mettere in luce le differenze tra i due metodi.

Algoritmi genetici. Il primo problema da risolvere per costruire un'applicazione che utilizzi gli algoritmi genetici è la codificazione delle mosse durante il gioco; e per definire la codificazione occorre primariamente decidere quali sono gli "stimoli" ambientali che verranno utilizzati e che quindi necessita codificare. Si può scegliere, ad esempio, di considerare le tre mosse precedenti proprie e dell'avversario. Se con la C si indica la cooperazione e con D la defezione si possono verificare i seguenti casi:

CC CC CC (caso 1)
CC CC CD (caso 2)
CC CC DC (caso 3)
CC CC CD (caso 4)
Ecc.

La codificazione è ovviamente arbitraria; si può pensare che le prima due lette rappresentino la propria mossa e quella dell'avversario di tre turni prima, la seconda coppia di lettere le analoghe mosse di turni prima e l'ultima coppia del turno immediatamente precedente.
I casi possibili sono $2^2 \times 2^2 \times 2^2 = 64$; secondo questo schema ogni singola strategia può essere codificata mediante una stringa di 64 posizioni del tipo CDDCCDCCCDDDCD… che rappresentano le azioni decise da quella strategia per ognuna delle 64 combinazioni possibili. Se, ad esempio, durante l'incontro si ha che dopo aver entrambe cooperato tre turni fa e due turni fa, al turno prededente l'avversario defeziona mentre il giocatore coopera; siamo nel caso 4 dell'esempio. Quale azione decide la strategia? Va a leggere la stringa di 64 posizioni alla quarta posizione e se vi trova C coopera mentre se vi trova D defeziona.

Per completare la descrizione di tutti i possibili casi, bisogna prevedere anche come definire il comportamento nelle prime tre mosse, poiché non vi è ancora "storia dell'incontro" sufficiente da permettere di applicare la codifica descritta. Si potrebbero usare ulteriori 6 posizioni (e quindi allungare la stringa a 70) per codificare una sequenza presunta che viene usata quando si è nelle prime tre mosse.

Definita la stringa nonché la sua codifica e decodifica, si possono applicare le tecniche di evoluzione e selezione prima descritte.

Classifier Systems. Vediamo ora una breve descrizione (i Classifier System sono argomento del successivo capitolo) di come codificare le stesse informazioni per un Classifier System. La codifica vista prima si presta ad una immediata applicazione; la sequenza delle tre mosse precedenti è la "condition part" di una delle regole inserite nel Classifier System mentre l'azione decisa è la "action part". Qui si utilizza però un codice ternario; infatti oltre a C e D prima visti, c'è anche una wild card **#** che segnala la non importanza, il "don't care" della posizione che occupa nella condition part.

Leggibilità dei risultati. Immaginiamo di aver costruito entrambe i modelli e di dover leggere ed eventualmente confrontarne i risultati.
L'output del modello evolutivo ad algoritmi genetici è rappresentato da una (o anche più, se il criterio di ricerca delle soluzioni non è di massimo punteggio ottenibile, ma di garantire un punteggio minimo) stringa di 64 posizioni di C e D (o di 0 e 1 se si preferisce una codifica numerica). Questa descrizione della strategia possiamo definirla "enumerativa", nel senso che descrive, una per una, tutte le condizioni - azioni possibili. Tutte le combinazioni hanno la stessa importanza relativa, può essere molto complicato capire se la stringa così determinata rappresenta una "logica" di soluzione.

Immaginiamo che la strategia migliore risulti Tit for Tat. Come verrebbe codificata come stringa? Poiché l'unica informazione che interessa a Tit for Tat è l'azione dell'avversario alla mossa precedente, nella codifica vista prima sarà:

CDCDCDCDCDCDCDCDCDCDCDCD…

Anche questo risultato ben noto può non essere così facile da decifrare.

Nel caso di Classifier System, Tit for Tat sarebbe codificato con due sole regole:

#D : D
#C : C

La leggibilità del risultato è sicuramente superiore in questo secondo caso.

Poiché entrambe i metodi offrono la possibilità di studiare modelli evolutivi, e che i Classifier System sembrano offrire una maggiore facilità di lettura dei risultati, si è optato per utilizzare questo secondo metodo per inserire nel modello ad agenti indipendenti la possibilità di far evolvere autonomamente strategie che giocano al "dilemma del Prigioniero".

Capitolo 3

Classifier Systems

3.1 Introduzione

I Classifier Systems (CS) sono nati nell'alveo degli studi sull'intelligenza artificiale e possono essere considerati come un'evoluzione dei più tradizionali sistemi esperti. Un CS può essere definito come un sistema di apprendimento automatico che impara regole sintatticamente semplici (chiamate appunto classifier) per decidere le proprie azioni in un ambiente arbitrario. L'inserimento delle tecniche sviluppate per gli algoritmi genetici a sistemi di regole consente quindi di superare un serio limite dei sistemi esperti: la loro incapacità di sviluppare autonomamente soluzioni per il problema a cui sono applicati. La definizione della tecnica è reperibile in Goldberg (1989):

> *A classifier system is a machine learning system that learns syntactically simple string rules (called classifiers) to guide its performance in an arbitrary environment*.

I CS, adottando strutture sintattiche elementari, avulse dal significato semantico delle regole, consentono di operare su di esse tramite un algoritmo genetico; questo agevola considerevolmente la produzione di regole nuove a partire da quelle esistenti e riduce, in pratica annulla, i rischi di ottenere strutture prive di significato o aberranti. Le regole sono "tradotte" in stringhe di zero ed uno: le informazioni provenienti dall'ambiente vengono tradotte nella stessa metrica per mezzo di componenti specializzate, *detector*, i messaggi verso l'ambiente sono correlate alle azioni da eseguire per mezzo di un secondo tipo di componenti, *effector*. Tutte le regole possono essere attivate dai "segnali ambientali" opportunamente tradotti dai *detector* e concorrono, fra loro, per ottenere il diritto di emettere un messaggio che verrà letto dagli *effector* per decidere quale azione proporre nell'ambiente. La partecipazione all'asta comporta costi, così come la immissione di messaggi nella lista, che vanno ad incidere sull'ammontare del patrimonio di crediti di ciascuna regola; tale patrimonio viene rigenerato mediante la ricezione di una ricompensa, proporzionale ai risultati ottenuti dalle azioni proposte. Un algoritmo di scoperta di nuove regole viene fatto agire, di tanto in tanto, sull'insieme delle regole esistenti, per ottenere, con le consuete operazioni di

riproduzione, *crossover* e mutazione, regole nuove; la selezione per la riproduzione o l'estinzione è basata sul valore del patrimonio di crediti di ciascuna regola.

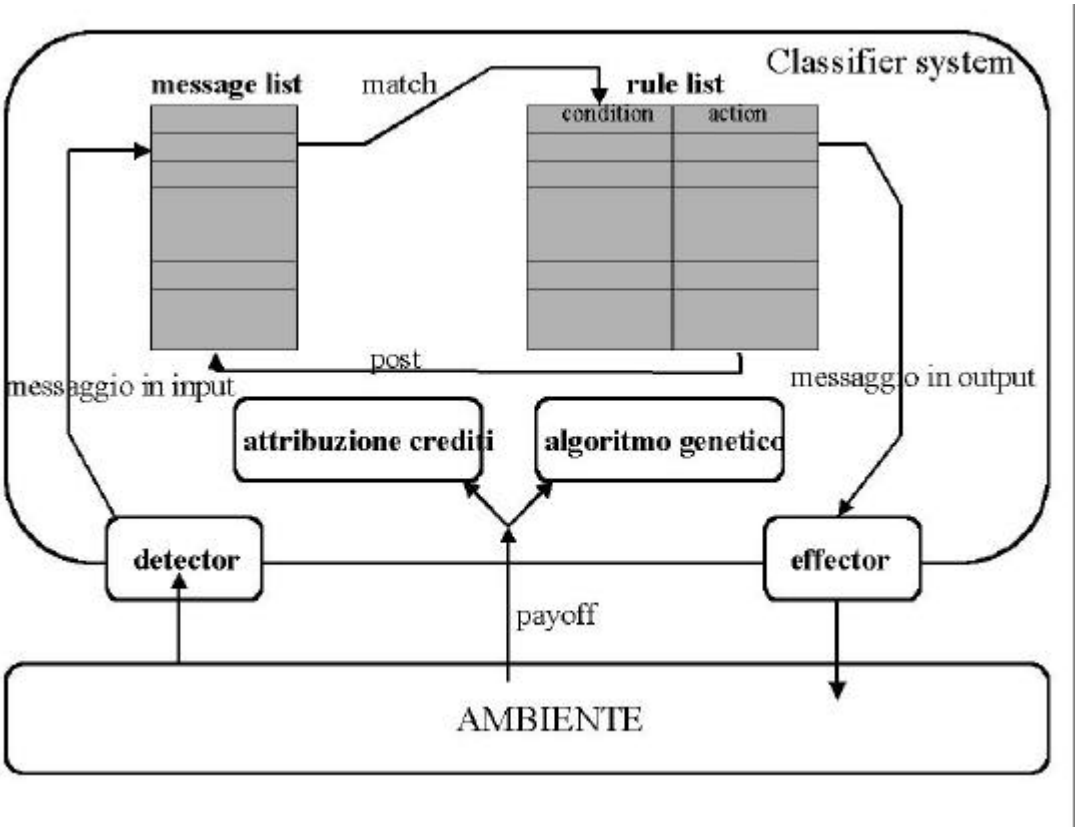
Vediamo ora nei dettagli il funzionamento dei CS.

3.2 Struttura del classifier system

La struttura base di un CS può essere descritta come articolata nelle seguenti parti:

- un insieme di regole; di solito organizzate in una lista, la *rule list*
- un insieme di messaggi che rappresentano gli input ricevuti dall'ambiente così come li ha strutturati il detector; anche i messaggi sono elementi di una lista, la *message list*
- un sistema di attribuzione dei crediti, vale a dire della forza di ciascuna regola
- un algoritmo genetico, per generare nuove regole e modificare quelle già presenti
- un sistema di sensori (*detectors*) per ricevere le informazioni dall'ambiente
- un sistema di attuatori (*effectors*) che traducono i messaggi emanati dalle regole in azioni verso l'ambiente.

Lo schema seguente serve per evidenziare le parti sopra descritte.



Due elementi caratterizzano particolarmente i CSs e li distinguono dai sistemi esperti. Il primo è l'utilizzo delle *rule list* e *message list* che permettono l'attivazione parallela delle regole superando i limiti insiti nell'attivazione sequenziale di regole. Il secondo è l'apprendimento automatico dell'importanza relativa da attribuire a ciascuna regola sulla base di un sistema competitivo di distribuzione dei crediti. Questo meccanismo e quello di creazione e modificazione di regole tramite algoritmi genetici, rendono il sistema molto dinamico e perfettamente adattabile alle sollecitazioni ambientali.

Regole e messaggi. Le regole ed i messaggi sono usualmente costruiti sulla base di un alfabeto ternario {0,1,#} ma nessuna limitazione particolare impedisce l'uso di un numero maggiore di caratteri aumentando così la complessità del sistema; nel modello presentato nel capitolo 5 si è usato l'alfabeto ternario {0,1,#} così come in tutti i lavori rintracciabili nella letteratura sull'argomento; il meccanismo degli algoritmi genetici nelle sue funzioni di mutazione e di crossover, infatti, risulta leggermente semplificato con l'adozione di questo alfabeto.

In sintesi le regole sono rappresentabili come:

SE (la condizione è vera) ALLORA (effettua l'azione)

Il significato di un a regola è perciò quanto mai semplice: al verificarsi della condizione viene effettuata l'azione descritta. Le regole sono quindi stringhe formate da una parte di condizione e da una di descrizione dell'azione; ciascuna regola possiede inoltre un patrimonio di crediti che rappresenta la sua valutazione. I CS adottano una concezione ben delimitata di regola ed assumono, per essa, una lunghezza fissa. Un semplice esempio di regola può essere:

0 1 # # : 0 0 0 1

dove 0 1 # # (cioè la parte a sinistra dei due punti) è la *condition part* e rappresenta la condizione che deve essere verificata affinché venga proposta l'azione descritta da 0 0 0 1 (la parte a destra dei due punti, *action part*). L'azione proposta (come è anche illustrato nello schema precedente) può essere indirizzata all'effector e, per suo tramite, all'ambiente oppure può essere ridiretta verso la message list e attivare altre regole.

Il carattere # (definito come wildcard da Holland 1975) rappresenta il carattere di indifferenza, in relazione al quale il confronto risulta sempre verificato, a prescindere dal valore del termine a cui viene paragonato. L'uso di questo carattere permette di lavorare, durante i primi cicli di apprendimento con regole molto generalizzate, in grado cioè di garantire il verificarsi di una condizione (un *match)* anche in mancanza di regole specifiche. L'azione proposta dalle regole generalizzate risulta piuttosto imprecisa e viene corretta con la produzione di regole specifiche durante l'apprendimento. Assume particolare importanza il concetto di specificità di una regola, quantificabile, in linea di massima, contando il numero di posizioni contenenti valori diversi da # nella stringa. Un buon accorgimento consiste nel rapportare questo valore alla lunghezza totale della stringa, limitandolo all'intervallo]0,1[: risulta, così, possibile confrontare le specificità a prescindere dalle lunghezze e la loro, eventuale, utilizzazione computazionale viene semplificata. La presenza del

carattere # può essere limitata alla parte di condizione della regola (Goldberg 1989) oppure essere estesa anche alla parte di azione.

Il sistema di distribuzione dei crediti. Ad ogni regola è abbinato un valore che ne rappresenta la forza (*strength*).Questo valore può essere interpretato come patrimonio o potere economico di un a regola in quanto è questo valore che entra nel meccanismo di asta per scegliere quale regola viene adottata. A seconda dell'obiettivo della simulazione la popolazione iniziale del CS può essere generata casualmente oppure in modo da incorporare regole e comportamenti predefiniti; anche l'assegnazione della forza iniziale di ciascuna regola può essere uguale per tutte o seguire invece altri criteri. Non esistono regole rigide per definire un CS, ma una serie di regole base su cui costruire il proprio modello ed esplorarne il comportamento.

L'attribuzione dei crediti (Holland 1975), in un *CS*, è basata, come abbiamo visto, sulla competizione: le regole, che hanno ottenuto il diritto di agire, pagano una quota del proprio patrimonio a quelle che ne hanno permesso l'attivazione, cioè che hanno prodotto una azione in grado di verificare il *match* con quella regola. Ogni regola viene ad avere, in senso metaforico, fornitori e clienti: i primi sono le regole che hanno concorso alla sua attivazione, i secondi sono quelle attivate dalla sua azione. Quando una regola ottiene il diritto di agire, il suo patrimonio viene ridotto dell'ammontare pagato; tale somma va ad incrementare il patrimonio dei suoi fornitori. Le regole possono recuperare le somme pagate sia perché hanno attivato altre regole vincenti, sia perché ricevono una ricompensa dall'ambiente; le migliori ricevono alte ricompense, quindi pagano forti somme ai propri fornitori, questi ultimi ai loro e così via. Il patrimonio di ciascuna regola cresce quando questa riceve ricompense ambientali superiori alle somme pagate oppure quando è in grado di attivare regole più "ricche" di lei. Se una regola propone una azione sbagliata riceve una ricompensa nulla, quindi il suo patrimonio si riduce; se, invece, attiva un'altra regola, a sua volta errata, provoca un impoverimento del suo cliente che, successivamente, le trasmetterà un minor ammontare di crediti: anche in questo caso il suo patrimonio tenderà a ridursi. Le regole possono sopravvivere solo se fanno parte di una catena di fornitori e clienti, in grado di effettuare le azioni corrette nell'ambiente. Questo algoritmo di distribuzione dei crediti viene chiamato *bucket brigade*: esso richiede che ciascuna regola interagisca esclusivamente con i propri clienti e fornitori diretti, quindi permette di evitare l'onere di memorizzare l'intera catena di regole che lega il messaggio, proveniente dall'ambiente, all'azione. Questo algoritmo consente di gestire, in modo efficiente, i trasferimenti patrimoniali, indipendentemente dal numero di regole utilizzate.

L'algoritmo *bucket brigade* può agire, in modo semplificato, anche quando ciascuna regola possa esclusivamente proporre azioni nell'ambiente: in questo caso non vi sono trasferimenti patrimoniali fra regole. L'analisi di questo algoritmo semplificato può agevolare la comprensione di quello generale. Il sistema funziona come un mercato (Goldberg 1989) dove gli operatori, singole regole, comprano e vendono il diritto di trattare le informazioni; esse costituiscono una catena di intermediari che collega i *detector*, produttori di informazione, agli *effector*, consumatori. Questo mercato è basato su due istituzioni: un'asta ed una stanza di compensazione; la prima permette di selezionare i *classifier* cui sarà attribuito il diritto di agire, la seconda di regolare i pagamenti. Le regole attivate effettuano

un'offerta per un'asta i cui vincitori potranno emettere le proprie azioni, dietro pagamento di un certo ammontare alla stanza di compensazione; le regole possono, inoltre, ricevere dei compensi dall'ambiente e devono pagare alcune tasse.

Il patrimonio, cioè la forza di una regola la tempo t [F(t)] sarà perciò espressione di tutti questi elementi:

F(t)=F(t-1) – pagamento alla stanza di compensazione – tasse + compensi ricevuti

L'ammontare del pagamento alla stanza di compensazione P è determinato dalla somma che la regola, se vincitrice dell'asta, deve versare alla stanza di compensazione, cioè dall'offerta fatta O; quest'ultima è proporzionale al patrimonio, o forza, al tempo t, secondo un coefficiente $\mathbf{a}$:

$P(t)=O(t)=F(t) * \mathbf{a}$

Il vincitore dell'asta potrebbe, facilmente, individuarsi nella regola che ha effettuato l'offerta maggiore; il metodo risulterebbe però (De Groot 1970) troppo deterministico, con conseguenze negative sui risultati. Può essere opportuno l'inserimento di un elemento capace di creare un disturbo casuale: ad esempio aggiungere un valore addizionale casuale calcolato in base ad una funzione di distribuzione (N), caratterizzata da una deviazione standard dipendente da un parametro ($\mathbf{s}$) del sistema:

$O'(t) = O(t) + N(\mathbf{s})$.

L'asta viene vinta dai *classifier* che hanno effettuato le più alte offerte, O', questi però versano alla stanza di compensazione l'ammontare O per ottenere il diritto di eseguire la propria azione.

La tassazione T può essere ispirata a diversi criteri e articolata in una pluralità di imposte. Per semplicità si considera il caso di una unica imposta proporzionale, il cui ammontare risulta determinato dal parametro $\mathbf{t}$:

$T(t) = F(t) * \mathbf{t}$

Aggregando i dati di tutte le regole attive al tempo t è possibile scrivere:

$\mathbf{S}\, F_i(t) = \mathbf{S}\, F_i(t-1) - \mathbf{S}\, F_i(t-1) * \mathbf{a} - \mathbf{S}\, F_i(t-1) * \mathbf{t} + \mathbf{S}\, C_i(t)$.

dove C(t) è il compenso ricevuto dalla iesima regola al tempo t. Con le ipotesi fatte, cioè che sia l'offerta fatta per partecipare all'asta (e poi pagata) sia la tassazione risultino proporzionali alla forza della singola regola, si può semplificare l'espressione mettendo a fattor comune F(t-1) (da qui in avanti si sottintende l'operazione di sommatoria).

$F(t) = (1 - \mathbf{a} - \mathbf{t}) F(t-1) + C(t)$

Ponendo $\mathbf{g} = \mathbf{a} + \mathbf{t}$ si ha:

$F(t) = (1 - \gamma) F(t-1) + C(t)$

Trascurando, per ora, i compensi e considerando n periodi successivi è possibile scrivere la precedente relazione come:

$F(n) = (1 - \gamma)^n F(0)$.

L'espressione F(n) non è divergente per qualunque valore iniziale di F solo se $0 \leq \gamma \leq 2$; la condizione deve essere ulteriormente ristretta, per evitare che F assuma valori negativi, in: $0 \leq \gamma \leq 1$.

