Relazione del programma in NetLogo

LA JAK BANK

"Un esempio di finanza etica in Svezia"

Prof. Pietro TERNA

Anno Accademico 2009/2010

Laura Cosa
Livio Nervo
Federico Volpe

CHE COS'E' LA "JAK BANK" E COME FUNZIONA IL SISTEMA JAK

L'associazione cooperativa Jord Arbejde Kapital è stata fondata in Danimarca durante la Grande Depressione nel 1931. L'associazione mise in circolazione una Valuta locale che è stata successivamente dichiarata fuori legge dal governo danese nel 1933. Nel 1934 la JAK danese fondò un sistema di risparmio e prestito senza interesse e un sistema locale di commercio e scambio di beni. Sebbene entrambi i sistemi furono costretti a chiudere, il sistema di prestito e risparmio riemerse nel 1944.

L'esperimento della banca JAK Danese ispirò un gruppo in Svezia che fondò un' associazione no-profit chiamata Jord Arbete Kapital - Riksfrening fur Ekonomisk Frigrelse (Associazione Nazionale per l'Emancipazione Economica) nel 1965, che sviluppò il sistema matematico basato sui punti di risparmio chiamato "sistema di risparmio bilanciato". L'associazione crebbe lentamente nel primo periodo e ricevette la licenza bancaria dall'Autorità di Vigilanza Finanziaria Svedese solo alla fine del 1997.

La banca etica JAK (JAK Medlemsbank) è una banca cooperativa con sede a Skovde, Svezia. Si tratta di una banca realmente posseduta dai suoi soci: ciascuno detiene una sola azione e ha lo stesso peso decisionale nell'annuale votazione del consiglio direttivo. JAK è un acronimo che sta per Jord Arbete Kapital, in svedese Terra Lavoro Capitale. Per Terra si intendono le risorse della natura, per Lavoro la risorsa data dal lavoro umano, per Capitale le infrastrutture create dall'uomo che permettono di potenziarne l'efficienza. Sono i tre principi chiave dell'economia reale. Su queste tre fondamenta si basa JAK, una banca che cerca di proporre un'alternativa concreta, in alcuni segmenti di mercato, al metodo di finanziamento offerto dal sistema bancario tradizionale. L'obiettivo della banca non consiste nel trarre profitto dal suo servizio; infatti come le Banche Islamiche la banca JAK non carica (paga) interessi sui suoi prestiti (risparmi); si cerca, invece, di creare una società interest-free. Tutte le

attività della banca avvengono fuori dal mercato finanziario poiché i suoi prestiti sono finanziati solamente dai risparmi dei soci.

In particolare la filosofia JAK si pone alla base il concetto secondo cui l'instabilità economica è la diretta conseguenza di un'economia basata sulla rendita da interesse e le premesse su cui si fonda il sistema sono le seguenti:

- ✓ la rendita da interesse è nemica di un'economia stabile
- ✓ l'interesse causa disoccupazione, inflazione e distruzione dell'ambiente
- ✓ l'interesse sposta matematicamente denaro dai poveri verso i ricchi
- ✓ l'interesse favorisce progetti che tendono a raggiungere alti profitti in poco tempo

Il sistema JAK si basa sul fatto che una persona in periodi diversi della propria vita può trovarsi ad essere sia risparmiatore che bisognoso di prestiti, e che se si riesce a bilanciare il flusso nel tempo tra debiti e crediti di ogni risparmiatore si può creare un sistema in cui chi oggi deposita i risparmi potrà domani ricevere un prestito. In pratica, chi deposita i propri risparmi presso la banca JAK ha un interesse nullo sul proprio conto ma acquisisce dei punti in base alla quantità e alla durata del suo prestito. Quando sarà lui ad avere bisogno di soldi, in base ai punti accumulati, potrà chiedere un prestito a condizioni vantaggiose. Il giornalista Giorgio Simonetti racconta così il funzionamento di questa rivoluzionaria implementazione dell'idea di banca etica: "*Un gruppo di 35.000 persone immagina una società libera dal concetto di interesse. Il ragionamento che hanno fatto è questo: se mettiamo dei soldi in banca, parliamo di persone con redditi medio bassi, cioè la maggior parte di noi, gli interessi che percepisco servono sì e no a pagare le spese. Però se chiedo un prestito mi spennano. Allora perchè non mettere il nostro denaro insieme e farlo circolare?*"

Nell'attuale sistema l'interesse è sostituito da un sistema di pagamento del servizio del prestito, con lo scopo di creare uno strumento finanziario vantaggioso per i propri soci, sostenibile per l'ambiente e al servizio delle economie locali.

I costi amministrativi e di sviluppo sono pagati dalla *quota associativa* (di circa 23 euro all'anno) e dalla *tassa sul prestito*, che è pari ad un Indice Sintetico di Costo (tasso annuo effettivo di interesse che comprende tutte le spese accessorie) mediamente del 2,5% fisso.

Il meccanismo del prestito è basato sul concetto dei *punti di risparmio*, punti che vengono accumulati nei periodi di risparmio e consumati nei periodi in cui si accede al prestito. L'idea di base che rende sostenibile l'intero sistema è che i punti di risparmio guadagnati debbano per forza eguagliare i punti di risparmio spesi. Per realizzare questo equilibrio, se all'accensione del mutuo i punti di risparmio consumati sono maggiori di quelli accumulati, bisogna obbligatoriamente continuare a risparmiare durante il periodo di restituzione del prestito, così da continuare ad accumulare punti di risparmio. Alla fine del periodo di restituzione del mutuo, quando i punti di risparmio presi in prestito eguaglieranno quelli guadagnati, si potrà prelevare nuovamente la somma totale dei risparmi "obbligatori" versati, che nel frattempo sono stati utilizzati per erogare prestiti ad altri soci.

LA REALTA' SEMPLIFICATA NEL MODELLO

Il nostro modello si pone l'obiettivo di simulare nell'ambiente NetLogo il sistema di funzionamento della Jak Bank. Il mondo è rappresentato da uno spazio in cui gli agenti interagiscono con la banca che, per rendere più intuitivo il modello, è stata suddivisa idealmente in due entità distinte che occupano due posizioni diverse nello spazio; la prima è la sezione in cui sono raccolti i depositi degli agenti, indicata con "*deposit_bank*", la seconda invece è quella a cui si rivolgono gli agenti per ricevere i prestiti, indicata con "*loan_bank*".

Nel modello gli agenti, che assumono la forma "person" e il colore blu, rappresentano le famiglie. Ogni famiglia presenta una diversa composizione e il numero dei componenti il nucleo famigliare è assegnato casualmente all'avvio del programma. Al tempo zero ogni famiglia riceve una ricchezza iniziale che rappresenta il risparmio accumulato fino a quel momento (*endowment*). La ricchezza iniziale è assegnata casualmente all'interno di un range (da 1000 a 2000 €). Agli istanti successivi in modo analogo è assegnato un salario al netto del consumo, *"netIncome*" (da 100 a 300 €).

Si ipotizza che la banca all'istante zero non abbia ricevuto alcun deposito né abbia erogato prestiti e che inizierà la sua attività di finanziamento dei soci solo se avrà

raggiunto un certo livello di liquidità pari all'importo di un singolo prestito moltiplicato per la stima del numero di famiglie che presumibilmente si indebiteranno nel successivo istante.

Nel modello ogni ciclo corrisponde ad un mese perché l'aggiornamento dei punti di risparmio accumulati da ogni socio è effettuato con cadenza mensile. Inoltre mensilmente analisti professionali e bancari Jak effettuano un'attenta valutazione della situazione finanziaria della Banca in termini di liquidità, in modo tale da individuare la quantità di denaro disponibile per l'erogazione dei prestiti attraverso un meccanismo di calibrazione del rapporto tra il bacino di risparmio e la consistenza dei prestiti erogati. L'importo dei prestiti, nel nostro modello, è fissato a 16000 €, come anche la durata del periodo di ammortamento (11 anni) e la banca può agire solo sul numero di finanziamenti erogati considerando la percentuale stimata di richiedenti un prestito. Nella realtà la Jak Bank stabilisce un massimale erogabile a ciascun cliente basandosi su eventuali garanzie reali offerte dal debitore, sull'affidabilità finanziaria del cliente, sull'esistenza di fiduciari disposti a garantire in caso di insolvenza del socio e anche sulla situazione patrimoniale della banca stessa. Come già spiegato, qui si è deciso di fissare sia l'importo del prestito sia la durata del periodo di ammortamento.