Riconsideriamo ora anche le ricompense; le regole che propongono un'azione ricevono, nel ciclo successivo, una ricompensa in funzione dei risultati ottenuti nell'ambiente, quindi in n periodi il sistema riceverà n-1 ricompense ambientali. Il sistema raggiunge uno *steady-state* quando, posto C_{ss} il valore costante ricevuto dall'ambiente e F_{ss} la forza del sistema:

$F_{ss} = C_{ss} / \gamma$

La somma delle offerte in *steady-state* risulta da:

$O_{ss} = (\alpha / \gamma) * C_{ss} = [\alpha / (\alpha + \tau)] C_{ss}$

Normalmente, il coefficiente di tassazione risulta decisamente inferiore a quello per determinare l'ammontare dell'offerta. Almeno in prima approssimazione è quindi possibile considerare sufficiente, per la stabilità del sistema, l'eguaglianza fra corrispettivi pagati dall'ambiente e offerte in asta:

$O_{ss} \gg C_{ss}$

Ogni regola trae un vantaggio dalla partecipazione all'asta: i *classifier* sostengono un costo proporzionale al loro patrimonio e ottengono ricompense in funzione dei risultati conseguiti. Se un *classifier* immette messaggi capaci di attivare un'altra regola si applica l'algoritmo *bucket brigade* completo, il meccanismo risulta meno diretto ma egualmente efficace: la ricompensa sarà proporzionale al patrimonio della regola attivata e quindi dipenderà, indirettamente, dai risultati da questa ottenuti. Diverse regole possono quindi contribuire a formare una catena di trasformazione dell'informazione che termina con la determinazione dell'azione corretta.

L'algoritmo genetico. La ripetizione di cicli di selezione e di apprendimento, basato sull'algoritmo genetico, permette di ottenere popolazioni formate da individui, cioè regole, progressivamente più abili nell'interazione con l'ambiente, fino a raggiungere il livello voluto di risultati. Non è richiesta, per i *CS*, la convergenza delle regole verso un tipo omogeneo, poiché la diversità viene gestita dal sistema di *matching* onde permettere l'evoluzione, contemporanea, di elementi specializzati nell'affrontare singoli problemi.

In realtà non si tratta di una vera e propria evoluzione di regole secondo il metodo degli algoritmi genetici. Le stringhe in cui sono codificate le regole vengono trattate secondo i meccanismi (tipicamente di crossover, di mutazione e di selezione) tipici degli algoritmi genetici, ma non vi è né valutazione della fitness (poiché è presente il meccanismo delle aste) né convergenza verso un tipo omogeneo. L'algoritmo genetico non fa altro che introdurre nel sistema nuove regole ed eliminare quelle vecchie.

Detector e effector. Le interfacce di input e output del sistema (sensori) sono definite rispettivamente detectors ed effectors. I primi trasformano le informazioni ambientali in messaggi compatibili con il CS (nell'esempio fatto stringhe di 0 e 1; il simbolo # *don't care* è usato solo per definire le regole). Questi messaggi vengono poi registrati nella message list e servono per attivare il Classifier. Ciascun messaggio viene confrontato con la condizione di ogni regola: si verifica la corrispondenza di valori a parità di posizione, tenuto presente che il valore # può valere sia come 1 sia come 0; se la verificazione risulta positiva la regola interessata può concorrere per ottenere il diritto di indirizzare verso l'ambiente la propria azione. Al termine dell'accertamento dei *match* viene attivato un algoritmo di scelta per stabilire quali regole potranno agire, cioè attivare un'altra regola o proporre una azione tramite gli *effector*. Come abbiamo visto, per concorrere con gli altri ed attivare la propria azione ogni individuo sostiene un costo che va ad incidere sul suo patrimonio di crediti.

Le azioni vengono comunicate all'ambiente per il tramite degli *effector* e questo fornisce una valutazione della bontà di ciascuna sotto forma di ricompensa (*reward*). L'ammontare del *reward* viene, successivamente, ripartito fra le varie regole che hanno concorso a proporre l'azione; la ricompensa viene, normalmente, mantenuta superiore o uguale a zero per motivi computazionali.

3.3 Proprietà e dinamica del sistema

Le principali proprietà dei CS si possono riassumere nei seguenti tre punti.

Non sequenzialità. Tutte le regole presenti nella rule list possono essere attivate contemporaneamente poiché il messaggio che arriva dall'ambiente (tradotto dal detector) può attivare una o più regole che, a loro volta, possono inviare altri messaggi alla message list.

Sincronismo mediante scambio di messaggi. Come si è visto le regole sono composte da una condition part e da una action part. L'attivazione avviene solamente quando vi sono messaggi nella message list che verificano la condition part; tutti i "meccanismi" del CS sono perciò esclusivamente attivati da messaggi, con la parziale eccezione dell'attivazione asincrona dell'algoritmo genetico.

Non necessità di "interpreti". Le interazioni tra regole avvengono, come detto prima, mediante semplici messaggi, non dipendono dall'ordine con cui sono

nella lista, non si sono necessità di garantire sequenzialità, le operazioni di matching sono piuttosto semplici; tutte queste condizioni fanno si che non vi sia la necessità di introdurre nel CS interpreti di alto livello. Sfruttando la flessibilità dei messaggi ottenibile direttamente utilizzando il linguaggio di programmazione Objective C, è possibile creare una struttura modulare e facilmente modificabile.

Vediamo ora come tutte le componenti descritte interagiscono, seguendo le varie fasi di attività del CS. Anche se il CS non è logicamente sequenziale, ciò non dimeno sono necessari alcuni cicli successivi per completare le operazioni innescate da un cambiamento ambientale.

I *detectors* rilevano o ricevono informazioni dall'ambiente, le "traducono" in uno o più messaggi e li inviano nella message list. Il compito dei messaggi nella lista è di attivare le regole la cui condition part è uguale al messaggio. Va qui ricordato che nella regola è previsto l'uso della wildcard #; nelle posizioni in cui c'è # l'uguaglianza con il messaggio è verificata, qualunque sia il contenuto del messaggio in quella posizione. Le regole attivate dai messaggi diventano candidate all'invio delle proprie azioni nel ciclo successivo. Nelle strutture più complesse di CS l'azione di una regola si traduce prima in un messaggio per altre regole per poi passare verso l'ambiente in un secondo momento. Nel caso di strutture più semplici l'azione viene comunicata immediatamente all'esterno.

Il fatto che una regola risulti effettivamente scelta dipende non solo dal fatto che venga attivata, ma anche dal risultato di una competizione tra tutte le regole attive (asta pubblica). La competizione avviene sulla base della forza di ciascuna regola, forza che viene continuamente modificata in base al risultato delle azioni proposte secondo il meccanismo di bucket brigade visto precedentemente.

Definita la regola che ha vinto l'asta, viene passata all'effector che traduce la action part in una azione verso l'ambiente. Effettuata l'azione riceve dall'ambiente il payoff ossia una valutazione dei risultati ottenuto da quell'azione; il payoff andrà ad influire sul patrimonio della regola responsabile dell'azione e, a catena, su tutte le regole che sono state attivate a monte di quest'ultima.

Ogni N cicli di attività interna del CS vengono attivati i metodi evolutivi e selettivi degli algoritmi genetici usando il patrimonio di ciascuna regola come valutazione della fitness della stessa (che è appunto la misura della capacità di adattamento di ciascun membro di una popolazione che si sta evolvendo). E' questa l'unica attività che non è sincrona con i messaggi.

Come ultima azione, prima di ricominciare con la selezione di nuovi messaggi dall'ambiente, vengono cancellati tutti i messaggi dalla message list.

3.4 I parametri interni del sistema

In un CS viene definita una seconda tipologia di parametri che servono a regolare il meccanismo interno di selezione e generazione delle regole. Questi parametri riguardano specificamente, la distribuzione dei crediti (attraverso i parametri della *bidTax* e della *lifeTax)* ed i coefficienti di determinazione dell'offerta effettuata, da ciascuna regola, nell'asta per ottenere il diritto di immettere il proprio

messaggio nella lista.

Possiamo considerare i CS come sistemi che evolvono due tipi di apprendimento distinto: uno per rispondere agli stimoli dell'ambiente e uno per scovare il migliore adattamento all'ambiente. Possiamo indicarli il primo come di livello superficiale e il secondo uno più profondo. Ambedue i sistemi sono guidati dlla variazione del valore del patrimonio di ciascun elemento. Il primo permette di assegnare possibilità di azione proporzionali ai risultati, ottenuti da ciascuna regola, e potrebbe essere interpretato come selezione nell'ambito delle soluzioni note; esso viene realizzato per il tramite dell'asta e del meccanismo di gestione dei trasferimenti. Il secondo è ottenuto attraverso le azioni di riproduzione, incrocio e mutazione, affidate all'algoritmo genetico; esso risulta, chiaramente, influenzato dai trasferimenti dall'ambiente e fra le regole ed ha come scopo la ricerca di soluzioni nuove e ancora più efficienti. Poiché l'apprendimento profondo si basa sulle risultanze di quello a breve termine, è importante che possa agire su una popolazione sufficientemente sperimentata, il cui patrimonio abbia subito un congruo numero di modificazioni in forza dell'agire dei singoli: l'algoritmo genetico deve essere attivato con frequenza tale da permettere l'accumulazione di patrimoni.

Poiché il CS deve rispondere con un'azione agli stimoli ambientali, è necessario, almeno nelle prime fasi di evoluzione del sistema, che vi siano molte regole contenenti un certo numero di wildcards #. La funzione fondamentale di regole poco specifiche, cioè contenenti molte wildcard, è quella di assicurare i *match* e, conseguentemente, di permettere al CS di fornire comunque una risposta anche se imprecisa. Però, via via che vengono prodotte regole più specifiche, queste devono essere favorite nell'azione e, conseguentemente, nella riproduzione. Un simile effetto può essere ottenuto penalizzando, nella conduzione dell'asta, le regole con bassa specificità.

L'idea di base è che le regole generiche debbano intervenire solo in mancanza di altre, più specifiche, atte ad agire a fronte della particolare condizione ambientale; esprimendo la specificità in termini relativi si ha:

s = numero di posizioni uguali a {0,1} / numero di posizioni nella stringa .

I valori di O (offerta di una regola) e O' (offerta modificata da un valore casuale a distribuzione nota) possono essere calcolati come:

$O(t) = F(t) * s * \mathbf{a}$.
$O'(t) = F(t) * s * \mathbf{a} + N(\mathbf{s})$.

I valori delle offerte non dipendono più solamente dalla forza ma anche dalla specificità della regola. Le regole molto specifiche avrebbero valori di s prossimi all'unità, mentre quelle più generali avrebbero valori tendenti a zero; le offerte delle regole generali risulterebbero sistematicamente inferiori, con l'effetto di limitarne l'applicazione ai soli casi di effettiva necessità. L'introduzione del valore di specificità, nel calcolo dell'offerta per l'asta, potrebbe simulare la naturale diffidenza per soluzioni troppo generiche che, molto probabilmente, sarebbero poco utili per il raggiungimento di risultati ottimali.

L'evoluzione di regole specifiche potrebbe, inoltre, essere favorita dall'imposizione di una tassa sulle offerte in asta, *bidTax*; il suo ammontare dovrebbe essere proporzionale al patrimonio della regola, senza riferimento alla specificità. Una simile tassa impoverirebbe ulteriormente le regole generali che, in presenza di antagonisti maggiormente specializzati, effettuerebbero comunque una offerta e pagherebbero la *bidTax*, ma avrebbero scarsa probabilità di recuperare la spesa perché difficilmente potrebbero vincere l'asta. Si noti che l'effetto risulterebbe, invece, annullato in assenza di regole specifiche.

Gli individui inutili permangono nel sistema fino a quando la staticità del loro patrimonio non comporti una povertà, relativa, sufficientemente marcata da provocarne l'estinzione. Queste regole, non risultando mai attivate, non subiscono gli effetti della *bidTax*: occorre, allora, introdurre una *lifeTax*, cioè una imposta prelevata, ad ogni ciclo, su tutti i patrimoni, per accelerare l'impoverimento e la scomparsa delle regole inutili. Mantenendo un'aliquota sufficientemente bassa, le regole attive possono facilmente recuperare l'ammontare della tassa con i compensi ricevuti, mentre quelle che non vengono mai attivate perdono, ad ogni ciclo, una parte del proprio patrimonio, divenendo buone candidate per l'estinzione durante l'azione dell'algoritmo genetico.

Ad ogni ciclo vengono dunque riscossi due tipi di tributi:

- *bidTax*, a carico di tutte le regole che partecipano all'asta,
- *lifeTax*, cui è soggetta l'intera popolazione.

Scopo della prima imposta è quello di rappresentare un costo di partecipazione all'asta, onde accelerare il decadimento del patrimonio delle regole sbagliate; la seconda serve a far decadere quelle regole che, possedendo una condizione che non trova riscontro nelle configurazioni ambientali, non vengono mai applicate e sono, perciò, inutili.

Capitolo 4

UNO STORICO ESPERIMENTO

4.1 Descrizione dell'esperimento originale

Come si è visto nel capitolo 2, situazioni riconducibili al "Dilemma del Prigioniero" sono molto diffuse, dai rapporti interpersonali alle relazioni internazionali; sarebbe perciò molto utile arrivare a determinare una strategia ottimale per situazioni di questo tipo. La strategia che si sceglie dipende però da quella che adotterà l'altro giocatore, dalle probabilità che si assegnano alle scelte dell'avversario; potrebbe benissimo anche dipendere da che cosa la nostra controparte prevede che noi si faccia.

Oltre alla simulazione mediante computer, altri filoni di ricerca sono attivi per cercare di approfondire la conoscenza di questo gioco. Purtroppo nessuno di questi riesce a svelare come giocare efficacemente.

Le ricerche sperimentali condotte studiando l'interazione tra persone all'interno di gruppi chiusi risentono del fatto che, per lo più, le persone coinvolte si trovano per la prima volta alle prese con un gioco formalizzato; la loro valutazione delle sottigliezze strategiche è di conseguenza limitata.

Altro importante banco di prova sperimentale è l'uso come piattaforma concettuale di modellizzazione di importanti fenomeni sociali quali la corsa agli armamenti, la concorrenza in regime di oligopolio o i problemi relativi all'azione collettiva per la produzione di un bene comune. In queste ricerche applicate le scelte dei gruppi elitari di esperti in materia di politica e di economia sono state sì oggetto di studio, ma le risultanze pubblicate non sono state di grande aiuto, se non altro a causa del ritmo piuttosto lento delle interazioni di più alto livello e della difficoltà di seguire da vicino la mutevolezza delle circostanze.

Si è sviluppato infine un terzo filone di ricerca che, spingendosi oltre le questioni empiriche di laboratorio, sfrutta maggiormente il gioco astratto per analizzare gli aspetti di alcuni temi strategici di fondo, quali il significato della

razionalità, le scelte che si ripercuotono sugli altri e la cooperazione in assenza di un'imposizione d'autorità.

Per approfondire la nostra conoscenza su come determinare la scelta più efficace da compiere nella forma iterativa del "Dilemma del Prigioniero" c'era dunque bisogno di una impostazione diversa. Innanzitutto bisognava sfruttare la conoscenza di persone che avessero studiato le possibilità strategiche inerenti a un ambito di gioco a somma diversa da zero, cioè in una situazione in cui gli interessi dei soggetti in parte si scontrino in parte coincidano.

Infatti nel gioco ripetuto due sono i fatti importanti da tenere in considerazione. Il primo è che quanto è da ritenersi efficace dipende non soltanto dalle caratteristiche di una data strategia, bensì anche dalla natura delle altre strategie con le quali deve interagire. Il secondo, che discende direttamente dal primo, è che una strategia deve essere capace di tenere conto, in ogni momento, dell'andamento progressivo dell'interazione.

Un torneo informatizzato per lo studio della scelta più efficace nel gioco del "Dilemma del Prigioniero" ripetuto può superare le limitazioni di studio viste precedentemente e venne effettivamente organizzato da Axelrod; i risultati sono stati pubblicati in Axelrod (1980a) e in Axelrod (1980b).

Il torneo prevedeva che ogni partecipante scrivesse un programma per accogliere una regola e scegliere, ogni mossa, tra la decisione di cooperare e la decisione contraria. Il programma aveva a disposizione la storia aggiornata dell'andamento del gioco che poteva quindi essere sfruttata ai fini della scelta.

In prima istanza i partecipanti furono reclutati tra quanti avevano una conoscenza del "dilemma" e si è potuto pertanto assicurare loro che avrebbero dovuto incontrarsi con norme decisorie corrispondenti a strategie di partecipanti già esperti. Questa forma di reclutamento ha garantito la presenza al torneo di partecipanti di buon livello. Alcuni professionisti della teoria dei giochi furono invitati a inviare le loro soluzioni strategiche per partecipare al torneo informatizzato strutturato come un torneo all'italiana (nel senso che ogni concorrente si incontra con ciascuno degli altri), comprendente anche un programma che collabora o tradisce con andamento casuale a parità di probabilità (chiamato RANDOM).

Ogni partita consisteva esattamente in 200 mosse. La matrice dei punteggi era la seguente:

	COOPERAZIONE	DEFEZIONE
COOPERAZIONE	R=3,R=3 Premio per la mutua cooperazione	S=0,T=5 (S) ricompensa del babbeo (T) tentazione a defezionare
DEFEZIONE	T=5,S=0 (T) tentazione a defezionare (S) ricompensa del babbeo	P=1,P=1 Penalità per la mutua defezione

La matrice assegna 3 punti ad entrambe i giocatori per la cooperazione reciproca e 1 punto per la defezione reciproca. Se poi un giocatore defeziona mentre

l'altro coopera, il "traditore" riceve 5 punti (T per temptation) mentre chi ha cooperato riceve 0 punti (S per sucker's pay, la ricompensa del babbeo).

Le quattordici strategie presentate dai concorrenti risalivano a cinque discipline diverse: psicologia, economia politica, scienze politiche, matematica e sociologia.

La vittoria è andata a TIT FOR TAT, strategia presentata dal professor Anatol Rapoport, psicologo dell'università di Toronto. Questa strategia inizia con una scelta cooperativa per proseguire successivamente con una serie di mosse ciascuna delle quali è esattamente identica a quella operata immediatamente prima dall'altro giocatore. Questa regola è probabilmente la più diffusa e più dibattuta in materia di "Dilemma del Prigioniero": facile da comprendere e da programmare è nota per suscitare un discreto grado di collaborazione nel gioco tra soggetti umani. (Oskamp 1971, Wilson 1971).

Già si sapeva che TIT FOR TAT era un concorrente potente e temibile poiché si era piazzata al primo e al secondo posto in due incontri preliminari al torneo vero e proprio. Tutti questi dati erano noti agli autori dei programmi partecipanti al torneo in quanto a tutti erano state inviate copie del resoconto degli incontri preliminari.

Non deve quindi sorprendere che molti concorrenti abbiano sfruttato, cercando di migliorarlo, il medesimo principio.

Il fatto imprevisto è che nessuno dei programmi più complessi presentati in concorso, sia riuscito ad eguagliare le prestazioni del semplice programma originario, il TIT FOR TAT. In un gioco a 200 mosse un utile riferimento di eccellente prestazione si colloca sui 600 punti, cioè l'equivalente del punteggio conseguito da un giocatore quando entrambe le parti cooperino sempre. Un riferimento di pessima prestazione può essere 200 punti che equivalgono al punteggio conseguito quando entrambe i giocatori non cooperino mai. La maggior parte dei punteggi ottenuti nel torneo oscillavano tra 200 e 600 punti

Nonostante il fatto che nessun tentativo di affinare le regole decisorie costituisse un miglioramento rispetto a TIT FOR TAT, è stato possibile individuare (ex-post e per quello specifico ambito di gara) alcune strategie che avrebbero potuto fare notevolmente meglio della prima classificata TIT FOR TAT. La presenza di queste regole decisorie dovrebbe mettere in guardia contro la facile convinzione che la migliore strategia sia necessariamente "occhio per occhio, dente per dente": esistono altre strategie che, se si fossero presentate nella prima gara, avrebbero potuto vincere.

Una di queste è TIT FOR TWO TATS, ovvero una versione più clemente di TIT FOR TAT che defeziona solamente dopo aver ricevuto due tradimenti di fila. Una seconda regola che avrebbe potuto vincere era una leggera variante della strategia effettivamente presentata DOWNING. Questa strategia è ritorsiva come Tit for Tat; attiva però l'azione di ritorsione in base al comportamento medio e non semplicementein base all'ultima azione dell'avversario. Al contrario di Tit for Tat, inizia l'incontro defezionando; se DOWNING fosse partita con l'ipotesi ottimistica di incontrare una controparte reattiva (invece che non reattiva) avrebbe potuto vincere con largo anticipo.