Il modello si sviluppa con il movimento casuale degli agenti che ad ogni ciclo si spostano casualmente in una diversa *patch*. Essi quindi potranno interagire con la banca depositi e la banca prestiti nel momento in cui si troveranno sulle *patches* occupate dall'agente rappresentante una delle due sezioni della banca. Si potranno perciò presentare due situazioni, quella in cui la famiglia si reca nella *deposit_bank* (*yellow patches*) con una probabilità dell'11 % (data dal rapporto tra dimensione *patches* con colore giallo e dimensione dell'intero mondo) e quella in cui si trova nella *loan_bank* (*white patches*) con un probabilità del 3 % (data dal rapporto tra dimensione *patches* con colore bianco e dimensione dell'intero mondo).

Il primo caso implica che:

- ✓ la famiglia effettui un versamento in denaro; in particolare l'agente depositerà il 30 % della totale ricchezza accumulata fino a quel momento, data dalla somma della ricchezza iniziale e degli stipendi percepiti.

- ✓ La regolamentazione prevede che il 20% sia accantonato come riserva dalla banca (*reserve*), che pertanto potrà usare solamente la restante percentuale per erogare prestiti (*liquidity)*. La riserva è una parte dell'attivo che assicura liquidità in caso di insolvenza;
- ✓ effettuando il deposito l'agente accumuli punti di risparmio (*saving_points*).

La relazione tra punti di risparmio e importo depositato è:

$$P_t = P_{t-1} + \left(D_{t-1} * Sf\right)$$

nel caso in cui non sia effettuato un nuovo deposito al tempo t;

$$P_t = P_{t-1} + \left[\left(D_{t-1} + V\right) * Sf\right]$$

nel caso di deposito in t,

dove:

P = punti di risparmio accumulati,

D = ammontare totale dei depositi,

V = singolo versamento effettuato nell'unità di tempo,

Sf = "*saving factor*".

Il *saving factor* è un coefficiente che è stabilito in base alla liquidità della banca e che permette di convertire l'importo in denaro dei depositi in punti di risparmio (ad esempio, dato un *saving factor* pari a 0,8 ed effettuando un versamento iniziale di 1.000 euro, si otterrebbero 800 punti).

Il secondo caso invece riguarda la concessione dei prestiti alle famiglie che potranno usufruire di questo servizio finanziario solo nel caso in cui la banca abbia raggiunto il livello di liquidità predeterminato. Inoltre il sistema prevede che il 6% del prestito

accordato venga pagato dall'agente a titolo di garanzia e immesso nella banca depositi.

Quindi ciò implica che:

- ✓ nel momento in cui una famiglia interagisce con la banca prestiti passando sulle *patches* corrispondente, subisca un incremento della propria liquidità pari all'ammontare del prestito diminuito del 6%;
- ✓ l'agente consumi una quota dei punti precedentemente accumulati in modo tale che al termine dell'ammortamento siano stati persi i punti corrispondenti all'ammontare del prestito. In questo modo si realizza la sostenibilità dell'intero sistema in quanto i punti di risparmio spesi sono eguagliati a quelli guadagnati. I punti residui, rimasti eventualmente inutilizzati, rimangono a disposizione del socio per futuri prestiti;
- ✓ la famiglia debba restituire gradualmente la somma ricevuta mediante rate costanti calcolate semplicemente dividendo l'importo ottenuto per la durata dell'ammortamento. A queste sia aggiunta la tassa sul prestito cioè una percentuale dell'importo totale uguale per tutti i soci e che sia stabilita in base ai costi strutturali sostenuti dalla banca;
- ✓ l'agente assuma colore arancione.

La tipologia fin qui descritta verrà considerata "prestito di base" (*basic_loan*).

Tuttavia non è necessario che la famiglia possegga già al momento dell'erogazione del prestito la quantità di punti di risparmio necessaria ad eguagliare l'importo del prestito, infatti la Jak Bank prevede la possibilità di effettuare un post-risparmio accumulando successivamente la quota di punti mancante. In tal caso il socio, oltre a pagare le rate di restituzione del prestito, dovrà obbligatoriamente depositare la parte di risparmio stabilita. Gli agenti durante il periodo di ammortamento dovranno effettuare versamenti costanti, quindi in questo caso le rate saranno composte dalla quota necessaria per ripagare il prestito di base più il deposito riguardante il post-risparmio. L'agente assume colore rosso.

Per capire se è necessario ricorrere al post-risparmio occorre calcolare i punti di risparmio che corrispondono all'ammontare prestato tramite la seguente formula:

$$\overline{P} = \frac{C * (T + T^2)}{2}$$

dove:

$\overline{P}$ = punti di risparmio necessari,

C = importo del prestito,

T = periodo di ammortamento.

Il risultato deve essere confrontato con i punti già accumulati dal socio. Se la differenza è negativa la famiglia ha i requisiti per richiedere un prestito di base, se invece è positiva dovrà ricorrere al meccanismo del post-risparmio. In quest'ultimo caso la rata necessaria per l'accumulo del post-risparmio sarà data da:

$$R_{post} = \frac{C}{T} - \frac{2 * P}{T + T^2}$$

dove:

R_{post} = rata del post-risparmio,

$\frac{C}{T}$ = rata del prestito base,

P = punti di risparmio accumulati.

Pertanto la rata totale è ottenuta sommando la rata del prestito base e la rata del post-risparmio e la tassa sul prestito. Il sistema prevede che non sia accantonata la percentuale di riserva per i depositi del post-risparmio. Da notare che nel caso in cui la famiglia abbia zero punti al momento della richiesta del prestito, la rata del post-risparmio coincide con la rata del prestito base. Inoltre al termine dell'ammortamento del debito il post-risparmio in denaro accumulato è restituito al socio.

Il modello dovrà prevedere che ciascuna famiglia non abbia la possibilità di chiedere l'erogazione di un prestito nel caso in cui non abbia completato l'ammortamento di un prestito precedente.

Inoltre annualmente le famiglie devono pagare una quota associativa pari a 23 Euro prevista dalla Jak Bank a parziale copertura dei costi di gestione; ciascun componente deve pagare una quota. Nella realtà il capo-famiglia paga la quota più elevata, gli altri membri ne pagano una ridotta e i bambini fino ai 18 anni sono esentati.

Il modello si pone l'obiettivo di verificare la sostenibilità del sistema Jak Bank, pertanto sarà necessario valutare eventuali casi di insolvenza sia da parte delle famiglie sia da parte della banca. Il default delle famiglie si verifica se la loro ricchezza raggiunge il valore zero, mentre la banca va in default nel momento in cui esaurisce la liquidità a disposizione per erogare nuovi prestiti.

In conclusione il modello mostrerà il numero di prestiti erogati, i casi di default verificatisi, la ricchezza totale delle famiglie e la liquidità della banca. Sarà possibile simulare diversi scenari apportando variazioni al numero di famiglie presenti nel sistema, al valore del *saving factor*, al tasso applicato sul prestito, alla stima della percentuale di famiglie richiedenti un prestito e all'intensità dello shock.

DESCRIZIONE DEL CODICE

L'ambiente di sviluppo di NetLogo interpreta direttamente il codice inserito nella sezione "*Procedures* " senza necessita di compilarlo; quindi in tempo reale è possibile interagire con il sistema attraverso pulsanti e sliders per modificare i parametri di controllo e visualizzare variabili e grafici relativi alla simulazione. In particolare l'ambiente è costituito da "*patches*", le quali corrispondono ai pixel del mondo virtuale, sono immobili, possono cambiare colore o contenere informazioni sotto forma di variabili proprietarie; e da "*turtles*", che rappresentano oggetti in grado di muoversi, cambiare colore, forma e proprietà.