L’analisi dei risultati di questo primo torneo ha portato alla conclusione che le strategie presentate fossero troppo competitive per poter riuscire a vincere.

In primo luogo molte prevedevano la defezione, in assenza di provocazione, fin dal primo turno, caratteristica che alla lunga si è rivelata fortemente punitiva. In secondo luogo la quantità di clemenza deve essere molto più alta di quella offerta dalla maggior parte delle strategie partecipanti. In terzo luogo un orientamento ottimistico sulla reattività/sensibilità dei partecipanti conduce a migliori risultati attraverso la facilitazione della collaborazione.

L’efficacia di una determinata strategia non dipende comunque solamente dalle sue caratteristiche intrinseche, ma anche dalla natura delle altre strategie con le quali si trova ad interagire, motivo per cui i risultati di un unico torneo non sono stati considerati definitivi: ed è per questo che si è proceduto ad organizzare la seconda tornata di gara.

Agli iscritti al secondo torneo (i professionisti partecipanti al primo, più un certo numero di dilettanti) è stata comunicata l’analisi dettagliata della prima tornata, contenente la trattazione approfondita delle strategie fuori concorso che avrebbero potuto vincere o, quanto meno, piazzarsi bene nel primo torneo.

Quindi i concorrenti erano a conoscenza, non soltanto dell’esito del primo torneo, bensì anche dei concetti usati nell’analisi dei risultati nonché dei trabocchetti strategici individuati nell’occasione. L’altro dato era che, a questo punto, tutti sapevano che anche gli altri erano a conoscenza delle strategie ideali.

Inoltre il secondo torneo rappresentava un netto miglioramento per la dimensione stessa del torneo. Con una risposta più vasta del previsto, si iscrissero complessivamente 62 concorrenti da 6 paesi diversi. Il secondo torneo offriva la possibilità di verificare la validità delle considerazioni sviluppate sui risultati del primo torneo poiché tutti i partecipanti si potevano giovare dell’esperienza maturata nel corso della prima tornata.

Sta di fatto che persone diverse hanno tratto conclusioni diverse, così che un elemento particolarmente interessante del secondo round è stato proprio il modo in cui hanno interagito strategie sulle diverse conclusioni dedotte dai risultati del primo torneo.

TIT FOR TAT, che era stato il programma più semplice in assoluto tra quelli presentati nella prima gara e che aveva facilmente vinto, è risultato essere il più semplice ed il vincitore anche della seconda gara. Questa strategia vincente era nota a tutti i partecipanti (avendo tutti ricevuto il resoconto della prima gara). La strategia era ben nota e documentata anche al di fuori dell’ambito del torneo informatizzato, e la documentazione consegnata riguardante la prima serie di incontri spiegava persino alcuni motivi del successo della strategia vincente. In particolare veniva segnalata la caratteristica di “bontà” (ovvero di non essere mai la prima a defezionare) e la “clemenza” (ovvero la propensione a cooperare dopo un’unica defezione della parte avversa). Nonostante questa strategia fosse ben nota è riuscita ugualmente a vincere.

Il secondo girone si svolse secondo le stesse modalità del primo, con un’unica eccezione. La durata di ciascun incontro non era più predefinita in 200 mosse ma determinata probabilisticamente con una probabilità di 0,00346 che la partita terminasse ad una certa mossa. In questo modo nessun giocatore poteva sapere

quando si sarebbe verificata esattamente l'ultima mossa e, di conseguenza, non poteva predisporre "colpi di fine - mano", ovvero defezioni agli ultimi turni predisposti sapendo che l'incontro sarebbe finito entro uno o pochi cicli.

Anche nella seconda tornata non è emersa alcuna regolarità statistica che permettesse di correlare significativamente le prestazioni delle varie strategie con le rispettive caratteristiche. La professione degli autori, la provenienza, il linguaggio usato (BASIC che presumibilmente indica un utilizzatore di personal computer o FORTRAN che indica la possibilità di accedere a computer più grandi) non presentano correlazioni di sorta con i risultati conseguiti. Nemmeno è stato possibile notare differenze che si potessero far risalire alla lunghezza (e quindi alla complessità) del programma usato.

L'individuazione delle ragioni del successo non è stata semplice poiché la matrice dei punteggio è risultata grandissima dovendo tenere conto di oltre un milione di mosse.

Come nel primo torneo si è potuto constatare che le strategie "buone" (ovvero strategie che non decidono mai di defezionare per prime) hanno ottenuto risultati nettamente superiori alle altre.

Inoltre nel secondo torneo sono state presentate diverse strategie che deliberatamente si servivano di un numero controllato di defezioni per vedere di volta in volta quale fosse il momento migliore per piazzare una defezione ed incassarne il punteggio. In larga misura la buona posizione in classifica delle strategie "buone" è stata determinata dalla capacità di cavarsela con sfidanti di questo tipo.

Come esempio di simili strategie Axelrod (1985) ne cita due.

> *TESTER, programma presentato da David Gladstein, si è piazzato al 47° posto. Progettato per l'individuazione delle difese più morbide, è anche dispostissimo a battere in ritirata quando l'altro giocatore dimostri di non essere affatto intenzionato a lasciarsi sfruttare. Si tratta di una strategia insolita, in quanto, se defeziona alla primissima mossa, lo fa soltanto per sondare, per verificare (da cui il nome) la reazione dell'altra parte: se si imbatte in qualsiasi momento in una defezione, evita lo scontro con mille scuse e da quel momento in poi ribatte colpo su colpo. Altrimenti coopera alla seconda e alla terza mossa, ma da quel momento in poi defeziona a ogni altra mossa. Tester se l'è cavata benissimo nello sfruttamento di diverse strategie fuori concorso che si sarebbero senz'altro piazzate nell'ambiente del primo girone. TIT FOR TWO TATS, per esempio, defeziona, come abbiamo visto, soltanto quando l'altro giocatore abbia defezionato alle due mosse immediatamente precedenti, mentre TESTER non defeziona mai due volte di seguito. Va a finire che TIT FOR TWO TATS, collaborando sempre con TESTER, esce dall'incontro malamente sfruttato in cambio della propria generosità. Si noti che, dal canto suo, TESTER non se l'è cavata molto bene in gara, anche se è servito a castigare con punteggi bassi un buon numero di strategie facilone.*

Nel modello qui presentato (che verrà più dettagliatamente descritto nei paragrafi successivi), questa strategia è identificata con il behaviour N.20, codificato come metodo della classe RuleMaster identificato come performBehaviour20. Anche

nel modello qui presentato i risultati di questa strategia, sono stati identici, essendosi piazzato tra il 45° e il 50° posto in classifica nelle varie prove effettuate.

L'altra strategia di questo genere è TRANQUILIZER presentato da Craig Feathers (nel modello ad agenti il behaviour 4). Questa strategia prevede la cooperazione ma se il numero delle defezioni dell'avversario supera certe soglie inizia a defezionare; almeno fino a quando conserva una vincita media di 2,25 punti per mossa, non defeziona mai due volte di seguito. Comunque non defeziona mai più del 25% dei casi. Cerca insomma di approfittare senza esagerare.

Come si è già detto prima, ai partecipanti al secondo torneo sono state fornite tutte le informazioni ed i risultati relativi al primo torneo. Ovviamente quanto appreso precedentemente ha influenzato le proposte dei partecipanti al secondo torneo Axelrod (1985).

> l rendiconto del primo girone del Torneo informatizzato del "Dilemma del Prigioniero" era giunto alla conclusione che si aveva tutto da guadagnare, non soltanto nell'essere buoni, ma anche nell'essere clementi. E i concorrenti avevano capito che nell'ambiente della seconda tornata certe strategie clementi come TIT FOR TWO TATS e il DOWNING MODIFICATO si sarebbero piazzate anche meglio di TIT FOR TAT.
> A quanto pare si è verificata dunque un'interazione molto interessante tra quanti avevano tratto dalla prima contesa un certo insegnamento e quanti ne avevano tratto un altro. Primo insegnamento: "sii buono e clemente". Secondo insegnamento "Se gli altri intendono essere buoni e clementi, vale la pena tentare di sfruttarli" Così coloro i quali si erano attenuti al primo insegnamento hanno dovuto patire la sconfitta a opera di coloro che avevano preferito il secondo, tanto che programmi come TRANQUILIZER e TESTER non hanno avuto nessuna difficoltà a sfruttare le strategie più bonaccione. Non per questo però i discepoli del secondo insegnamento se la sono cavata molto bene, in quanto, cercando di sfruttare i punti morbidi delle altre strategie, finiscono per farsi castigare quanto basta per rendere improduttivo l'intero incontro per entrambi i giocatori e, comunque, non tanto premiante quanto potrebbe esserlo in caso di reciproca cooperazione…Del resto nessuna delle strategie che tentavano di applicare il suggerimento opportunistico del secondo insegnamento si è avvicinata alla parte superiore della classifica.

Le caratteristiche che sembra avere TIT FOR TAT e che lo rendono così efficiente in questo ambiente di torneo sono state riassunte da Axelrod in tre punti principali:

- "bontà", ovvero rinuncia a defezionare per primi
- "clemenza", perdonare dopo una defezione isolata
- "essere ritorcente", ovvero rispondere immediatamente a defezioni, qualunque sia stato l'andamento del rapporto fino a quel momento.

L'esame dei risultati del torneo non può evitare la seguente domanda TIT FOR TAT si comporterebbe ugualmente bene anche in ambienti diversi? Questa strategia è sufficientemente robusta?

La costruzione del modello ad agenti indipendenti qui proposta tende a semplificare la ricerca sia delle condizioni di robustezza della strategia allora vincente (TIT FOR TAT) sia la possibile determinazione di strategie più efficienti nelle medesime condizioni ambientali.

4.2 Ristrutturazione ad agenti indipendenti

Il modello ad agenti indipendenti è stato realizzato essenzialmente per poter sfruttare compiutamente le due maggiori potenzialità proprie di questi modelli:

- utilizzo di software standard per conseguire facilmente uno dei requisiti chiave di un esperimento scientifico – la *riproducibilità*
- completa esplorazione delle variabili che determinano le caratteristiche principali del modello.

L'utilità di un modello è tanto più alta quanto più consente di riprodurre agevolmente i risultati conseguiti e quanto più consente variazioni dei parametri del sistema in modo semplice e senza dover alterare la struttura del modello, rendendo possibile isolare gli effetti sotto esperimento.

Il principale elemento di evoluzione rispetto all'esperimento originario è però costituito dalla possibilità di sondare l'emergenza di comportamenti non previsti attraverso l'introduzione, nello stesso ambiente, di agenti con comportamento di tipo evolutivo basato su Classifier Systems. Nello stesso modello è quindi possibile confrontare comportamenti che obbediscono a regole precise e predeterminate con agenti che, al contrario, sono strutturati per sviluppare regole adattative e che possono sviluppare strategie di gioco non prevedibili a priori.

Anche un modello creato utilizzando:

- un codice di programmazione ben diffuso nella comunità scientifica come l'Objective C,
- biblioteche di oggetti e di funzioni standard
- un insieme di protocolli che realizzano un ambiente standard e ben diffuso di sperimentazione (SWARM)

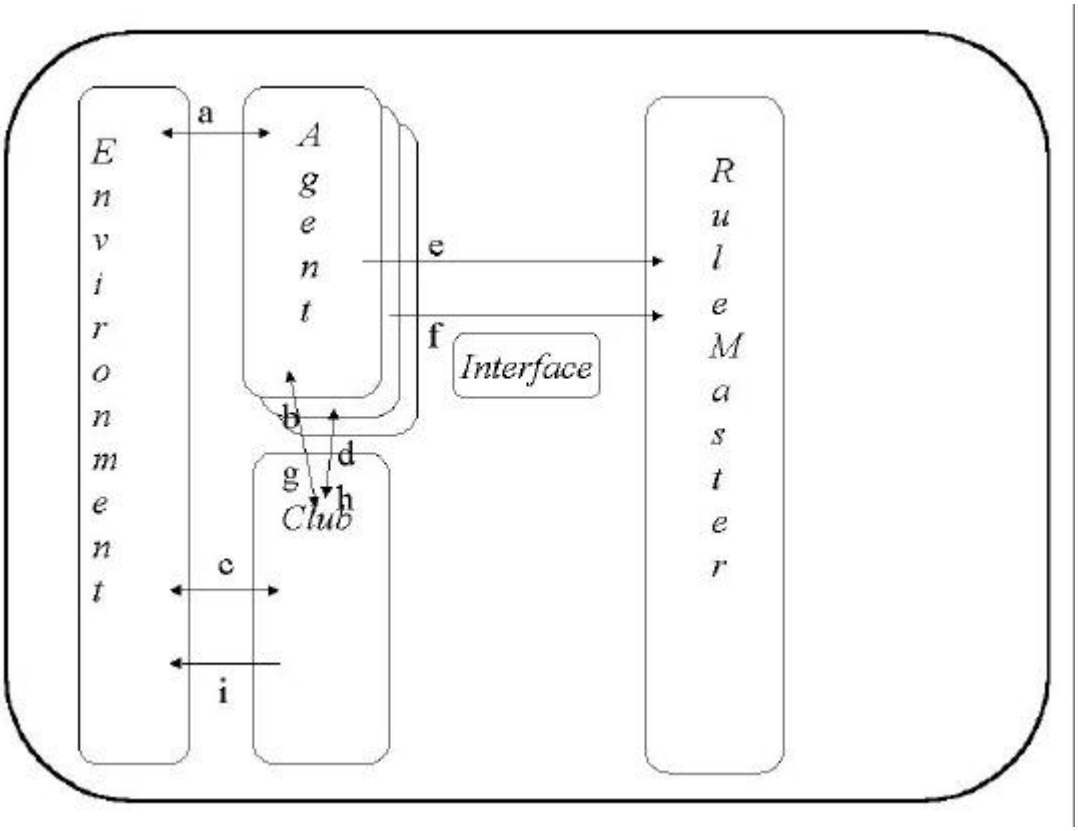
potrebbe non garantire la necessaria replicabilità degli esperimenti portati a termine.

Può infatti risultare molto difficile la comprensione ed il riutilizzo di un modello da parte di chiunque (al limite anche dall'autore stesso purché intercorra un po' di tempo dall'ultimo utilizzo) poiché software ed ambienti standard di sperimentazione non garantiscono di per sé la comprensibilità del modello in questione.

L'utilizzo dello schema Environment Rules Agents (ERA) (visto precedentemente) consente di strutturare i modelli in costruzione secondo uno schema standard che ne semplifica la comprensione e, di conseguenza, l'utilizzo e la replicabilità.

Sulla base dello schema ERA verrà qui di seguito descritto il modello realizzato. Questo modello ha consentito di:

- riprodurre il secondo torneo dello storico esperimento di Axelrod, passo necessario per ottenere la certezza di essere nello stesso ambito sperimentale anche se ne è stata completamente modificata la struttura.
- valutare gli effetti del "rumore" introdotto nel modello
- valutare la possibilità che emergano strategie ottimali di comportamento non previste originariamente attraverso la sostituzione di uno o più agenti a comportamento predeterminato con altri a comportamento adattativo.



Per quanto riguarda le altre possibilità di indagine che si rendono disponibili attraverso l'utilizzo di un modello ad agenti indipendenti, un punto certo importante, anche se non approfondito in questo lavoro, è la possibilità di verificare, per esperimento, una congettura di evoluzione della cooperazione avanzata da Axelrod (1985) commentando i risultati del torneo.

> Qui abbiamo in nuce un'importante teoria: l'evoluzione della cooperazione richiede che i soggetti abbiano una probabilità sufficientemente grande di incontrarsi almeno una seconda volta così da avere una certa posta in gioco per l'interazione futura. Se ciò è vero, la cooperazione può evolvere in tre stadi successivi.
>
> 1) L'inizio della vicenda è che la cooperazione può essere innescata anche in un mondo di defezione incondizionata. Tale sviluppo non può verificarsi se viene sperimentato soltanto da alcuni soggetti sparsi i quali non abbiano praticamente alcuna occasione di interagire tra loro, mentre la cooperazione può benissimo evolvere da piccoli gruppi di soggetti i quali fondino la cooperazione sul principio della reciprocità e abbiano almeno un'occasione minima di interagire tra loro.
> 2) La parte centrale della storia vede una strategia che, fondata sulla reciprocità, possa prosperare in un mondo in cui siano sperimentate molte strategie di tipo diverso.
> 3) La conclusione della storia è che la cooperazione, una volta affermatasi sulla base della reciprocità, riesce a proteggersi da sola dall'invasione di strategie meno competitive. A questo punto gli ingranaggi dell'evoluzione sociale hanno trovato il loro motorino.

E' evidente che il modello ad agenti indipendenti in grado di muoversi in un

territorio, come questo proposto, sia perfettamente in grado di simulare le condizioni citate di “soggetti sparsi” o “evoluzione da piccolo gruppi”. Simulazione che certo non è agevole implementare in un torneo in FORTRAN come quello attivato nello “storico esperimento”.

La superficie su cui si muovono gli agenti, anche se graficamente rappresentata da un quadrato, è in realtà un toroide. Quindi gli agenti possono muoversi indefinitivamente verso destra/sinistra o sopra/sotto poiché se si esce dal quadrato da sinistra si rientra da destra ed analogamente per le direzioni sopra/sotto.

4.2.1 I parametri di input

L’ambiente è realizzato in modo standard, attraverso la codifica di un’istanza di ObserverSwarm e di ModelSwarm. Oltre a gestire l’ambiente di sperimentazione, questi oggetti si incaricano anche di acquisire diversi parametri di definizione del modello.

(dal metodo createBegin di ModelSwarm)

```
// Now fill in various simulation parameters with default values.

   obj->worldXSize               = 50;
   obj->worldYSize               = 50;
   obj->times                    = 200;
   obj->sensitivity              = 5;
   obj->distance                 = 2;
   obj->numberOfInteractiveAgents = 4;
   obj->numberOfReactiveAgents    = 63;
   obj->behaviour                = 0;
   obj->noise                    = 0.00;
// the number of single matchs for a round robin tournament between
// N=numberOfReactiveAgents (match and return match) is:
// N*(N-1)
   obj->maxNumberOfMatches       = 200;
```

I valori di worldXSize e di worldYSize servono per definire le dimensioni del “mondo” in cui si spostano gli Agenti. Oltre che per la grafica, la determinazione di queste dimensioni è importante poiché, in caso di interazioni di tipo “geografico” (vale a dire che la possibilità di incontro tra due agenti dipende dalla loro reciproca posizione) definisce il denominatore del rapporto NUMERO AGENTI/SPAZIO che rappresenta la densità di agenti, ovvero la loro probabilità di interagire.

Il parametro times consente di determinare il numero di volte che viene ripetuto il “Dilemma del Prigioniero” quando due agenti si incontrano.

Sensitivity serve per controllare il movimento degli agenti: per gli esperimenti condotti fin’ora e qui presentati non è stato in pratica utilizzato.

Il numero di agenti interattivi e di agenti reattivi, come è facilmente intuibile, permette di determinare il numero degli agenti appartenenti a queste due categorie.

Perché definire due tipi di agenti? L'esperimento originario, descritto nel primo paragrafo di questo capitolo non prevedeva questa distinzione, e le strategie descritte sono state ridefinite come metodi (da -performBehaviour0 a -performBehaviour62 della classe RuleMaster) e quindi disponibili per tutti gli agenti. La distinzione non riguarda quindi le strategie messe in atto, ma solamente la "forma" del torneo.

Se il numberOfInteractiveAgents è zero il torneo si svolge solamente tra gli agenti reattivi che realizzano, nei primi esperimenti, esattamente le 62 (più RANDOM) strategie dell'esperimento originale.

Se il numberOfInteractiveAgents è maggiore di zero il torneo si svolge con gli agenti interattivi che incontrano i reattivi. Ci si garantisce, cioè, che la strategia degli agenti interattivi venga confrontata con tutte quelle dei reattivi che assumono un comportamento passivo (solamente reattivo) senza che avvengano incontri tra agenti reattivi. Questo essenzialmente per due motivi:

- per semplificare le fasi di test (impostando il valore di behaviour si ottiene di avere il numero imputato di agenti reattivi che si comportano tutti secondo lo stesso metodo e di poterlo testare negli incontro con gli agenti interattivi)
- per consentire anche una struttura il torneo differenziando tra l'insieme delle strategie originali che fungono da "ambiente" ed alcune strategie in valutazione o in evoluzione, secondo uno schema successivo presentato sempre da Axelrod (Axelrod The Complexity of Cooperation Princenton Paperbacks 1997) e reperibile anche in http://pscs.physics.lsa.umich.edu/Software/CC/CC2/TourExec-Doc.html

Behaviour è stato utilizzato principalmente nelle fasi di test e consente di controllare il comportamento di uno specifico metodo definito per gli agenti interattivi testandolo contro le strategie degli agenti reattivi.