Sono stati definiti due diversi tipi (*breeds*) di agenti in modo da separare il comportamento delle famiglie e della banca. La parola chiave *breed* va utilizzata all'inizio del codice, prima di qualsiasi altra procedura, e si struttura come:

breed [<*breeds*> <*breed*>]

Il primo input stabilisce il nome dell'insieme di agenti associati alla *breed* creata, il secondo definisce il nome di ogni singolo membro. Ai fini dello studio eseguito sono state create le due seguenti *breeds*:

breed [families family]
breed [banks bank]

La prima categoria rappresenta l'insieme delle famiglie osservate, cioè i soci della Jak Bank, mentre la seconda riguarda le due sezioni della banca, una dedicata alla raccolta dei depositi effettuati dalle famiglie (*deposit_bank*) e l'altra destinata all'erogazione dei prestiti (*loan_bank*).

Una *variabile* è invece un "luogo virtuale" in cui sono immagazzinati determinati valori (numeri ad esempio). NetLogo mette a disposizione dell'utente tre diverse tipologie di variabili: le *variabili globali* (*globals*), quelle *locali* (*locals*) e quelle possedute esclusivamente da un dato insieme di agenti. E' lecito sia utilizzare delle variabili standard riconosciute automaticamente dall'interprete (come, per esempio, *color*, *shape* e *size*), sia di costruirne di nuove, definendole all'inizio del codice, esattamente come le *breeds*. Le variabili globali esistono in esemplari unici e sono visibili da qualunque punto del codice. Vengono dichiarate fuori da ogni altra procedura e aggiornate col comando "*set nome_var valore*" oppure direttamente dall'interfaccia di sviluppo attraverso gli sliders o gli switch. Si tratta di variabili accessibili a tutti gli agenti. Anche le variabili locali esistono in esemplari unici e sono visibili solo all'interno della procedura dove sono state dichiarate con "*let nome_var valore*". Le variabili proprietarie invece sono assegnate ad una specifica b*reed*

(*turtles-own* ad esempio) o alle patches (*patches-own*) ed avranno perciò validità esclusivamente per ciascun membro di esse.
Nel codice le dichiarazioni di variabili precedono le procedure vere e proprie e servono a dichiarare le variabili globali, le "breeds" e le variabili proprietarie. Il loro corpo é delimitato da parentesi quadre.

Nel modello qui sviluppato sono state definite le seguenti tre *globals*:

```
globals
        [
        deposit_account
        liquidity                ; liquidity is the amount available to grant loans
        reserve]
```

La prima variabile rappresenta la somma di tutti i depositi effettuati da parte delle famiglie; la seconda è l'ammontare disponibile per la banca nella sua attività di concessione prestiti; la terza infine rappresenta la quota dei depositi accantonata a riserva.
Sono inoltre dichiarate le seguenti variabili proprietarie, valide soltanto per la *breed* "families"[1]:

```
families-own
        [
        endowment
        family_members
        netIncome                ; monthly income after consumption
        saving_points
        personal_deposit         ; amount deposited in the deposit_bank by each family
        repayments               ; amount of the payed repayments
        borrower?                ; true if the family has got a loan
        shocked?                 ; true if the family has suffered an economic shock
        dead?                    ; true if the family wasn't able to repay the loan and so it has left the bank
        ]
```

[1] Le parti di codice precedute da ";" sono i commenti inseriti per facilitare la comprensione del codice stesso.