Il parametro maxNumberOfMatches viene usato per definire il numero complessivo di incontri. Se il torneo si svolge all'italiana (round robin, tutti gli agenti incontrano tutti) il parametro non viene considerato. Se però il torneo viene svolto su base "geografica", ovvero gli incontri vengono definiti tra agenti che si trovano alla distanza stabilita, occorre aver definito quanti incontri complessivi si intendono svolgere. Il metodo –getOpponentFar della classe Club (metodi e classi del modello sono descritti più avanti in questo capitolo) prima di cercare un avversario per il match, controlla di non aver già raggiunto il numero stabilito di incontri. Questo numero complessivo viene calcolato come

(Numero degli agenti reattivi) X (Numero degli agenti reattivi) X times

ovvero, se ci sono 63 agenti reattivi e ogni incontro prevede 200 ripetizioni del dilemma del prigioniero, si avrà 63*63*200=793.800. Se questo numero è più grande del parametro maxNumberOfMatches si adopera il valore calcolato, altrimenti si utilizza il valore del parametro. Questo per consentire di, eventualmente, continuare il torneo per un numero più grande di incontri.

Distance è il parametro che serve a regolare a quale distanza dall'agente cercare (nel mondo) un altro agente con cui iniziare un match. Se questo parametro è impostato a zero serve a indicare che si intende realizzare un torneo round robin, torneo in cui la ricerca dell'avversario è fatta per liste e non per localizzazione geografica.

Noise, che deve essere un numero compreso tra 0 e 1, indica la percentuale di volte in cui l'azione decisa da una certa strategia viene invertita in modo casuale. In altre parole, impostando questo valore a 0.12, in media il 12% delle volte capiterà che l'azione decisa da una strategia venga invertita, introducendo in questo modo un "rumore sul canale di comunicazione tra gli agenti". Se il parametro viene tenuto a zero, ovviamente non c'è rumore di canale.

Utilizzando in modo combinato i parametri qui descritti è possibile realizzare diversi schemi di esperimento utilizzando sempre il medesimo modello ad agenti indipendenti.

- Distance=0/numberOfInteractiveAgents=0/numberOfReactiveAgents=63: in questo caso si replica fedelmente l'esperimento originale descritto nel primo paragrafo; un torneo round robin tra 63 agenti a comportamento predefinito senza introdurre altro "rumore" che non sia il comportamento RANDOM di uno degli agenti (esattamente come nell'esperimento descritto)
- Distance=0/numberOfInteractiveAgents=4/numberOfReactiveAgents=63: in questo caso si attua un torneo round robin tra le 4 strategie definite per gli agenti interattivi che incontrano tutte le strategie degli agenti reattivi; si effettuano perciò 4*63 incontri (della "durata" definita da times)
- Distance=1/numberOfInteractiveAgents=0/numberOfReactiveAgents=63: poiché il valore di Distance non è zero il torneo viene condotto scegliendo gli avversari (solamente tra gli agenti reattivi) che sono alla distanza definita dall'agente attivo in quell'istante. Non viene pertanto più garantito che tutti incontrino tutti, ma (come detto precedentemente) questa modalità di torneo può essere utile per esperimenti che coinvolgono considerazioni geografiche
- Distance=1/numberOfInteractiveAgents=N/numberOfReactiveAgents=63: poiché il valore di Distance non è zero il torneo non è round robin; per ogni agente interattivo viene cercato un agente reattivo a distanza minore del parametro specificato Non viene più garantito che tutti gli agenti interattivi incontrino tutti gli agenti reattivi.
- Noise diverso da zero; viene introdotta una certa percentuale di "rumore di canale", indipendentemente da quale dei quattro tipi di torneo prima descritti si sta svolgendo
- NumberOfReactiveAgents > 63; vengono aumentati il numero di agenti a comportamento casuale, incrementando il "rumore ambientale", anche in questo caso indipendentemente da quale dei quattro tipi di torneo si sta svolgendo.

4.2.2 Gli oggetti Agent

Su una superficie toroidale, tale per cui i confini in ciascuna direzione toccano quelli della direzione opposta, si muove una collettività di agenti, tutti istanze della stessa classe Agent. Come si è visto prima sono possibili due tipi diversi: agenti interattivi ed reattivi. Si differenziano però solamente nel fatto che, se sono presenti agenti interattivi, solamente a questi è assegnata la possibilità di prendere l'iniziativa per un incontro.

La schedulazione delle azioni è quanto mai semplice: per ciascun agente del

modello (e quindi compreso nella agent list) viene richiesta l'esecuzione di due metodi:

- move
- doIteraction.

In sostanza, ogni agente, come prima azione modifica la propria posizione. Poiché questo metodo è completamente slegato da come si sta svolgendo il torneo, è possibile definire anche strategie di movimento. Vale a dire prendere decisioni se muoversi, in quale direzione muoversi e per quanto muoversi. Nelle prime fasi della sperimentazione questa opzione è stata tralasciata poiché per approfondire gli effetti del "rumore" e per l'evoluzione di nuove strategie non è necessaria.

Effettuata la fase di movimento viene attivato il secondo metodo che effettua le seguenti azioni:

- chiede a Club se c'è un altro agente con cui iniziare un incontro basato appunto sulla ripetizione del "Dilemma del Prigioniero"; se Club risponde che non c'è il ciclo di schedulazione per quell'agente si chiude. In caso di torneo round robin, Club cerca l'agente da opporre scandendo la lista degli agenti. In caso di torneo geografico lo cerca interrogando il "mondo" per sapere se ci sono altri agenti nelle vicinanze.
- Se c'è incontro l'agente inizializza le variabili che verranno utilizzate nell'incontro e chiede a Club di fare altrettanto per l'agente avversario. NON ESISTE COMUNICAZIONE DIRETTA TRA GLI AGENTI, secondo gli standard prefigurati nello schema ERA.
- Per un numero di volte pari al parametro times, dall'agente viene chiesto al RuleMaster quale azione compiere, ovvero se cooperare o defezionare. Analogamente viene chiesto a Club di chiedere all'agente avversario quale azione intende compiere.
- Definite le due azioni, si chiede a Club di applicare la matrice dei punteggi basandosi sulle azioni decise e di scrivere su file il risultato parziale.

Come si può dedurre da questa descrizione, non esiste uno modo automatico di fermare il modello. Club però, arrivato ad un certo punto smette, di "trovare" agenti avversari ed il modello opera, di fatto, solamente per il movimento degli agenti.

Se il torneo è di tipo round robin Club smette di "trovare" avversari quando ha esaurito tutti i possibili incontri; se è un torneo di tipo geografico quando raggiunge il numero prefissato di incontri come descritto nel paragrafo precedente.

4.2.3 Gli oggetti Interface

Per accentuare la separazione tra gli agenti, che interagiscono con l'ambiente e le componenti di decisione e di evoluzione del comportamento (RuleMaster e RuleMaker) che sono le componenti di applicazione e di evoluzione delle regole, si è introdotto nel modello una classe Interface.

Di questa classe c'è un'istanza per ogni agente, in modo che possa agire, se necessario, da "traduttore" dalla metrica degli agenti a quella del RuleMaster e del RuleMaker.

4.2.4 Gli oggetti Club

La classe Club istanzia un solo oggetto club che ha il compito essenziale di gestire l'incontro tra due agenti. La necessità di questa classe e dell'unica istanza discendono dalle seguenti esigenze:

- nella costruzione di questo modello si è scelto di non far comunicare direttamente gli agenti tra di loro
- poiché ogni agente deve decidere se cooperate o defezionare SENZA SAPERE IN ANTICIPO QUANTO HA DECISO L'AVVERSARIO serve, nel modello, una specie di arbitro / club di incontri per raccogliere in modo "sincrono" le due decisioni
- è necessario avere nel modello un unico punto in cui applicare la matrice dei punteggi; altrimenti qualsiasi modifica del modello sarebbe estremamente onerosa
- avere un unico punto in cui gli agenti si incontrano e scambiano la propria scelta (cooperare o defezionare) consente di introdurre facilmente e di gestire con un unico parametro la quantità di "rumore di canale" che si vuole introdurre nel modello.

L'unicità del punto di incontro consente anche di semplificare la gestione della scrittura su file dei punteggi che rappresentano l'output del modello verso l'esterno e che sono la base delle considerazioni che verranno svolte nei paragrafi seguenti.

4.2.5 Gli oggetti RuleMaster

Anche questa classe istanzia un solo oggetto che racchiude tutte le strategie che erano state presentate nell'esperimento originale più alcune altre che, sebbene non presenti originariamente, sono costantemente citate nella letteratura relativa al "dilemma de prigioniero" ripetuto.

Sostanzialmente realizza una sola funzione: chiamato da un agente, che si qualifica con il proprio identificatore e con il proprio interface deve restituire la decisione di cooperazione o defezione.

Ogni agente ha un proprio valore di behaviour e ad ogni valore corrisponde un metodo interno che implementa una strategia NON evolutiva (nel senso che la strategie è prefissata e non cambia con l'evolversi del modello). La scelta di non legare direttamente il numero dell'agente ad una strategia, ma di farlo attraverso i parametri appena descritti, consente di popolare di agenti che, al limite, possono tutti decidere secondo la stessa strategia. Oltre ad essere utilissimo in fase di test, permette anche di gestire in modo dinamico il numero di agenti che si comportano in modo RANDOM, incrementando via via il "rumore ambientale" del modello.

4.3 L'esecuzione dei tornei

Il modello costruito come descritto nel paragrafo precedente ha permesso, dopo aver riprodotto l'esperimento originale condotto da Axelrod, di approfondire lo studio degli effetti del "rumore" nello svolgimento di un torneo basato sulla reiterazione del "Dilemma del Prigioniero".

Si è voluto, negli esperimenti di seguito descritti, indagare due "forme" diverse di errori casuali chiamati, come più dettagliatamente descritto ne cap. 2, "rumore ambientale" e "rumore di canale".

Il parametro "noise" inserito nel modello, come si è visto prima, influenza direttamente la percentuale di "rumore di canale" nel modello; viceversa non esiste nessun parametro specifico per il rumore ambientale che si può aumentare aumentando il numero di agenti che seguono la strategia RANDOM. Ad esempio, introducendo 30 agenti di questo tipo, si ottiene che il 30/(30+63)*100=32,26% delle azioni decise nel modello sia casuale.

Il modello usato per gli esperimenti è veramente sensibile al rumore? I due grafici che seguono sembrano offrire un esempio chiaramente interpretabile degli effetti del rumore. Sono due grafici a dispersione che riportano sull'asse delle X semplicemente i numeri che identificano i 63 agenti del modello. Sull'asse delle Y vi è invece il punteggio totale ottenuto sommando i punti complessivi di tre successivi tornei di 200 interazioni per incontro di due agenti. Ogni agente ha quindi partecipato a 63(incontri)x3(tornei)x200(reiterazioni)=37.800 incontri singoli. Sebbene ogni incontro possa assegnare un massimo 5 punti, ottenere 3 punti per incontro (incontro che determina una doppia cooperazione) come media può essere considerato un ottimo risultato. Ci si può quindi attendere che i punteggi vadano da 37.800 (1 punto in media per ogni incontro) fino a 113.400 (punteggio medio 3) anche se il massimo teorico ottenibile è, in questo caso, 189.000.

Come si può vedere dai grafici, l'introduzione di un 10% di rumore di canale, "confonde" le strategie che non sanno interpretare correttamente la nuova situazione. Come vedremo meglio in seguito, non tutte le strategie sono ugualmente sensibili al rumore, ma è innegabile che le due situazioni siano molto diverse tra loro. Nel modello senza rumore si possono definire, ed in modo netto, due gruppi distinti. Il primo che tendenzialmente coopera e che riesce ad ottenere un punteggio medio per singolo incontro di oltre 2,5 ed un altro composto da strategie nettamente meno efficienti. Con l'introduzione del rumore, non solo la situazione è molto meno nettamente definibile, ma anche il punteggio medio e massimo si avvicinano, "comprimendo" i risultati in un range molto più piccolo.

Qui grafico excel file: grafico4_1

Qui grafico excel file: grafico4_2

Quali sono stati i passi effettuati per studiare questo fenomeno? Come è stato descritto nel paragrafo precedente, il primo passo è stato quello di costruire un modello di simulazione ad agenti indipendenti, secondo lo schema ERA ed usando

SWARM. Questo modello è stato utilizzato per una serie di esperimenti che verranno più dettagliatamente descritti nei paragrafi successivi, ma che possono essere visti come una sequenza logica di approfondimento del problema degli effetti del rumore:

- come prima prova si è ripetuto, fedelmente, l'esperimento di Axelrod; in questo modo, attraverso l'analisi dei risultati, si è potuto confermare la vittoria della strategia Tit for Tat e, di conseguenza, la bontà del modello qui proposto;
- il secondo test analizza l'impatto del "rumore ambientale"
- il terzo approfondisce lo studio degli effetti del "rumore di canale"
- il quarto analizza specificatamente quattro nuove strategie.

Delle quattro nuove strategie proposte nel quarto esperimento, due (che sono delle variazioni di Tit for Tat) servono per mettere in pratica dei suggerimenti avanzati da Axelrod stesso nel commento ai risultati del primo esperimento.

Viene messo in evidenza come alcune strategie (tra cui la strategia vincitrice del torneo, Tit for Tat) possano essere molto sensibili all'"effetto eco". Se due giocatori giocano entrambe la stessa strategia Tit for Tat e uno dei due giocatori sperimenta una defezione da parte dell'altro (causata dal "rumore"), l'eco di questo errore si propaga per tutto il resto dell'incontro.

Axelrod suggerisce (come visto nel capitolo 2) due modi per trattare questo effetto eco. Il primo riguarda chi riceve la defezione ed è non rispondere immediatamente ad una defezione con un'altra, atteggiamento definito di *generosità*; il secondo è a carico di chi ha defezionato per errore e presuppone di non rispondere con una defezione alla conseguente defezione dell'avversario, atteggiamento definito di *pentimento.*

Le altre due strategie introdotte si basano invece su un concetto diverso, già studiato dallo stesso autore di Tit for Tat, Anatol Rapoport e presentato in una pubblicazione del 1965 (Rapoport e Chammah 1965) come strategia Simpleton. La strategia proposta si fonda su questa idea: se il punteggio ottenuto nella mossa precedente è stato alto (ho defezionato mentre il mio avversario collaborava oppure abbiamo collaborato entrambe) mantengo la stessa scelta fatta, altrimenti la cambio.

Qui si sono sperimentate due versioni. La prima, chiamata Pavlov, coopera se entrambe hanno cooperato o defezionato nella mossa precedente. La seconda non è che una versione "generosa" della prima, che coopera anche nel 10% dei casi in cui la prima avrebbe defezionato.

Le varie strategie presenti nel modello (e che sono identificate come metodi della classe RuleMaster) sono perciò le seguenti:

- i metodi da performBehaviour0 a performBehaviour62 sono le 63 strategie originali dell'esperimento di Axelrod. In particolare il metodo performBehaviour23 è la strategia chiamata RANDOM e che, già nell'esperimento originale, aveva il compito di introdurre "rumore" nel modello, come descritto nei paragrafi precedenti; il metodo performBehaviour5 e Tit for Tat.
- Il metodo performBehaviour63 che realizza la strategia di Tit for Tat "generoso"; nel 10% dei casi in cui Tit for Tat avrebbe defezionato , questa strategia decide di cooperare; la scelta dei casi in cui attivare la "generosità" è casuale.

- Il metodo performBehaviour64 che implementa la versione di Tit for Tat con "pentimento"; è qui introdotto uno stato interno. Questa strategia ha tre possibili stati interni: "pentita", "contenta", e "provocata". Inizia nello stato di "contenta" cooperando e rimane in questo stato a meno che sperimenti una defezione unilaterale. Se è stata vittima di una defezione mentre era in uno stato di "contenta", entra nello stato di "provocata" e defeziona fino a quando una cooperazione da parte dell'avversario non ne causa il ritorno allo stato di "contenta". Se è stata lei a defezionare mentre era nello stato di "contenta" (defezione dovuta al "rumore") entra nello stato di "pentita" e coopera. Quando è in questo stato di pentimento, ritorna nello stato di contenta solamente in caso di cooperazione anche dell'avversario.
- il metodo performBehaviour86: è la strategia Pavlov che coopera se entrambe hanno cooperato o tradito alla mossa precedente.
- il metodo performBehaviour66: è la strategia Pavlov in versione "generosa"; la scelta dei casi in cui attivare la "generosità" è casuale.
- i metodi performBehaviour80/81/82/83 sono stati usati in fase di test del modello senza poi entrare nei vari tornei.

L'esecuzione di un torneo round robin tra 63 strategia genera 3969 incontri di centinaia di reiterazioni del dilemma del prigioniero. Il numero dei dati da valutare per ogni torneo (e ogni torneo è ripetuto almeno 3 volte) impone che vengano trattati mediante appositi programmi. Si è scelto di utilizzare il SAS poiché permette sia la raccolta e l'ordinamento dei dati, sia analisi di tipo statistico.

Alla fine di ogni incontro tra due agenti, Club raccoglie il punteggio finale e scrive su file un record con le seguenti informazioni:

- NAg; (non è un dato vero e proprio bensì un identificativo; vuol dire che i dati che seguono si riferiscono all'agente)
- il numero dell'agente ;
- il punteggio ottenuto dall'agente nell'incontro ;
- il punteggio totale fino a quel momento ;
- NOp ; (anche questo è un identificativo; vuol dire che i dati che seguono si riferiscono all'agente opponent)
- numero dell'agente opponent;
- il punteggio ottenuto nell'incontro dall'opponent;
- il punteggio totale fino a quel momento dell'opponent;

Ogni campo è separato dal successivo da un punto e virgola; in questo modo il file di output dei risultati assume un formato facile da trattare con il SAS.

I programmi utilizzati per l'analisi dei dati sono in appendice A e sono stati utilizzati per ottenere:

- dati somma o dati medi delle diverse versioni fatte di ciascun torneo
- preparazione di tabelle poi elaborate con EXCEL per ottenere i grafici di seguito presentati
- analisi fattoriale di cui al paragrafo 4.4.

Un ultima precisazione: per non appesantire il teso, le tabelle relative ai risultati ottenuti nei vari esperimenti sono state tutte raccolte nella appendice B.

4.4 I risultati del torneo senza rumore

Il primo torneo è la replica esatta del torneo di Axelrod con 63 partecipanti; se, con gli stessi partecipanti, si ritrovano i risultati allora ottenuti, si è certi di avere un modello che replica esattamente le condizioni dei primi esperimenti e che può essere usato per approfondire lo studio del Dilemma del Prigioniero ripetuto.

Le modalità del torneo sono quelle descritte nel paragrafo 2 di questo capitolo. In particolare, per essere quanto più fedeli possibile all'esperimento originale, si è rinunciato a determinare probabilisticamente il numero delle mosse di ciascuna partita e si sono mantenute inalterate le durate dei singoli incontri per i 5 tornei previsti. Axelrod (Axelrod 1985) riferisce che la probabilità di concludere un incontro a ciascuna mossa venne scelta in modo che la durata media prevista di ciascuna partita fosse di 200 mosse (come era stata nel primo torneo). In pratica la durata di questi incontri fu determinata una volta per tutte mediante estrazione di un campione casuale. Il campione casuale risultante dall'implicita distribuzione mostrò come le lunghezze delle cinque partite per ciascuna coppia di giocatori sarebbero state di 63, 77, 151, 156 e 308 mosse. La durata media dell'incontro risultò così essere inferiore a quanto previsto (le 200 mosse circa). La matrice 63x63 dei punteggi dei cinque tornei è in appendice B.

I risultati ottenuti con il modello ad agenti indipendenti hanno confermato i risultati dello storico esperimento, validando in questo modo il modello.