La variabile *endowment* indica il risparmio accumulato da ciascun agente; *family_*members si riferisce al numero di componenti il nucleo famigliare; netIncome è invece il salario al netto del consumo ricevuto mensilmente da una famiglia; *saving points* sono i punti di risparmio accumulati dal socio presso la Jak Bank; *personal_deposit* rappresenta l'ammontare depositato da ciascuna famiglia presso la banca; *repayments* è una variabile che presenta valori diversi da zero solamente nel periodo di ammortamento di un prestito e che indica i ripaga menti già effettuati; *borrower?* è una variabile booleana che assume valore vero (*true*) nel caso in cui l'agente si trovi in fase di restituzione del finanziamento ricevuto; *shocked?* assume valore vero se l'agente è stato colpito dallo shock economico in atto e, infine, *dead?* assume valore vero se l'agente non è stato in grado di ripagare il debito ed ha perciò dovuto abbandonare la banca.

In NetLogo tutto il codice del modello è suddiviso in procedure che possono essere destinate all'esecuzione da parte del modello, delle patches o delle turtles. Le procedure possono essere di due tipi diversi: le cosiddette "*subroutines*" sono richiamate da altre "*subroutines*" oppure direttamente dall'interfaccia grafica per mezzo della pressione di un pulsante. Possono prendere valori in ingresso ma non restituiscono alcun valore in uscita. Iniziano sempre con la parola chiave "*to*" e finiscono con "*end*". I "*reporters*" invece sono chiamati da altri "*reporters"* o "*subroutines"*.

Possono prendere valori in ingresso e restituiscono sempre un valore in uscita, attraverso il comando "report". Iniziano sempre con la parola chiave "*to-report*" e finiscono con "*end*". Nel nostro modello è stata utilizzata solamente la prima tipologia di procedura.

In NetLogo è detta "procedura" una combinazione di comandi pre-esistenti volta a costituire a sua volta un nuovo comando utilizzabile dagli agenti. La prima procedura definita è quella di *setup*:

```
to setup
        clear-all
        setup-patches
```

```
        setup-turtles
        do-plots
end
```

Setup pone in essere lo stato iniziale del modello ed è richiamato dall'omonimo bottone presente nell'interfaccia grafica. La procedura inizia con il comando *clear-all*, che annulla tutti i processi in precedenza avviati e azzera i valori raggiunti dalle variabili chiamate. Successivamente, all'interno di *setup*, vengono richiamate alcune procedure necessarie per la definizione dello stato iniziale del modello: *setup-patches*, *setup-turtles*, le quali richiamano le successive procedure che definiscono le caratteristiche di default iniziali di ciascuna categoria. La procedura *setup* si conclude con il richiamo alla procedura *do-plots*, che produce la realizzazione della rappresentazione grafica e per la cui spiegazione nel dettaglio si rimanda alle pagine successive.

La procedura *setup-patches*, assegnando i valori desiderati alla variabile *pcolor* (accessibile per altro alle sole *patches*), consente di visualizzare graficamente la presenza delle due sezioni della Jak Bank all'interno dello spazio:

```
to   setup-patches
        ask patches [
            set pcolor green
            if ((pxcor > 9) and (pxcor < 15) and (pycor > 9) and (pycor < 15))
                [ set pcolor yellow ]  ; the yellow square represents the deposit section of the bank
            if ((pxcor > -13) and (pxcor < -2) and (pycor > -13) and (pycor < -2))
                [set pcolor white]    ; the white square represents the loan section of the bank
]
end
```

Si noti l'importanza del comando *ask* in NetLogo: viene utilizzato per impartire agli agenti, o ad un loro insieme, le istruzioni da compiersi, ed è pertanto la premessa fondamentale per l'assegnazione della maggior parte dei comandi. Da evidenziare anche la presenza di *if*, che, controllando per una data condizione (in questo caso le coordinate x e y delle *patches*), modifica il colore della *patch* in giallo o bianco se la

condizione è verificata o lo mantiene verde in caso contrario, seguendo il seguente schema generale:

if *condition* [*commands*]