- Tit for Tat risulta essere sempre il vincitore del torneo (che è stato ripetuto nella forma descritta 5 volte).
- Le strategie che occupano le ultime 10 posizioni del torneo di Axelrod e di questo modello ad agenti indipendenti sono le stesse; vi possono essere lievi variazioni nella posizione in classifica finale tra un esperimento ed un altro e rispetto ai risultati presentati da Axelrod, ma sempre viene confermata la composizione del gruppo di coda.
- Le strategie presenti nelle prime dieci posizioni sono le stesse, con un'unica eccezione di cui si dirà in seguito; anche per le prime posizioni in classifica vi possono essere lievi variazioni tra un esperimento ed un altro e rispetto ai risultati presentati da Axelrod. Queste variazioni non sono state giudicate significative, essendo, in genere, spostamenti di una sola posizione più avanti o più indietro. Come si può osservare nel grafico dei risultati del torneo a "rumore zero" per 200 reiterazioni al paragrafo precedente, vi sono molte strategie che riescono ad ottenere punteggi molto vicini tra loro, causando frequentemente scambi di posizione in classifica.
- Le strategie, per avere probabilità di entrare nella parte alta della classifica, devono essere «buone». Di tutte le caratteristiche delle 63 strategie presenti nel torneo, l'unica che ha un netta correlazione con i risultati è il fatto di essere «buona». Ai fini dell'analisi del torneo e della classificazione delle strategie, si sono considerate «buone» le strategie che non sono mai le prime a defezionare. Ebbene, una sola delle strategie non buone (quella dell'agente N.41 cioè la K75r dell'esperimento di Axelrod presentata da P. D. Harrington) è riuscita ad entrare nei primi dieci; e una sola (quella dell'agente N.38 cioè la K39r dell'esperimento di Axelrod presentata da Tom Almy) nelle posizioni dall'undicesima alla

ventesima.

La tabella in appendice B classifica le 63 strategie (quindi compreso RANDOM) a seconda che siano state o no definite «buone». Poiché, come si è detto, i concorrenti del girone a 63 erano a conoscenza dei risultati del primo (a 15 partecipanti, qui non incluso nel modello ad agenti indipendenti), pubblicati Axelrod (1980a e 1980b) e distribuiti a tutti i partecipanti, non c'è dubbio che le lezioni apprese nel primo torneo sia state usate per preparare le strategie di questo secondo. Il consiglio non solo ad essere «buoni», ma anche ad essere clementi è stato ben presente nelle menti dei partecipanti quando hanno elaborato le strategie; sono state classificate «buone» 39 strategie sulle 63 presenti (62%) e questo sicuramente caratterizza il torneo.

L'unica differenza tra i risultati del modello ad agenti qui presentato ed il torneo originale di Axelrod sta nella posizione della strategia dell'Agente N.6 che rappresenta quella presentata da Danny C. Champion. Mentre nel torneo originale si piazza in seconda posizione, nel modello ad agenti indipendenti risulta assai meno efficace, tanto da scivolare sotto la ventesima posizione. Dall'esame del software pubblicato sul sito Internet (http://pscs.physics.lsa.umich.edu/Software/CC/CC2/TourExec1.1.f.html) si evidenzia, nell'opinione di chi scrive, un errore di codifica. Qui di seguito è riportato il metodo che «traduce» la strategia di D. C. Champion.

```
//---------------------------------------------------------------------
//---------------------------------------------------------------------
- (int) performBehaviour6
{                       /* Axelrod-K61r  DANNY C. CHAMPION */
   int ICOOP;
  Interface* anInterface;
  anInterface = (Interface*) saveInterface;

// LOAD PARAMETERS - there was not in Axelrod's experiment
// it is here necessary because of the environment of indipendent Agents
// load parameters
        ICOOP = [anInterface getForK61r];
// istruction added- necessary because the Axelrod's tournament fixed
// the first action of the agent and the one of his opponent  at =0
// here are fixed at the "eye catcher value" =2
        if (your1a == 2) your1a=0;          /* instruction added */
// end section added
    /* first of all I extract a random integer  */
    d1 = [uniformDblRand getDoubleWithMin: 0 withMax: 1];
    /* ITURN = cycle
       ISPICK = your1a */

//    if (cycle == 1) { action = 0; ICOOP=0; } /* added ICOOP=0 */
    if (cycle == 1) { action = 0; }          /* version on the Net */

    if (your1a == 0) ICOOP++;
    if (cycle <= 10 ) goto lexit;
    action = your1a;
    if (cycle <= 25) goto lexit;
    action = 0;
```

```
// it is necessary multiply for 1.0 otherwise the division between two
// integers gives an integer even if the variabile is defined float
    f1 = (ICOOP*1.0)/(cycle*1.0) ;              /* here f1 is COPRAT */
    if ((your1a == 1) && (f1 < 0.6) && (d1 > f1)) action = 1;
lexit:

// STORE PARAMETERS before ending this behaviour for this agent
        [anInterface setForK61r: ICOOP];
// end store parameters -end of section added
     return action;
}
```

In grassetto sono evidenziate la due versioni del metodo provate; quella attualmente attiva (senza // ad inizio riga) è quella conforme a quanto di trova sul sito, ma che da risultati diversi. La differenza è unicamente nell'aggiunta dell'istruzione di inizializzazione della variabile ICOOP. Con questa modifica effettivamente questa strategia entra stabilmente nelle prime dieci, ma mai in seconda posizione come invece riferisce Axelrod. La versione del programma di Champion reperibile sul sito è frutto di una successiva trascrizione avvenuta nel 1993; è possibile che si siano perse alcune istruzioni. Poiché i risultati complessivi ed i conseguenti ragionamenti su di essi non vengono inficiati da questa differenza di risultato, si è preferito rimanere aderenti a quanto specificato sul sito Web.

Per approfondire l'indagine sui risultati si è poi effettuata un'analisi fattoriale della tabella dei punteggi. I programmi SAS utilizzati nonché alcuni degli output prodotti sono riportati in appendice A.

Come si può osservare dalla matrice di correlazione, il valore cumulato di variabilità spiegata dai primi sei fattori è del 76%, valore che può essere considerato soddisfacente. Si è perciò rilanciata la procedura SAS che effettua l'analisi fattoriale dando come input il numero (6) di fattori da considerare, e di questo secondo run si è riportato l'output. Per cercare di semplificare l'aspetto interpretativo si è effettuata una rotazione degli assi di riferimento con il metodo VARIMAX.
La lettura del Rotated Factor Pattern, ovvero dei fattori dopo che è stata effettuata la rotazione, consente alcune deduzioni.

- I primi due fattori ruotati sembrano essere fortemente correlati con le variabili che rappresentano i punteggi conseguiti con alcune delle strategie che sono risultate nelle prime posizioni. Compaiono però anche correlazioni con i punteggi ottenuti dall'incontro con strategie che si sono piazzate tra la 40° e la 50° posizione; questo, probabilmente, indica come per ottenere un buon punteggio non è sufficiente essere cooperativi con chi lo è (tendenzialmente le strategie «buone», che si sono posizionate nelle prime posizioni) ma che bisogna essere sufficientemente «ritorsivi» con chi non è collaborativo (tendenzialmente le strategie non buon che si sono piazzate nelle ultime posizioni di classifica).
- I fattori 3 e 6 sono interpretabili come i risultati che si sono ottenuto con selezionate strategie. E' interessante notare come nessun fattore sia significativamente correlato in modo positivo con le due strategie casuali (la 23 RANDOM e la 22 Hotz), fatto probabilmente interpretabile con la scarsa necessità di «fare bene» con strategie casuali, generatrici di rumore, che si ha nel torneo così strutturato.
- Il fattore 4 rappresenta la correlazione positiva con le posizioni di bassa classifica (sotto la 40° posizione) mentre il fattore 5 presenta la stessa correlazione ma negativa.

4.5 I risultati del torneo con rumore ambientale

In questa fase dell'esperimento si sono effettuate una serie di prove in cui il numero degli agenti veniva incrementato di 5 in 5 a partire da 63 fino a 93. Ogni incontro tra due agenti è composto di 200 iterazioni del Dilemma del Prigioniero con la stessa matrice dei pagamenti vista precedentemente; la struttura del torneo è ancora round robin tra agenti tutti reattivi.

Nelle tabelle dei punteggi riportate in appendice B, i vari esperimenti sono stati contrassegnati come b0 quando c'erano zero agenti in più, b5 quando erano 5, b10 per 10 agenti in più e così via fino a b30. Tutti gli agenti aggiuntivi si comportano secondo la strategia RANDOM in modo da aumentare, via via fino al 32%, il numero delle scelte che, nel torneo considerato nel suo complesso, sono definite in modo casuale.

Ogni esperimento viene ripetuto tre volte e si considerano i punteggi medi, in modo da renderli confrontabili anche tra tornei con un numero di partecipanti diverso. La molte di dati da trattare è cospicua e si è nuovamente fatto uso di programmi SAS (riportati in appendice A). Per aumentarne la leggibilità, i risultati sono stati riportati nei grafici qui di seguito.

Qui grafico excel file: Grafico4_3.pdf

Nel grafico non viene tracciato l'andamento del punteggio di un singolo agente, ma del punteggio ottenuto da quell'agente che ha terminato il torneo in quella posizione di classifica. La linea più in alto rappresenta così il punteggio medio dei vincitori (massimo punteggio) mentre quella più in basso è la linea del punteggio minimo ottenuto.

Come si può vedere dal grafico, l'effetto del rumore ambientale è tutto sommato modesto; la differenza tra punteggio massimo e minimo dal 53,9% sale fino al 62,2% e la differenza è tutta dovuta all'aumento del punteggio minimo. Il vincitore continua a riuscire a collezionare tra i 570 e i 578 punti: però, come vedremo nei grafici successivi, il vincitore non è sempre lo stesso.

Non sembra quindi che l'introduzione di questo «rumore ambientale» renda realmente caotico il modello; sembra piuttosto che faccia emergere altre strategie, diverse da Tit for Tat, meglio attrezzate per affrontare avversari che generano azioni in modo casuale. Questo è d'altra parte comprensibile. E' vero che si introducono percentuali anche molto elevate di scelte effettuate a caso nel modello, ma queste scelte casuali sono concentrate nel comportamento di alcuni agenti. Poiché l'esistenza di almeno una strategia RANDOM era uno dei termini definitivi del torneo originale, è possibile che alcune delle strategie ne abbiano tenuto conto e riescano a confrontarsi efficacemente con RANDOM.

Nei grafici che seguono vengono messe a confronto le posizioni nei vari torneo a «rumore ambientale» crescente di alcune famiglie di strategie che si

comportano in modo analogo al mutare delle condizioni ambientali.

Qui grafico excel file: Grafico4_4.pdf

Nel primo grafico si può osservare come il vincitore del torneo Tit for Tat, non si riveli una strategia particolarmente robusta se deve far fronte a dosi crescenti di «rumore ambientale». Quando un agente ne incontra un altro di cui non riesce a decifrare il comportamento, si trova in una situazione in cui non può sapere se la scelta di cooperare o defezionare del suo avversario è stata effettuata a caso o se segue una strategia ma che è per lui incomprensibile.

L'introduzione di rumore ambientale crea delle condizioni difficili specialmente per Tit for Tat; questa strategia basa la sua efficacia non solo sulla disponibilità a cooperare ma anche sulla ineluttabilità della ritorsione in caso di defezione da parte dell'avversario. Se però Tit for Tat riceve una defezione non intenzionale, vale a dire causata da un comportamento scelto a caso o dall'azione del rumore, può innescare una eco di ritorsioni che rendono molto meno efficace questa strategia. Tit for Tat e le strategie analoghe hanno bisogno di chiarezza, di assenza di rumore per essere efficaci: poiché sono estremamente rapide nel decidere la ritorsione, non si possono permettere di interpretare male la decisione dell'avversario.

Le altre strategie del gruppo presentano caratteristiche in qualche modo simili. Ad esempio l'agente N. 14 opera secondo la strategia proposta da Abraham Getzler; questa tiene conto non solo dell'ultima azione dell'avversario (come invece fa Tit for Tat) ma di tutte quante, valutando però con importanza via via decrescente quelle più lontane nel tempo. Nel torneo svolto con rumore zero è in classifica nelle prime dieci posizioni, ma, al crescere del numero degli agenti RANDOM, rapidamente scende in classifica. Come Tit for Tat ha necessità di essere certa di interpretare correttamente i "segnali" che riceve dall'avversario.

Qui grafico excel file: Grafico4_5.pdf

Decisamente opposto è l'andamento delle strategie evidenziate nel secondo grafico. Più c'è rumore più migliorano le loro performances. Come abbiamo visto prima, il punteggio delle prime posizioni rimane sostanzialmente inalterato e perciò ne consegue che effettivamente migliorano i punteggi conseguiti. L'ipotesi avanzata precedentemente, cioè che certe strategie inglobassero delle parti specificatamente dedicate allo scopo di determinare se il proprio avversario fosse la strategia RANDOM che si sapeva essere comunque presente, va ora verificata.
Qui di seguito si riporta parte del codice del metodo che realizza la strategia di Charles Kluepfel (agente N. 13).

```
// after cycle 26 try detecting random
e540:       if(cycle < 27) goto e550;
            d1=((C1+C2)-1.5*sqrt(C1+C2))/2.0;
            if (C1 < d1) goto e550;
            d1=((C3+C4)-1.5*sqrt(C3+C4))/2.0;
            if (C4 < d1) goto e550;
            action = 1;
```

```
           goto e590;
e550:      action=0;
           if (J1 != your1a) goto e570;
           if (J2 != J1) goto e580;
// respond in kind to 3 in row.
```

In grassetto il commento dell'autore che indica chiaramente la volontà di scoprire se si ha come avversario proprio quella strategia. Evidentemente il sistema studiato dall'autore funziona e consente al relativo agente di rimontare molte posizioni in classifica.

Questo risultato si presta a considerazioni di carattere più generale. Il "rumore ambientale" previsto da questo modello, ha la caratteristica essenziale di essere "concentrato" in alcuni agenti e può perciò essere scoperto, isolato e, di conseguenza, si possono basare le proprie strategie di risposta sulla convinzione che il proprio avversario sia un giocatore RANDOM o incomprensibile dir si voglia.

Qui grafico excel file: Grafico4_6.pdf

Nel terzo grafico, in cui si sono riuniti quegli agenti che sembrano poco influenzati dal "rumore ambientale", compare il vincitore di molti tornei. L'agente N.7 ovvero la strategia di Otto Borufsen. Questa strategia si dimostra eccezionalmente robusta poiché, praticamente per qualsiasi livello di rumore riesce a mantenere o la prima o la seconda posizione in classifica.

Senza riportare per intero la codifica della strategia, si mettono qui in evidenza alcuni commenti dell'autore particolarmente interessanti.

```
//--------------------------------------------------------------------
- (int) performBehaviour7
{                                  /* Axelrod-K42r  Otto Borufsen */
   int* K42r;                       /* matrix of values for K42r */

if (IDEF == 0) goto lab30;
// opponent has been proved "random" or "defective", I defect for 25 moves
          action = 1;
          goto lab100;
lab30:
          if ((IPICK == 0) || (your1a == 0)) goto lab40;
// mutual defection on the last move
          L3MOV++;
          if (L3MOV < 3) goto lab50;
// mutual defection on the last threee moves.
// I cooperate once on the next move
          action = 0;
          L3MOV = 0;
          L3ECH = 0;
          goto lab100;
// one (or both) cooperate last move
lab40:
          L3MOV = 0;
          if (IPICK == your1a) goto lab45;
          if ((your1a != I2PCK) || (IPICK != J2PCK)) goto lab45;
// echo-effect on the last move
          L3ECH++;
          if (L3ECH < 3) goto lab50;
// echo-effect on the last three moves.
// my next defection will be substituted by a cooperation
          L3ECH = 0;
```

```
          L3MOV = 0;
          ICOOP2 = 1;
          goto lab50;
lab45:
          L3ECH = 0;
// play Tit for Tat as main rule
lab50:
```

Come si può vedere anche solo dalla lettura dei commenti qui evidenziati in grassetto, questa strategia accoglie molte delle suggestioni provenienti dall'esame di questo torneo:

- giocare Tit for Tat come regola principale
- cercare di capire se il proprio avversario è RANDOM
- controllare gli effetti eco
- essere pronti a "perdonare" e, quindi, a cooperare anche se si è ricevuto una defezione (almeno sotto certe condizioni)

Questa strategia, che nel torneo base non riesce a superare Tit for Tat, si dimostra in compenso molto più robusta, tanto da risultare vincitrice 4 volte nei 7 diversi tornei organizzati.

Vengono in questo modo provate le intuizioni presentate nei primi due paragrafi e relative alla necessità di "bontà", "clemenza" e capacità di effettuare efficaci ritorsioni. A queste qualità sembra ora aggiungersi anche la capacità di valutare la "comprensibilità" dell'avversario e decidere strategie di comportamento diverse a seconda che venga valutato RANDOM (o incomprensibile dir si voglia) oppure no.

Qui grafico excel file: Grafico4_7.pdf

Il quarto grafico presenta strategie non molto efficaci che con l'aumentare del rumore vanno anche peggio; è qui presentato solamente per confronto con i risultati dell'esperimento sul "rumore di canale" che sarà commentato nel paragrafo seguente.

4.6 I risultati del torneo con rumore di canale

Una seconda situazione in cui può accadere che si generino dei fraintendimenti tra gli agenti impegnati in un incontro basato sulla reiterazione del dilemma del prigioniero è quando il canale di comunicazione tra gli agenti è soggetto a rumore, vale a dire che l'azione decisa da un agente viene comunicata in modo errato. Nel nostro modello, poiché le azioni possibili sono solamente due, cooperare o defezionare, l'effetto del rumore è piuttosto drastico, facendo diventare cooperazione una defezione e viceversa.

In questa fase dell'esperimento si sono effettuate una serie di prove in cui il numero degli agenti rimaneva di 63, ma veniva incrementata progressivamente la probabilità che vi fosse generato del rumore.

Il canale di comunicazione si interpone tra i due agenti che stanno effettuando un incontro, o meglio tra ciascuno di essi e Club che gestisce l'incontro stesso. Per mantenere la stessa struttura del modello, almeno in questa fase sperimentale, si è scelto di introdurre il rumore di canale tra il RuleMaster che applica la strategia e decide l'azione da eseguire e l'agente che ne ha richiesto l'applicazione. Il numero di iterazioni del dilemma del prigioniero che compongono un incontro tra due agenti è mantenuto a 200; la struttura del torneo è ancora round robin tra agenti tutti reattivi.

I vari esperimenti sono stati contrassegnati nelle tabelle dei punteggi come c000 quando c'era un livello di "rumore di canale" pari a zero (ovvero probabilità zero che un'azione venisse invertita), c001 quando la probabilità era dello 0,1%, c003 se era del 0,3% e così via fino a c100 in cui la probabilità è del 10%. Si sono fatti esperimenti anche con livelli del 20% e del 40% di probabilità di invertire la scelta. A questi livelli però il comportamento è sostanzialmente caotico per tutte le strategie e ciascun run dà luogo a risultati completamente diversi. Si è perciò scelto di non mettere in grafico i valori di questi ultimi due esperimenti, anche se nelle tabelle dei punteggi possono comparire.

Ogni esperimento viene ripetuto almeno tre volte e si considerano i punteggi medi. Anche in questo caso la quantità di dati da trattare è grande e si è nuovamente fatto uso di programmi SAS sostanzialmente uguali a quelli visti nel paragrafo precedente e perciò non più riportati in appendice A.

Qui grafico excel file: Grafico4_8.pdf

Il grafico, costruito esattamente come l'analogo del paragrafo precedente, viene evidenziato l'andamento del punteggio ottenuto dall'agente che ha terminato il torneo in una certa posizione di classifica. La linea più in alto rappresenta così il punteggio medio dei vincitori (massimo punteggio) mentre quella più in basso è la linea del punteggio minimo ottenuto.

In questo caso l'effetto del "rumore di canale" è piuttosto evidente. Il rapporto percentuale tra il punteggio minimo ottenuto ed il massimo passa dal 54,6% al 76,6%; non compaiono in grafico i casi c200 e c400 rispettivamente con probabilità di rumore al 20% e 40% in cui questo rapporto supera il 90%. Si entra in somma in una situazione molto simile ad una puramente caotica.

L'andamento del valore minimo è assai simile nei due esperimenti effettuati sui due diversi tipi di rumore (da 315,5 a 352,8 contro una variazione per il rumore ambientale vista prima di 356,9 su 311,7) mentre l'andamento del valore massimo è chiaramente diverso. In questo caso nessuna strategia, anche se vincente riesce a mantenere un alto livello di punteggio.

E' da notare, infine, come la competizione per le prime posizioni sia piuttosto serrata; la maggior parte delle strategie (almeno tutte quelle che occupano fino alla 50° posizione in classifica) ottengono punteggi medi per 200 iterazioni superiori a 500. Per evidenziare meglio il fenomeno si è riportato in grafico i valori percentuali rispetto al punteggio massimo dei punteggi delle varie posizioni per il torneo a rumore zero.

Qui grafico excel file: Grafico4_9.pdf

L’introduzione del “rumore di canale” rende progressivamente più caotico il modello tanto che, già per il valore del 10%, si assiste ad una grande variabilità di risultati da un torneo ad un altro. Una percentuale più bassa, ma non concentrata su singoli agenti, di azioni decise a caso, provoca effetti di “disorientamento” tra le strategie molto più accentuati. Anche in questo caso, come in quello precedente, l’assoluto vincitore Tit for Tat non riesce a gestire la situazione di torneo con una percentuale di rumore superiore allo 0,3%. Si può osservare come vi sia un intervallo di valori per la percentuale di rumore di canale introdotto compresi tra lo 0,3% e il 5% in cui la situazione non è completamente caotica ma l’effetto del rumore è sensibile. Se vi sono strategie che meglio affrontano il problema vanno cercate tra i vincitori di questi tornei.

Nei grafici che seguono vengono messe a confronto le posizioni nei vari tornei a «rumore di canale» crescente di alcune famiglie di strategie che si comportano in modo analogo al mutare delle condizioni ambientali. Ognuno dei quattro grafici corrisponde all’analogo fatto in base ai risultati ottenuti introducendo l’altro tipo di rumore testato.

Qui grafico excel file: Grafico4_10.pdf

Nel primo grafico si possono osservare lo stesso gruppo di strategie, tra cui il vincitore del torneo senza rumore Tit for Tat. L’andamento della posizione in classifica è assolutamente analogo al primo caso: Questa strategie è sensibile ad entrambe i tipi di rumore e, valgono anche in questo caso le considerazioni fatte precedentemente.

L’introduzione di rumore ambientale crea delle condizioni difficili specialmente per Tit for Tat; strategia che ha bisogno di chiarezza per poter interpretare senza dubbi il comportamento dell’avversario e attivare le azioni di ritorsione senza errori e senza il pericolo di innescare effetti eco non più controllabili.

Come visto anche prima, le altre strategie del gruppo presentano caratteristiche simili. In questo caso l’andamento delle posizioni raggiunte in classifica è meno preciso dell’altro caso. Questo può essere causato dal fatto che ora il punteggio massimo (quello del vincitore) diminuisce mentre quello minimo aumenta; vengono così “schiacciate” in meno punti più posizioni di classifica, rendendo più probabili degli scambi di posizione.

Parzialmente diverso è l’andamento un po’ altalenante dell’agente N. 19 che mette in atto la strategia elaborata da Eatherley.

```
//------------------------------------------------------------------------
- (int) performBehaviour19
{                       /* Axelrod-K46r GRAHAM J. EATHERLEY */
  int NJ;
  float P ;
  Interface* anInterface;
```

```
  anInterface = (Interface*) saveInterface;

// LOAD PARAMETERS - there was not in Axelrod's experiment
//    it is here necessary because of the environment of indipendent Agents
// load parameters
        NJ = [anInterface getForK46r];
// istruction added- necessary because the Axelrod's tournament fixed
// the first action of the agent and the one of his opponent  at =0
// here are fixed at the "eye catcher value" =2
        if (your1a == 2) your1a=0;          /* instruction added */
// end section added

  /* first of all I extract a random integer  */
    d1 = [uniformDblRand getDoubleWithMin: 0 withMax: 1];
    if (cycle == 1) NJ=0;
   NJ = NJ + your1a;
   action =0;
   if (your1a == 0) goto eexit;
   f1=NJ*1.0; f2=(cycle-1)*1.0; P=f1/f2;
   if (d1 < P) action = 1;
eexit:
// STORE PARAMETERS before ending this behaviour for this agent
        [anInterface setForK46r: NJ];
// end store parameters -end of section added
    return action;
}
```

Come si può osservare questa strategia è così riassumibile:

- se l'avversario ha collaborato, io collaboro (esattamente come Tit for Tat)
- se l'avversario ha tradito, tradisco se il numero estratto a caso è inferiore al rapporto tra tradimenti ricevuti e mosse giocate; cioè è tanto più improbabile che tradisca di fronte ad un tradimento quanto più il mio avversario è stato cooperativo (in media) nelle mosse passate.

Questo significa introdurre una sorta di "capacità a perdonare" una singola defezione, perdono che è tanto più probabile quanto più l'avversario è stato collaborativo nel passato. Quando la probabilità di una defezione non voluta (causata cioè dal rumore di canale) aumenta, questa strategia sembra piuttosto efficace, almeno in termini relativi.

Qui grafico excel file: Grafico4_11.pdf

L'andamento di queste quattro strategie è di netto miglioramento in classifica. Occorre però notare che, sebbene l'andamento del grafico sia analogo a quello visto per l'altro tipo di rumore, vi è un'importante differenza: le strategie sono *tutte diverse*. Infatti sono quelle rappresentate nel quarto grafico del paragrafo precedente. Cambiando tipo di rumore il risultato è assai diverso. Qual è la caratteristica che hanno in comune queste strategie?

Gli agenti N.59, 4 e 16 giocano un Tit for Tat modificato in modo da tradire solamente se il proprio avversario ha tradito e se il numero di volte che nel passato ha cooperato è basso. Si affida cioè la decisione di perdonare un tradimento alla "reputazione" dell'avversario: valutare quante volte ha cooperato e basare su questo la decisione se tradire in caso di tradimento significa riconoscere che un avversario

che generalmente coopera, può aver tradito per sbaglio (come in effetti avviene a causa del rumore di canale introdotto). Da questo andamento (efficace per il "rumore di canale" ma non tanto per il "rumore ambientale") si possono trarre due interessanti conclusioni:

- i due tipi di rumori sono strutturalmente diversi, e strategie atte ad affrontare il primo tipo non sono altrettanto efficiente nell'affrontare il secondo
- per cercare di vincere un torneo con una sensibile percentuale di "rumore di canale" una strategia semplice ed efficace è basare la scelta di tradire (non quella di cooperare che viene comunque effettuata in caso di cooperazione da parte dell'avversario) semplicemente sulla "reputazione".

L'agente N.58 mette in atto la strategia Tit for Two Tats, vale a dire tradisce solo se è tradito due volte di seguito. Non cerca quindi di valutare la "reputazione" dell'avversario, ma si ingegna semplicemente ad ammortizzare l'effetto eco che può avere effetti assai pesanti per la strategia Tit for Tat. Come si può vedere dal grafico questa scelta comincia a dare i suoi frutti da percentuali di rumore del 3% in poi.

Qui grafico excel file: Grafico4_12.pdf

Come si comportano le strategie che con il tipo precedente di rumore avevano visto un netto miglioramento delle posizioni al crescere del rumore nel modello? Come si può vedere dal grafico, per alcuni agenti il miglioramento è semplicemente scomparso, per altri si è molto attenuato. Questo a dimostrazione di quanto affermato prima e cioè che anche le strategie che si prefiggono di affrontare il problema del rumore devono avere ben chiaro che tipo di rumore vogliono affrontare.

Qui grafico excel file: Grafico4_13.pdf

In questo grafico sono raccolte le strategie che negli esperimenti precedenti si erano dimostrate capaci di "mantenere il livello delle prestazioni" all'aumentare del rumore. Per tutte esiste un livello di rumore (per alcune lo 0,3% per altre il 3%) oltre il quale la posizione in classifica si fanno via via meno interessanti.

Le buone qualità evidenziate con l'altro tipo di rumore dalla strategia di Otto Borufsen si rivelano (almeno per percentuali superiori al 5%) insufficienti ad affrontare la situazione. Sembra più efficiente basare ora la strategia sulla "reputazione". L'inaffidabilità progressivamente più alta della bontà della singola comunicazione fa emergere quelle strategie che si basano sul comportamento "complessivo" del proprio avversario, insomma sulla "reputazione".

4.7 Quattro nuove strategie

In quest'ultima fase dell'esperimento il numero degli agenti viene portato a 67, in modo da inserire nel torneo le quattro nuove strategie descritte nel paragrafo 3. Il numero di iterazioni del dilemma del prigioniero che compongono un incontro tra due agenti è mantenuto fisso a 200; la struttura del torneo è ancora round robin tra agenti tutti reattivi.

Il canale di comunicazione si interpone tra i due agenti è soggetto a "rumore", come negli esperimenti descritti nel paragrafo 6.

Anche qui i vari esperimenti sono stati contrassegnati nelle tabelle dei punteggi come c000 quando c'era un livello di "rumore di canale" pari a zero (ovvero probabilità zero che un'azione venisse invertita), c001, c003, e così via fino a c100. Non sono più stati effettuati esperimenti con livelli del 20% e del 40% di probabilità; a questi livelli il comportamento del modello è sostanzialmente caotico.

Ogni esperimento viene ripetuto almeno tre volte e si considerano i punteggi medi. I programmi SAS usati per l'elaborazione dei dati sono sostanzialmente uguali a quelli visti nel paragrafo precedente e perciò non più riportati in appendice A.

Qui grafico excel file: Grafico4_14.pdf

Il grafico è costruito esattamente come l'analogo del paragrafo precedente ed evidenzia l'andamento del punteggio ottenuto dall'agente che ha terminato il torneo in una certa posizione di classifica.

Sia l'aumento del valore minimo sia la diminuzione del valore massimo al variare della quantità di "rumore di canale" introdotta sono molto simili a quelli visti nel torneo a 63 agenti. Anche in questo caso nessuna strategia, pur vincente, riesce a mantenere un alto livello di punteggio.

L'ultima serie di esperimenti qui presentata, riguarda essenzialmente, come anticipato nel paragrafo 3, il test di quattro strategie non presenti nel torneo originale. Si sono perciò riportati in grafico solamente i risultati di queste quattro strategie.

Qui grafico excel file: Grafico4_15.pdf

Sia la strategia PAVLOV che la versione "pentita" di Tit for Tat hanno la positiva caratteristica che quando giocano con una strategia gemella sono in grado di ricuperate velocemente un errore isolato causato dal rumore. Esaminiamo che cosa succede in caso di errore isolato. Se, in un incontro tra due strategie PAVLOV, una delle due manifesta all'altra una defezione (non intenzionale) entrambi defezioneranno alla mossa successiva; ma alla mossa dopo, entrambi ricominceranno a collaborare. Se uno dei due giocatori di Contrite TFT (la versione "pentita" di Tit for Tat) comunica all'altro una defezione non causata da defezione dell'avversario,

entra in uno stato di pentimento che lo farà collaborare quando riceverà la ritorsione dell'avversario; alla mossa successiva entrambe entreranno in uno stato di "contento" e ricominceranno a collaborare. Sfortunatamente, per gli agenti che usano la strategia PAVLOV, la volontà di collaborare dopo una mutua defezione si presta ad uno sfruttamento opportunistico da parte dell'avversario. Infatti, come su può vedere nel grafico successivo che mostra il punteggio conseguito da queste quattro strategie e da Tit for Tat, i risultati di questa strategia e della sua omologa "generosa" non sono particolarmente brillanti e nemmeno migliorano con l'aumento del rumore inserito nel torneo.

Introdurre uno stato di "pentimento" può essere molto utile per correggere i propri errori ma può essere assolutamente inefficace nel correggere gli errori dell'avversario. Ad esempio, se Contrite Tit for Tat si incontra contro Tit for Tat e il "rumore di canale" fa si che Tit for Tat defezioni accidentalmente, l'eco di questo errore si propaga fino a quando un altro errore accidentale non la ferma. Bisogna quindi tenere conto del fatto che le originarie 63 strategie erano state studiate per un ambiente senza "rumore di canale". E' possibile che in un torneo con strategie che tengono conto del rumore, il "pentimento" sia molto più efficace di quello che può esserlo in questo torneo. Con simili strategie, correggere i propri errori è sufficiente poiché gli agenti che si incontrano giocano anch'essi strategie che correggono i loro errori.

La "generosità" di una strategia è invece efficace per fermare l'eco continua di un solo errore sia questo causato da se stessa sia che venga causato dal proprio avversario. Il livello di generosità determina la velocità con cui la cooperazione può essere ristabilita. Questa capacità non è però senza costi, poiché bisogna bilanciare il tradeoff tra questa capacità e la possibilità di essere sfruttata dall'avversario. (Axelrod e Dion 1988).

Come si può vedere dal grafico, in presenza di rumore la scelta di replicare l'azione ricevuta ("reciprocità") funziona ancora, purché sia accompagnata o da "generosità" (possibilità di scegliere di cooperare anche quando si è ricevuto una defezione) o da "pentimento" (cooperazione dopo aver ricevuto una defezione a seguito di una propria defezione). La "generosità" sembra meglio in grado di affrontare i problemi causati dal rumore poiché, come si è visto, è in grado di "disinnescare" l'effetto eco anche se causato da un errore dell'avversario.

Capitolo 5

Evoluzione di nuove strategie

5.1 Utilizzo del classifier system CW

Per procedere all'ultima parte della sperimentazione prevista è stato necessario modificare il modello presentato nel capitolo precedente, al fine di inserire una componente in grado di evolvere nuove strategie. Come già anticipato e discusso nell'ultimo paragrafo del capitolo 2, si è optato per l'utilizzo di un Classifier Systems. Il Classifier System inserito nel modello è essenzialmente quello sviluppato da Ferraris (1999) con SWARM usando lo schema ERA, denominato Classifier Workbench (CW). Al classifier CW sono state apportare le necessarie modifiche in modo da renderlo perfettamente idoneo ad operare all'interno del modello ad agenti indipendenti discusso nel precedente capitolo.

Prima di procedere alla descrizione di questo secondo modello, verranno riassunte le caratteristiche più importanti di CW.

CW è basato sul paradigma dei Classifier System: le singole regole, tradotte come stringhe di 0 e di 1, sono considerate come le idee che realizzano la strategia di un agente. Il nucleo di CW è dunque un sistema di inferenza basato su regole, in grado di evolvere verso il conseguimento di determinati obiettivi, modificando le regole stesse. Il grado di raggiungimento degli obiettivi prefissati è comunicato, per il tramite della valutazione delle singole azioni suggerite, dall'agente dopo ogni interazione; in altre parole, viene comunicato a CW quanto l'azione suggerita è stata efficace. Ogni idea (cioè ogni regola) viene così ad accumulare un patrimonio di forza intrinseca che permette di valutare l'efficacia della regola stessa. L'apprendimento avviene secondo i canoni degli algoritmi genetici:

- selezione delle regole più efficaci
- riproduzione e crossover delle medesime per ottenerne delle nuove.

Tutti i nomi dei file vengono comunicati ai vari oggetti che li utilizzano nel modelSwarm, al momento della loro creazione; nomi e numero dei file utilizzati possono quindi essere agevolmente modificati durante l'utilizzo di CW nel modello

ad agenti indipendenti. Ad esempio, i numerosi parametri interni di funzionamento di CW possono essere letti dal file *classifierParm.dat* che contiene le seguenti variabili:

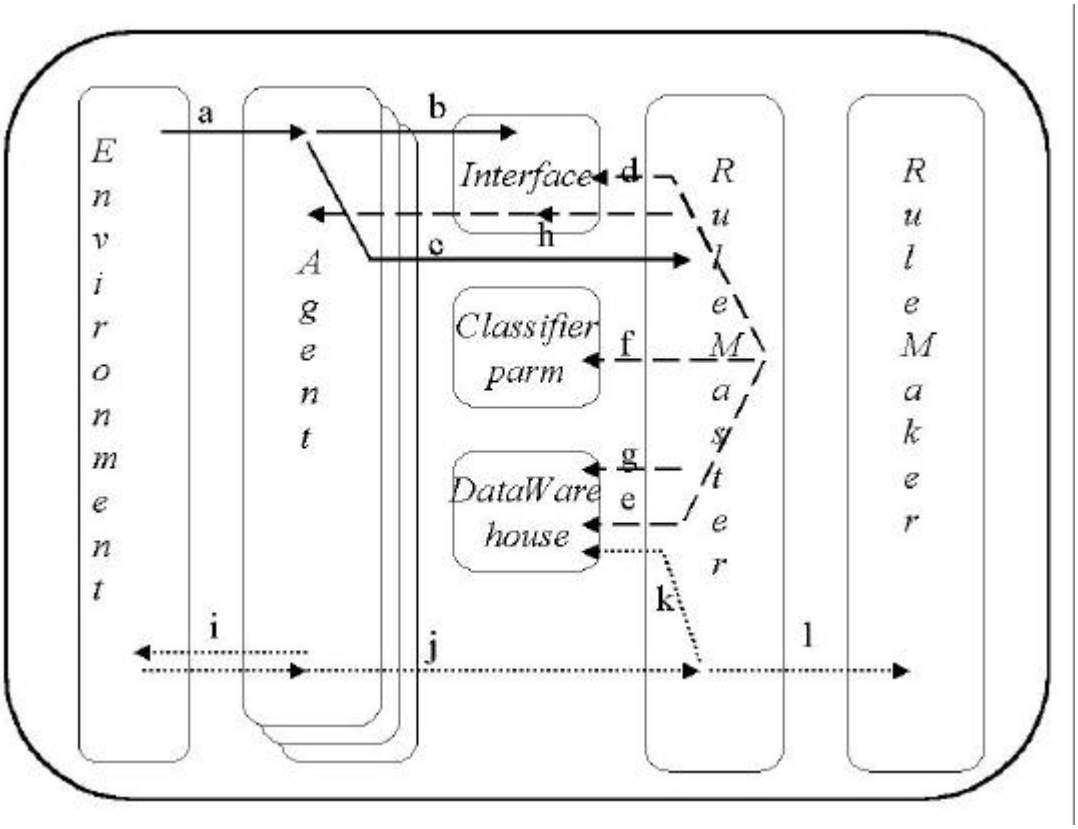
(dal metodo print di ClassifierParm)

```
printf("numberOfRules       = %d\n",numberOfRules);
printf("numberOfEffectors   = %d\n",numberOfEffectors);
printf("geneLength          = %d\n",geneLength);
printf("maxNumberOfMessages = %d\n",maxNumberOfMessages);
printf("effectorsFlag       = %d\n",effectorsFlag);
printf("wildCardRate        = %f\n",wildCardRate);
printf("confidence          = %f\n",confidence);
printf("initialStrength     = %f\n",initialStrength);

printf("bidTaxRate          = %f\n",bidTaxRate);
printf("lifeTaxRate         = %f\n",lifeTaxRate);
printf("bidRatio            = %f\n",bidRatio);
printf("linearBid1          = %f\n",linearBid1);
printf("linearBid2          = %f\n",linearBid2);
printf("effectivelinearBid1 = %f\n",effectiveLinearBid1);
printf("effectivelinearBid2 = %f\n",effectiveLinearBid2);
printf("bidSigma            = %f\n",bidSigma);
printf("bidMu               = %f\n",bidMu);

printf("evolutionRate       = %f\n",evolutionRate);
printf("turnoverRate        = %f\n",turnoverRate);
printf("crossoverRate       = %f\n",crossoverRate);
printf("mutationRate        = %f\n",mutationRate);
printf("crowdingRate        = %f\n",crowdingRate);
printf("crowdingFactor      = %f\n",crowdingFactor
```

La classe DataWarehouse, contenuta in CW, possiede i metodi sia per generare automaticamente, in modo casuale, la popolazione iniziale delle regole sia per effettuarne il caricamento da file. Risulta, quindi, possibile far apprendere il sistema in condizioni controllate per poi inserirlo nel torneo completo a 63 agenti. Ogni agente deve essere proprietario di un dataWarehouse, dove sono contenute tutte le informazioni relative al suo stato attuale di evoluzione: indirizzi dei gruppi di oggetti rappresentanti gli individui che formano il suo genotipo, indirizzo della regola attiva che ha proposto l'ultima azione, indirizzo dell'oggetto contenente i parametri di funzionamento del classifier per quell'agente, contatori delle varie azioni evolutive effettuate (per fini statistici), eventuale nome del file da cui effettuare il load iniziale della popolazione.

Per rendere la descrizione di CW più immediata, si fa riferimento allo schema qui di seguito raffigurato, che riporta anche la sequenza delle interazioni tra i vari oggetti componenti il Classifier System. Si possono distinguere tre momenti principali:

- l’ambiente comunica all’agente una variazione di stato e le relative informazioni che vengono opportunamente codificate (linee continue);
- il RuleMaster viene attivato e determina l’azione che l’agente deve effettuare verso l’ambiente (linee a tratteggio lungo);
- l’agente esegue l’azione, ne valuta le conseguenze e le comunica al RuleMaster il quale può eventualmente attivare il RuleMaker (linee a tratteggio corto).



Innanzitutto l'agente riceve (a) , dall'ambiente una segnalazione contenente le informazioni necessarie a definirla completamente. I dati ottenuti vengono comunicati (b) all'interface, perché li traduca nella metrica dello specifico classifier. Avendo terminato la codificazione della richiesta, l'agente può richiedere (c) un consiglio al ruleMaster, avendo cura di comunicare l'indirizzo dell'interface e del dataWarehouse da usare.

Il ruleMaster così attivato interroga (d, e, f) l'interface e il dataWarehouse indicati dall'agente, e il classifierParm indicato dal dataWarehouse e ottiene: le informazioni ambientali, gli indirizzi dei dati su cui operare ed i valori dei parametri da applicare durante l'elaborazione. Provvede inoltre a modificare (g) il contenuto delle variabili statistiche ed a variare l'indirizzo della regola attiva, contenuti nel dataWarehouse: viene aggiornata l'immagine del livello evolutivo dell'agente. Stabilita l'azione da effettuare questa viene comunicata (h) all'agente attraverso la sua interface.

Successivamente l’azione stabilita dal ruleMaster viene effettuata (i) verso l’ambiente dall’agente. L'agente valuta le conseguenze dell'azione e le comunica (j) al ruleMaster. Quest'ultimo, ricostruito il classifier specifico interrogando dataWarehouse ed interface indicati dall'agente, provvede ad alterare i valori dei patrimoni delle regole in base alla ricompensa ricevuta; i dati nel dataWarehouse vengono aggiornati (k). In dipendenza dai parametri utilizzati e dallo stato dell'evoluzione, il ruleMaster decide se richiedere (l) i servizi del ruleMaker, per evolvere l'insieme di individui contenuto nel dataWarehouse che, terminata l'attività, sarà ulteriormente aggiornato a cura del ruleMaker stesso.

Per completare la descrizione del Classifier CW è necessario fornire ancora alcuni elementi sul funzionamento delle componenti più “interne” del sistema

Genomi. Ciascuna rule contiene un cromosoma formato da due genomi, array di caratteri, che rappresentano, rispettivamente, la condition-part e la action-part. La loro estensione, numero di posizioni utilizzate, può variare fino ad un massimo codificato comunque facilmente estensibile. La terna di valori usata è {0, 1, #}; quest'ultimo rappresenta il carattere di indifferenza. Apposite variabili contengono l'ammontare del patrimonio di crediti di ciascuna regola (*strength*), il valore di specificità della stessa (*specificity)*, e l'ammontare dell'offerta, fatta dalla regola nell'asta per determinare quali individui avranno il diritto di immettere il proprio messaggio nella messageList. Ogni regola contiene una partnerList dove memorizzare gli indirizzi delle entità autrici dei messaggi che hanno permesso l'attivazione, conseguente al match, della regola stessa.

Messaggi. I messaggi possiedono un cromosoma formato da un solo genoma, contenente il testo del messaggio, cioè la traduzione delle informazioni ambientali o la copia della action-part di una regola; in ciascun messaggio è indicato, inoltre, l'indirizzo del suo autore, o proprietario (*owner*).

Effector. Gli effector sono simili alle regole, ma la loro action-part contiene la codifica binaria di un identificativo dell'azione che viene usata dall’interface per tradurre questo valore in un codice azione. Anche gli effector possiedono una partnerList, per memorizzare quali regole li abbiano attivati; un'apposita variabile è utilizzata per registrare il valore di support, cioè la somma delle offerte fatte dalle varie regole che hanno attivato l'effector, utile per decidere quale, fra gli effector attivi, debba prevalere e suggerire l'azione, di cui è portatore, all'agente.

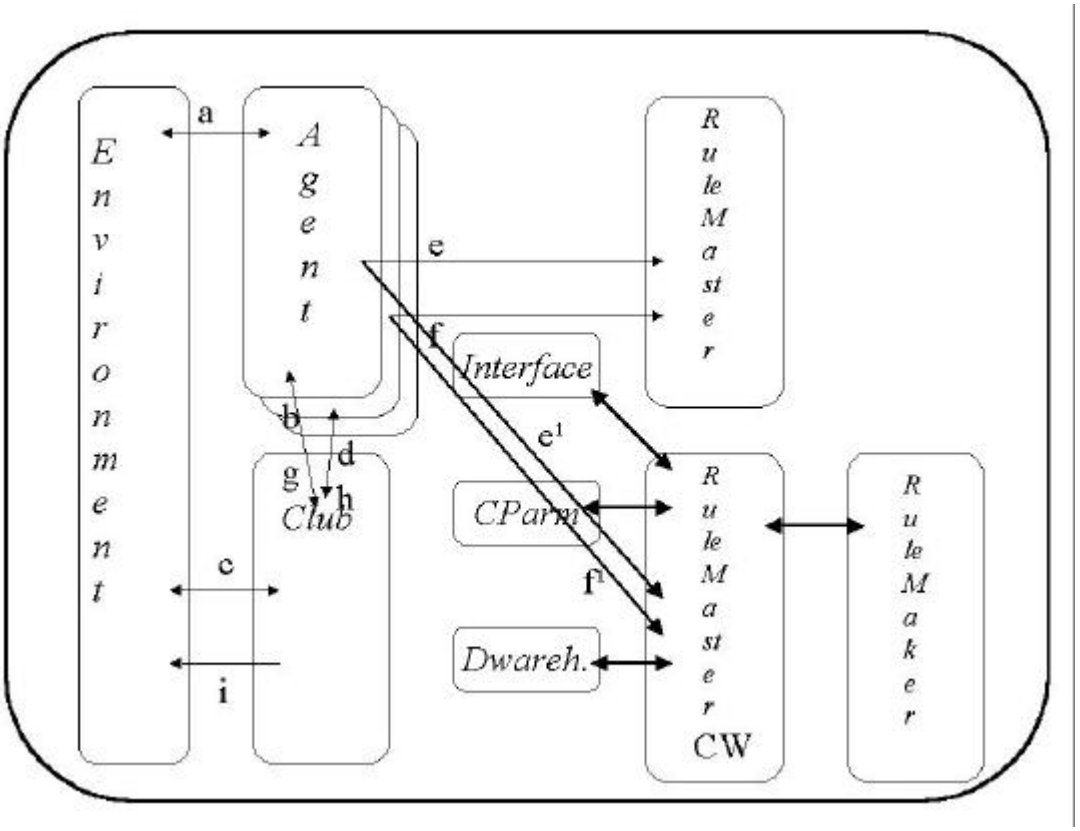
Treasury. La treasury è la controparte contabile dei movimenti di crediti effettuati durante l'elaborazione: fornisce il numerario per pagare le ricompense, incassa le imposte e le ricompense verso la routine di detecting.

5.2 Implementazioni della struttura precedente

Dopo aver descritto (nel precedente capitolo) la struttura del modello ad agenti indipendenti e nel precedente paragrafo la struttura del Classifier CW utilizzato vediamo ora come è stato costruito il modello che mette insieme questi due elementi. Il modello risultante dall’inserimento del Classifier CW nel modello preesistente permetterà di confrontare comportamenti determinati da regole precise con altri che, al contrario, derivano da regole adattative e che possono sviluppare strategie di gioco non prevedibili a priori.

Per iniziare la descrizione del modello risultante si può partire dal commento dello schema successivo, che ne illustra le parti principali.

Come si può notare, sono state mantenute tutte le notazioni di cui al capitolo 4 poiché la struttura del modello e quindi dei possibili tornei è rimasta assolutamente inalterata. Sia il modello ad agenti sia il Classifier CW sono stati costruiti secondo lo schema ERA; di conseguenza è stato possibile inserire uno nell'altro senza modificare la struttura del modello base, senza modificare il funzionamento del classifier e con poche modifiche agli elementi costitutivi. Modifiche dovute essenzialmente al fatto che entrambe i modelli usavano gli stessi nomi di classe interface e ruleMaster.



Le differenze di questo modello rispetto al precedente riguardano essenzialmente i seguenti aspetti:

- lo scopo del torneo
- l'utilizzo della classe interface
- l'inserimento di nuove classi di oggetti
- la dinamica delle chiamate al ruleMaster.

Scopo del torneo. Mentre il modello descritto nel capitolo precedente aveva lo scopo di riprodurre le condizioni originali del torneo di Axelrod (1980a, 1980b) e di controllare la robustezza della strategia vincente Tit for Tat al cambiare delle condizioni di "rumore" nel torneo, ora lo scopo del modello è diventato verificare se il Classifier System implementato nel modello è in grado di trovare autonomamente

strategie vincenti o diverse da quelle inventati dai partecipanti “umani” al torneo originale di Axelrod.

Diventa perciò essenziale prevedere una serie di incontri che non hanno lo scopo di determinare il vincitore ma di consentire al Classifier System di imparare, di definire la sua propria strategia, leggibile attraverso le regole presenti nella rule list e la loro forza relativa.

Il modello precedente era strutturato per definire un numero massimo di incontri (vedi capitolo 4 paragrafo 2 il parametro maxNumberOfMatches) giunti al quale si stilava la classifica del torneo; per consentire al classifier di imparare liberamente si è introdotto un nuovo parametro di input “reuse”. Se il valore di questo parametro di input è a zero, il modello si comporta esattamente come prima, cioè si ferma dopo un certo numero di cicli secondo la logica descritta nel paragrafo 4. Se, al contrario è impostato a uno, ripete ciclicamente il torneo realizzando lo scopo di far imparare il Classifier System. I parametri diventano quindi (in grassetto il parametro nuovo)

(dal metodo createBegin di ModelSwarm)

```
// Now fill in various simulation parameters with default values.
  obj->worldXSize             = 50;
  obj->worldYSize             = 50;
  obj->times                  = 200;
  obj->sensitivity            = 5;
  obj->distance               = 0;
  obj->numberOfInteractiveAgents = 0;
  obj->numberOfReactiveAgents    = 63;
  obj->behaviour              = 0;
  obj->noise                  = 0;
  obj->reuse                  = 0;
// the number of single matchs for a round robin tournament between
// N=numberOfReactiveAgents (metch and return match) is:
// N*(N-1)
  obj->maxNumberOfMatches       = 200;
```

Gli altri parametri non vengono qui commentati essendo esattamente gli stessi già visti nel capitolo 4.

Interface. La classe interface, già presente nel modello, è anche usata dal Classifier System CW. L’utilizzo è assolutamente identico, poiché realizza in entrambe gli ambienti una funzione di “traduzione” tra la metrica degli agenti e quella dei ruleMaster. Si è perciò deciso di fondere le due classi interface in una sola che comprendesse tutti i metodi necessari per le “tradizioni” verso i due diversi ruleMaster; le funzioni di questa classe sono l’unione delle funzioni che avevano nei due modelli. Accanto ai metodi tipici del Classifier System si trovano quelli sviluppati per il modello ad agenti indipendenti.

Nuove classi. Il ruleMaster è l’oggetto ai cui gli agenti chiedono quale regola applicare, cioè quale azione comunicare all’ambiente. Anche in questo caso, come per la classe Interface, il ruleMaster sviluppato per il modello ad agenti indipendenti e quello del Classifier CW realizzano la stessa funzione. Si è però scelto di tenere separate le due classi.

In primo luogo perché, se è vero che entrambe "consigliano" un'azione agli agenti, è anche vero che lo fanno in modo completamente diverso. Uno applicando regole precostituite, l'altro attivando procedure adattative.

Secondariamente perché il numero delle righe di codice coinvolte (i file .m dei ruleMaster, cioè quelli che sostanzialmente contengono solo istruzioni, sono di circa 4800 istruzioni per uno e di circa 700 per l'altro) consiglia di tenere separate le due classi.

Si è perciò creata la classe RuleMasterCW che altro non è che la classe RuleMaster del Classifier System CW rinominata. Naturalmente si sono dovute anche introdurre tutte quelle classi necessarie al funzionamento del Classifier System CW. Il ModelSwarm crea ora anche gli oggetti delle classi:

- ClassifierParm
- DataWarehouse (evidenziate anche nello schema precedente)
- Dump
- Effector
- Message
- Rule
- Treasury
- Trace (non riportate nello schema per semplicità di esposizione).

Per lo schema completo e dettagliato di funzionamento del Classifier System CW si rimanda a Ferraris (1999).

Dinamica. Durante la creazione degli agenti si generano tutti gli oggetti necessari per il funzionamento di un agente sia come agente a regole determinate sia come agente che utilizza il classifier. Ad ogni agente viene associato un valore della variabile behaviour; è infatti attraverso questo valore che si sceglie il metodo del ruleMaster e cioè la strategia. Si è perciò aggiunto al ruleMaster

- il metodo performBehaviour70. Questo metodo non costituisce, in realtà, una strategia nuova, ma effettua la chiamata al ruleMasterCW.

Gli agenti possono chiamare il "vecchio" ruleMaster (freccia **e** nello schema) o chiamare il ruleMasterCW del Classifier Sistem (freccia $\mathbf{e^1}$ nello schema). Il fatto che questa seconda chiamata avvenga per tramite del ruleMaster non è evidenziato in figura per semplicità di esposizione. Introdurre questo tramite, che non ha alcun significato strutturale nel modello, ha permesso di ridurre al minimo gli interventi di modifica del codice degli agenti. Analogamente possono fare gli agenti avversari (opponent; frecce **f** e $\mathbf{f^1}$ nello schema).

I tornei, che verranno dettagliatamente descritti nei paragrafi successivi, sono stati organizzati in questo modo. La prima fase è rappresentata dal training del Classifier e viene realizzato mediante un torneo con un solo agente interattivo (che utilizza il metodo performBehaviour70, cioè attiva il Classifier System) e un certo numero di agenti reattivi che possono attivare tutti la stessa strategia (il Classifier impara in un mondo in cui tutti gli agenti si comportano nello stesso modo), o strategia diverse.

Per valutare l'efficacia dell'apprendimento si sono utilizzati due metodi diversi. Il primo è il confronto con una strategia nota e vincente nelle stesse condizioni: si ripete perciò la simulazione imponendo all'agente interattivo di utilizzare, invece del Classifier System CW, una strategia nota. Dopo di che si

mettono a confronto i risultati del Classifier System con la strategia nota. Il secondo è “bloccare” l’apprendimento del Classifier System, inserirlo nel torneo e controllarne i risultati.

5.3 La definizione del metodo di reward

La prima serie di esperimenti, i cui risultati sono raccolti nei grafici presentati in questo paragrafo, hanno un duplice obiettivo:

- controllare la capacità di apprendimento del Classifier System CW inserito nella struttura del torneo
- verificare quale, tra i metodi di reward possibili, è il più efficiente nell’indirizzare l’apprendimento.

Oltre al metodo di reward occorre anche definire quali sono gli eventi che dal modello vengono trasferiti all’interno del CW attraverso i detector. Per questi primi esperimenti la regola nella condition part è composta di 6 posizioni (6 bit) ciascuno dei quali riporta direttamente la storia recente dell’incontro tra CW e l’agente contro cui sta giocando. Tre bit (si ricorda che nel modello 0 rappresenta la cooperazione e 1 la defezione) sono le tre mosse precedenti di CW stesso e agli altri tre bit sono le tre mosse precedenti dell’agente avversario.

La ricompensa che totalizza l‘azione decisa da CW non può, ovviamente, che dipendere dal punteggio ottenuto mediante l’applicazione della tabella degli score che rappresenta il gioco del Dilemma del Prigioniero in forma strategica; viene quindi assegnato il punteggio massimo di 5 se il giocatore tradisce mentre l’avversario coopera, 3 in caso di reciproca cooperazione, 1 in caso di reciproco tradimento e 0 se si coopera mentre l’avversario tradisce. Questo punteggio è, in realtà, l’unico feedback che il modello fornisce per valutare la bontà dell’azione decisa: la valutazione del proprio avversaria avviene attraverso la sequenza delle azioni.

Come utilizzare tecnicamente questo feedback? Si sono sperimentate tre diverse soluzioni.

- Utilizzo del punteggio tale e quale, passando al classifier system CW come reward, il punteggio ottenuto diviso per 5 in modo da ottenere un valore compreso tra 0 e 1 (nei grafici è identificato come “rough”).
- Utilizzo di una valutazione mediata del punteggio ottenuto, in modo da seguire, in qualche modo, la logica del “win-stay, lose-shift”. In questo modo, se il punteggio ottenuto è 0 o 1 si è in un a situazione di “lose” è il reward comunicato a CW è zero, ovvero di valutazione negativa dell’azione comunicata all’ambiente da CW. Se viceversa il punteggio ottenuto è 3 o 5, si comunica un reward di uno, ovvero di valutazione positiva dell’azione. In questo caso vengono volutamente eliminate le sfumature di valutazione per passare ad un feedback di soli due valori (nei grafici è identificato come “win-stay”).
- Confronto del punteggio ottenuto con il valor medio dei punteggi fino a quel momento; se è superiore il reward è uno altrimenti è zero. Nuovamente la logica

del feedback è a due valori; in questo caso la soglia che fa passare da una valutazione dell'azione negativa a una positiva non è un valore prefissato ma il confronto con i risultati conseguiti fino a quel momento (nei grafici è identificato con "mean") .

La prima prova è stata effettuata strutturando il torneo tra il classifier system CW e alcuni agenti che usano tutti la medesima strategia. Per prima usata è stata Tit for Tat. Come si può vedere dai grafici seguenti, tutti e tre i metodi di reward sono in grado di indirizzare l'apprendimento di CW verso la definizione di un comportamento che porta al raggiungimento del punteggio di cooperazione stabile con Tit for Tat (150 punti nei primi tre grafici, ottenibili da un torneo di 50 iterazioni in cui entrambe glli agenti cooperano sempre). Talvolta la definizione delle regole del classifier è talmente rapida che già alla fine del primo torneo CW riesce ad ottenere il punteggio massimo possibile. Il processo di apprendimento più rapido sembra essere quello ottenibile con un reward "win-stay", seguito da "rough" e da "mean". La "velocità di apprendimento" non è però un parametro significativo, poiché può variare anche molto a secondo di come avviene la generazione casuale delle regole iniziali.

Qui grafico excel file: Grafico5_01.pdf
Qui grafico excel file: Grafico5_02.pdf
Qui grafico excel file: Grafico5_03.pdf

La seconda prova è stata effettuata tra CW e una strategia creata ad hoc chiamata "do the opposite". Questa strategia non è stata utilizzata nel torneo ma solo per le fasi di messa a punto del modello; propone, come propria azione, semplicemente l'azione contraria rispetto a quella fatta dall'avversario alla mossa precedente. E' quindi una strategia particolarmente semplice da battere, poiché, se si tradisce sempre, questa coopererà sempre, e si otterrà il massimo punteggio ottenibile. In questo fase il torneo era di 50 iterazioni e perciò il punteggio massimo ottenibile di 250 punti.

Come si può vedere dai grafici la capacità di arrivare alla soluzione del problema di determinare la migliore strategia contro "do the opposite" dipende fortemente dal metodo di reward.

Qui grafico excel file: Grafico5_04.pdf
Qui grafico excel file: Grafico5_05.pdf
Qui grafico excel file: Grafico5_06.pdf

Se si adopera il metodo "rough" CW si stabilizza su soluzioni non ottimali in due prove su tre. Con questo metodo il classifier system ottiene un reward superiore a zero sia che ottenga 1 sia che ottenga 3 o 5. Questo non consente di indirizzare CW verso la soluzione migliore.

Con il metodo di reward "win-stay" in nessuna delle tre prove e nonostante siano stati svolti 400 tornei, il classifier system è riuscito a determinare la strategia vincente.

L'unico metodo che si è mostrato efficace, anche se non particolarmente rapido, è stato "mean", ovvero il confronto con il valore medio dei punteggi ottenuti fino a quel momento.

Si è perciò stabilito di utilizzare questo metodo di reward in tutte le successive prove effettuate con questo modello.

5.4 I risultati dei tornei a 63/5 partecipanti

Definito il metodo di reward, si è utilizzato il modello in modo da ottenere l'esecuzione ripetuta di un torneo strutturato in modo da avere come unico agente interattivo il Classifier System CW e i 63 agenti reattivi già utilizzati nei precedenti esperimenti. In questo modo, il torneo è semplicemente il modo in cui il Classifier System CW può evolvere la propria strategia per giocare il "Dilemma del Prigioniero" nell'ambiente definito dagli altri 63 agenti indipendenti. I risultati ottenuti da CW sono stati confrontati con i risultati ottenuto da Tit for Tat in situazione esattamente analoga, ovvero in un torneo in cui vi è un solo agente interattivo (Tit for Tat) e 63 agenti reattivi. I risultati sono rappresentati in grafico come valore medio di un singolo incontro; in questo modo si possono confrontare direttamente tornei che si sono svolti con un numero di ripetizioni del Dilemma del Prigioniero diverse. Come si può vedere dal grafico. I risultati di Tit for Tat sono piuttosto stabili; perciò, negli esperimenti successivi, si è talvolta utilizzato come parametro di confronto il valore medio del punteggio ottenuto da Tit for Tat, calcolato su tutti i tornei eseguiti. Le due serie di risultati rappresentano i punteggi ottenuti da CW in due diversi esperimenti.

Qui grafico excel file: Grafico5_07.pdf

In grafico vengono riportati solamente i risultati degli ultimi tornei, fino al torneo numero 289. Come si può chiaramente osservare dal grafico, i punteggi ottenuti dal Classifier CW sono:

- sensibilmente e stabilmente inferiori a quelli ottenuti da Tit for Tat
- non presentano nessun andamento convergente verso un risultato stabile, anche subottimale.

Il numero di "chiamate" al Classifier CW in un torneo strutturato come descritto e che viene ripetuto 289 volte è

289 x 63 (agenti) x 200 (iterazioni) = 3.641.400,

un numero di chiamate molto più grande di quelle che sono state necessarie, nel peggiore dei casi, nei tornei visti nel paragrafo precedente. Le condizioni sperimentali si possono quindi riassumere in 4 punti:

- l'ambiente fornisce a CW 6 informazioni, che sono le tre mosse precedenti proprie e le tre precedenti dell'avversario
- il reward è positivo se il punteggio ottenuto dall'azione (0, 1, 3 o 5) supera il valore medio del punteggio fino a quel momento ottenuto

- CW gestisce tra le 20 e le 50 regole (a seconda degli esperimenti)
- Deve confrontarsi con 63 agenti indipendenti diversi.

In queste condizioni sperimentali, si deve concludere che CW non è ingrado di sviluppare autonomamente una strategia di gioco.

In appendice B è riportato l'output generato dall'ultimo torneo della prova indicata come CW1 nel grafico. Per comodità di commento ne vengono qui riprodotte alcune righe.

```
…
NAg;100;600;10963320;NOp;5;600;0;
NAg;100;600;10963920;NOp;6;600;0;
NAg;100;600;10964520;NOp;7;600;0;
NAg;100;600;10965120;NOp;8;600;0;
NAg;100;600;10965720;NOp;9;600;0;
NAg;100;600;10966320;NOp;10;600;0;
NAg;100;600;10966920;NOp;11;600;0;
NAg;100;600;10967520;NOp;12;600;0;
NAg;100;600;10968120;NOp;13;600;0;
NAg;100;600;10968720;NOp;14;600;0;
NAg;100;600;10969320;NOp;15;600;0;
NAg;100;600;10969920;NOp;16;600;0;
NAg;100;600;10970520;NOp;17;600;0;
…
```

Questo è il record scritto dal modello ad agenti indipendenti quando tutte le iterazioni tra due agenti sono terminate. Gli elementi costituenti il record sono:

- NAg che è semplicemente un flag per semplificare la lettura e che introduce i dati dell'agente
- 100 è il numero che identifica l'agente, in questo esperimento è CW
- 600 è il punteggio ottenuto per le 200 iterazioni previste
- 10970520 è il punteggio complessivo ottenuto fino a quel momento da CW
- NOp è il flag che introduce i dati dell'opponent
- 17 è il numero che identifica l'agente che in quel momento è opponent
- 600 è il punteggio ottenuto per le 200 iterazioni previste
- 0 indica il campo originariamente previsto per il punteggio complessivo dell'opponent ma non implementato

Le altre segnalazioni provengono invece direttamente dal Classifier CW e sono dettagliatamente descritte in Ferraris (1999). I messaggi qui riportati sono emessi per ultimi e riferiscono dati sul funzionamento interno di CW.

```
Statistics:
Rule's selection has been performed 4120200 times
CoverDetector has been applied 54855 times
CoverEffector has been applied 0 times
Rule's genetic evolution has been performed 273700 times
```

Da qui si può osservare come il classifier abbia operato 4.120.200 volte; questo valore è superiore a quello riportato in grafico poiché l'altra prova terminava al valore prima citato e si è preferito strutturare il grafico in modo che entrambe le prove avessero tutti i dati relativi.

Le altre segnalazioni si riferiscono alle singole regole attive in quel momento. Sono qui di seguito riportate quelle relative a due regole particolarmente significative. La prima stringa di 6 bit rappresenta la condition part; leggendo la stringa da sinistra a destra si hanno i seguenti significati:

l'azione di CW decisa 3 iterazioni prima (m3)
l'azione di CW decisa 2 iterazioni prima (m2)
l'azione di CW decisa 1 iterazioni prima (m1)
l'azione dell'opponent di 3 iterazioni prima (y3)
l'azione dell'opponent di 2 iterazioni prima (y2)
l'azione dell'opponent di 1 iterazioni prima (y1).

La seconda stringa è la action part e l'unico bit significativo è quello più a destra: 0 per cooperazione 1 per defezione.

Le due regole qui riportate rappresentano quindi Tit for Tat. Se il mio avversario ha tradito la volta precedente io tradisco.

…
My partnerList contains 0 entries for:
4B42EA0 strength: 0.565343 specificity 0.833333
00##01
000001
…

Se il mio avversario ha cooperato la volta precedente io coopero.

…
My partnerList contains 0 entries for:
4BCAC68 strength: 0.866252 specificity 0.833333
00##00
000000
…

Esaminando però tutte le regole prodotte da CW la situazione diventa meno chiara. Accanto alle due regole riportate sopra, ve ne sono altre che a fronte di una collaborazione rispondono con una defezione.

…
My partnerList contains 0 entries for:
4BC7B78 strength: 0.927960 specificity 0.833333
01##00
000001
…

Oppure che rispondono ad una defezione con una collaborazione.

…
My partnerList contains 0 entries for:
4B47750 strength: 0.927960 specificity 0.833333
00##01
000000
…

E' da notare come queste ultime due regole abbiano una forza (strength) superiore alle due viste prima. Le altre regole hanno forza molto inferiore a queste viste e questo vuol dire che sono state meno applicate (hanno perso forza per via della tassazione di inattività –paramentro lifetax) o che hanno prodotto azioni inefficaci.

Certamente CW impara a collaborare (come è anche ben evidenziato dalla

serie di risultati complessivi di 600 punti riportati prima e ottenibili solamente con 200 cooperazioni reciproche); impara a collaborare molto velocemente. Infatti dall'esame dell'output dei primi tornei risulta come CW sia in grado di generare collaborazione da parte di Tit for Tat e di molti altri agenti già nei primissimi tornei. Viceversa vi sono strategie che CW non riesce ad affrontare efficacemente.

NAg;100;155;10972608;NOp;22;600;0;	CW perde 155 a 600
NAg;100;15;10973099;NOp;24;950;0;	CW perde 15 a 950
NAg;100;110;10978653;NOp;36;715;0;	CW perde 110 a 715
NAg;100;11;10981051;NOp;43;986;0;	CW perde 11 a 986

Come ve ne sono altre contro le quali riesce ad ottenere punteggi molto elevati.

NAg;100;1000;10975534;NOp;29;0;0;	CW vince 1000 a 0

E' possibile che la variabilità dei risultati evidenziata derivi dal fatto che il classifier CW incontra, a rotazione, tutti gli agenti e per ciascuno di essi cerca di definire la strategia migliore. Ma il parametro di life tax fa decadere la forza delle regole non utilizzate e quindi, se vi sono regole applicate raramente, possono perdere di forza ed essere eliminate.

Come semplificare il torneo senza perdere troppo di generalità? Utilizzando i programmi SAS listati in appendice A, si sono effettuate alcune analisi di regressione statistica sulla matrice dei punteggi 189 x 63 ottenuta dai punteggi di più tornei. In particolare le prime 63 x 2 righe rappresentano due tornei mentre le ultime 63 righe sono i risultati medi dei 5 tornei descritti nel paragrafo 4.3.

Senza entrare nell'analisi dettagliata dell'output prodotto si possono riassumere i risultati nelle seguenti osservazioni:

- esiste una forte correlazione tra i risultati ottenuti contro alcune strategie
- anche con un numero ridotto di agenti (5 ad esempio) è possibile spiegare oltre il 90% dei risultati ottenuti dalla strategia in esame
- l'identificazione degli agenti più significativi da considerare è variabile a seconda dei metodi usati, probabilmente a causa delle correlazioni evidenziate prima.

Per dare fedelmente conto del comportamento di CW rispetto all'intero insieme dei 63 agenti è possibile utilizzare solamente 5 partecipanti. Queste 5 strategie possono essere pensate come rappresentanti l'intero insieme, in quanto i punteggi realizzati da una data strategia contro queste cinque possono essere utilizzati per stimare il punteggio in media di quella strategia per l'intero insieme. Il valore di r^2 che si ha stimando i risultati del torneo con i risultati ottenuti contro queste 5 strategie è di 0.96. Ciò significa che il 96% della varianza nei punteggi di gara si spiega conoscendo il risultato conseguito da CW nell'incontro con le sole cinque rappresentanti.

Le regole scelte sono risultate:

la K91r di J.Pinkley	nel modello è il metodo performBehaviour0
la K40r di R. Adams	nel modello è il metodo performBehaviour2
la K67r di C. Feathers	nel modello è il metodo performBehaviour4

la K60r di Graaskamp & Katzen	nel modello è il metodo performBehaviour10
TESTER di D. Gladstein	nel modello è il metodo performBehaviour20.

Il torneo è stato perciò organizzato in modo da avere CW come unico agente interattivo che, giocando contro i 5 agenti che attuano le strategie sopra elencate, elabora nuove strategie per giocare il Dilemma del Prigioniero ripeturo.

Senza più riportare in grafico i risultati dei tornei di CW contro i 5 agenti se ne riassumono qui le conclusioni.

- Anche con un numero di chiamate molto elevato al Classifier CW (in alcuni tornei si è andato oltre 3 milioni e mezzo di chiamate) questi non riesce ad elaborare una strategia stabile.
- I risultati sono molto variabili; il rapporto tra il punteggio minimo ottenuto e quello massimo è di oltre 1 a 2.

5.5 I risultati dei tornei contro TESTER

L'analisi dei punteggi ottenuti contro le singole 5 strategie viste nel paragrafo precedente ha però evidenziato che la maggior variabilità di risultato era causata dai punteggi ottenuti giocando contro l'agente che effettua la strategia TESTER (già descritta nel capitolo 4, metodo performBehaviour20).

Per approfondire l'esame del comportamento di CW contro questa strategia, si è ulteriormente modificato il torneo, facendo giocare CW sempre contro 5 agenti, ma, questa volta, tutti quanti operano secondo la strategia TESTER.

I risultati non sono riportati in grafico poiché non significativi. Le condizioni di torneo sono:

- CW si confronta contro 5 agenti indipendenti ciascuno dei quali gioca la stessa strategia TESTER
- Il numero delle iterazioni contro ciascun agente è di 50
- Il reward è definito per confronto con il valore medio ottenuto fino a qule momento.

Ebbene, a queste condizioni, il Classifier CW non riesce ad elaborare una strategia per contrastare TESTER.

Anche se già descritto nel cap.4, si ricorda che il comportamento di TESTER è basato su poche regole semplici:

- TESTER parte sempre defezionando
- poi coopera per due volte di seguito
- successivamente decide se tradire o cooperare in base al rapporto di cooperazioni sul totale del suo avversario.

Poiché alcuni tratti caratteristici di questa strategia si manifestano nelle prime iterazioni, invece di istruire CW contro 5 agenti per 50 iterazioni e in questo modo ripetere più sovente le condizioni iniziali, si è ristrutturato il torneo in modo da avere 1 solo agente per 350 iterazioni. I risultati sono riportati in grafico.

Qui grafico excel file: Grafico5_08.pdf

Dall'esame dei messaggi emessi da CW e riportati in appendice B si possono ricavare alcune considerazioni.

- La maggior parte delle regole ha forza zero, cioè la strategia elaborata si basa su poche regole definite
- Le regole che hanno forza diversa da zero formano la strategia Tit for Tat
- La regola di ritorsione, cioè quella che prevede la defezione di CW a fronte della defezione di TESTER ha forza molto debole (negli altri due casi ha addirittura forza a zero)

Certamente il Classifier System CW è molto sensibile alle situazioni sperimentali. Infatti il torneo a 5 agenti per 50 iterazioni attiva CW un numero di volte simile a quello del torneo ad un agente solo ma con 350 iterazioni; eppure i risultati sono molto diversi. Questo, in generale, può diminuire l'applicabilità di questo strumento, poiché se non si conosce in anticipo l'ambito sperimentale è difficile dire se il Classifier non ha trovato soluzioni perché non ci sono o perché i parametri non sono stati opportunamente impostati.

Per cercare un modo per ovviare ai problemi sollevati dalla sensibilità mostrata da CW rispetto al numero di iterazioni si sono modificate le regole del classifier in modo che fosse possibile far arrivare dall'ambiente altre informazioni oltre a quelle già viste. Occorre qui ricordare che il numero di iterazioni è sempre stato mantenuto ignoto agli agenti e, almeno in linea di principio, determinato casualmente, in modo da impedire "tradimenti dell'ultimo colpo" come più diffusamente specificato nei capitoli 2 e 4.

Il classifier è stato quindi modificato nelle sue regole che diventano di 18 bit e che consentono di codificare le seguenti informazioni ambientali.

- Bit [0] – mossa dell'avversario al turno precedente
- Bit [1] – mossa dell'avversario 2 turni prima
- Bit [2] – mossa dell'avversario 3 turni prima
- Bit [3] – mossa di CW al turno precedente
- Bit [4] – mossa di CW 2 turni prima
- Bit [5] – mossa di CW 3 turni prima
- Bit [6] [7] [8] – 8 classi di valori per il rapporto tra numero di cooperazioni e giocate totali dell'avversario
- Bit [9] [10] [11] – 8 classi di valori per il rapporto tra numero di cooperazioni e giocate totali di CW
- Bit [12] [13] [14] – 8 classi di valori per il punteggio medio dell'avversario
- Bit [15] [16] [17] – 8 classi di valori per il punteggio medio di CW

Si è tornati alla struttura di torneo originario, cioè CW contro 5 agenti che giocano tutti la strategia TESTER per 50 iterazioni a incontro. I risultati sono evidenziati nel grafico seguente.

Qui grafico excel file: Grafico5_09.pdf

Come si può facilmente vedere dall'andamento del punteggio, CW riesce ad elaborare una strategia che ottiene quasi lo stesso risultato di Tit for Tat (148 punti per 50 iterazioni mentre Tit for Tat ne totalizza 149). Per capire quale tipo di strategia viene trovata occorre esaminare i messaggi emessi dal classifier e che sono riportati, solo per l'ultimo torneo, in appendice B. Le uniche regole che hanno forza diversa da zero sono quelle riportate qui di seguito.

My partnerList contains 0 entries for:
4A1BAC8 strength: 0.870845 specificity 0.861111
010010##1#0#1#0110
000000000000000000
My partnerList contains 0 entries for:
4A1E4A8 strength: 0.853515 specificity 0.861111
000000##0#0#0#0000
000000000000000001
My partnerList contains 0 entries for:
4A11BF8 strength: 0.868189 specificity 0.861111
000000##1#1#0#1001
000000000000000000
My partnerList contains 0 entries for:
4A1A118 strength: 0.869336 specificity 0.916667
001001101#0#1#0110
000000000000000000
My partnerList contains 0 entries for:
4A17D98 strength: 0.852036 specificity 0.916667
100100000#0#1#0100
000000000000000000

Dall'esame di queste regole si possono trarre alcune semplici conclusioni:

- la strategia trovata NON è Tit for Tat (manca la defezione in seguito alla defezione dell'avversario)
- la strategia trovata prevede cooperazione anche quando TESTER ha defezionato purchè ci sia defezione anche da parte di CW (vedi ultima regola della lista)
- le altre informazioni aggiunte (oltre alle tre mosse precedenti) non sono molto usate (molti i caratteri #) con l'eccezione del proprio score medio
- ottiene un punteggio (148 contro 149) leggermente inferiore di Tit for Tat
- l'unica ritorsione prevista è quando CW è nella classe di punteggio medio più bassa, dopo una sequenza di mosse di cooperazione (che possono indicare anche la prima mossa).

5.6 I risultati dei tornei a 5 partecipanti: CW riceve tutte le informazioni ambientali disponibili

Come si è visto nel paragrafo precedente, è stata dimostrata la possibilità di determinare strategie che si basano anche sulle altre informazioni ambientali disponibili.

Per studiare l'effetto dell'introduzione di maggiori informazioni nel Classifier CW si è modificato il torneo in questo modo:

- CW contro 5 agenti che giocano secondo le 5 strategie definite precedentemente dallo studio sulla regressione
- Il numero delle iterazioni è fissato a 500
- Il reward è mediante confronto con il punteggio medio fino a quel momento.

I risultati di un esperimento sono mostrati nel grafico seguente.

Qui grafico excel file: Grafico5_10.pdf

Come si può vedere dal grafico, il Classifier non si stabilizza su alcun punteggio, indice che non riesce a determinare un strategia ben definita. I messaggi emesse dal modello e da CW nell'ultimo torneo dell'esperimento rappresentato nel grafico sono riportati in appendice B.

Un'ultima serie di prove, sempre con tornei a 5 agenti, è stata fatta per verificare se, modificando i valori dei parametri qui di seguito elencati si ottenenvano comportamenti diversi. Le variazioni di volta in volta sperimentate sono state quelle qui di seguito elencate.

- E' stato variato il valore di LIFETAX, cioè di quanto vengono penalizzate le regole inattive. Il valore fino ad ora utilizzato era di 0,01; sono stati provati valori di 0,001 0,0001 0,00001 0,00005.
- È stata attivata la possibilità di emettere messaggi "interni" al Classifier stesso
- E' stato modificato il numero di regole presenti nel Classifier.
- Sono state scelti diversi valori per il numero di iterazioni che compongono un singolo incontro.
- La condition part delle regole è stata ulteriormente estesa fino a raggiungere i 24 bit e comprendere anche il numero dell'agente contro cui CW si sta incontrando.

Nessuna delle prove effettuate ha dato esiti positivi. Vengono qui di seguito proposti due grafici come esempio, poiché tutti i risultati mostrano un andamento assolutamente identico. Il Classifier CW non sembra in grado di stabilizzarsi su una strategia definita.

Perché il Classifier System è in grado di elaborare strategie in un torneo contro una sola strategia mentre sembra incapace di fare la stessa cosa in un torneo con più strategie presenti contemporaneamente?

La risposta può venire dalla definizione stessa di gioco strategico ricordata nei capitoli precedenti. Un gioco si può definire strategico quando occorre tener conto non solo delle proprie e altrui azioni ma anche delle ipotesi che noi e i nostri avversari facciamo sul gioco stesso.

Il Classifier System ha necessità, per funzionare correttamente, di ricevere un feedback dall'ambiente **ad ogni azione che compie**. Mentre il risultato che si vuole ottenere sono buoni risultati di torneo. Dovrebbe essere possibile valutare il Classifier System sui risultati del torneo, ma la struttura ed il funzionamento interno non consentono di differire il reward a fine torneo; non si saprebbe infatti a quale regola assegnarlo. La stessa efficienza nel determinare soluzioni tatticamente efficaci (situazione simile a quella di un torneo contro una sola strategia) rende estremamente complessa la ricerca di soluzioni strategicamente valide.

Qui grafico excel file: Grafico5_11.pdf
Qui grafico excel file: Grafico5_12.pdf