Il comando *set*, invece, serve per assegnare un dato valore alla variabile considerata:

set *variable value*

La seconda procedura stabilisce invece la creazione delle turtles e le caratteristiche a loro assegnate allo stato iniziale del modello:

```
to   setup-turtles
  create-families number_of_families
  ask families [
      setxy random-pxcor random-pycor
      set shape "person"
      set size 1.5
      set color blue
      set endowment (1000 + random 1001 )  ; each family starts with a different initial endowment
      set family_members (1 + random 5)          ; we define a random number of family's members
      set netIncome (100 + random 201)
      set borrower? false
      set shocked? false
      set dead? false]
  create-banks 1                                              ; this is the deposit section of the bank
  ask bank (number_of_families) [
      setxy 12 12
      set shape "house"
      set color brown
      set size 2]
  create-banks 1                                              ; this is the loan section of the bank
  ask bank (number_of_families + 1) [
       setxy -7.5 -7.5
       set shape "house"
       set color brown
       set size 3]
```

```
end
```

Le *turtles* vengono create con un proprio numero identificativo (*who number*) corrispondente all'ordine con cui vengono inizializzate e in posizione casuale all'interno dello spazio.

All'avvio del programma, la procedura prevede che ad ogni turtles appartenente alla categoria *families* siano attribuite le seguenti caratteristiche: forma "*person*", colore blu, ricchezza iniziale assegnata casualmente all'interno di un range (da 1000 a 2000 euro[2]), numero casuale di componenti il nucleo familiare (da 1 a 5) e reddito mensile al netto del consumo assegnato casualmente all'interno di un range (da 100 a 300 euro).

Ai due agenti rappresentanti le due sezioni della banca sono invece attribuite la forma "*house*", il colore marrone ed le coordinate della precisa posizione occupata all'interno dello spazio.

Una volta inizializzato il modello, interviene la procedura *go* che rende dinamico il mondo appena creato. Essa è associata ai bottoni *step* e *go*, i quali differiscono tra loro semplicemente per il numero di volte con cui viene ripetuta la procedura. A questo punto risulta necessario spiegare che l'unità di misura del tempo adottata in NetLogo sono i *ticks*. La loro utilità emerge soprattutto nel momento in cui si vogliono confrontare diversi esperimenti, sia all'interno dello stesso modello, sia fra modelli simili. Nel nostro caso specifico si è scelto di considerare un tick equivalente ad un mese.

```
to go
    if (liquidity < reserve) [stop]                ; condition of bank's default that stops the program
        move-turtles
        make-deposit
        pay-membership-fee
        get-saving-points
        get-basicLoan
```

[2] I dati utilizzati fanno riferimento ai reali valori medi di reddito e consumo in Svezia disponibili sul sito www.worldsalaries.org/sweden.shtml.

```
        repay-basicLoan
        stop-repay-basicloan
        get-postSavingLoan
        repay-postSavingLoan
        stop-repay-postSavingLoan
        default_families
        default_bank
        do-plots
end
```

Tramite il comando *if* si definisce la condizione necessaria per l'arresto del modello: la variabile *liquidity*, che indica l'ammontare a disposizione della banca per l'erogazione dei prestiti, è minore della variabile *reserve*, che indica invece la porzione di depositi accantonata a riserva.

Il richiamo della procedura *move-turtles* consente di fatto il funzionamento dell'intero modello nella sua fase di realizzazione. Questa porzione di codice prevede infatti che ad ogni *tick* ciascun agente appartenente alla categoria *families* cambi posizione. Si è scelto di rendere lo spostamento delle *turtles* completamente casuale per fare in modo che ciascuna *patch* abbia la stessa probabilità di essere occupata. Si noti che gli agenti appartenenti alla seconda categoria, cioè la banca depositi e la banca prestiti, mantengono sempre la stessa posizione e non si muovono in seguito all'avvio del programma. L'ultima riga della procedura stabilisce inoltre che l'*endowment* di ciascuna famiglia sia incrementato mensilmente, perciò ogni *tick*, di un ammontare pari al *netIncome*.

```
to move-turtles
    ask families [
      right random
      setxy random-pxcor random-pycor
      set endowment (endowment + netIncome)]      ; every month the endowment is increased by
the net income
end
```

Conclusa la parte iniziale del codice in cui vengono fissate le impostazioni di avvio del programma, si sviluppa la parte centrale in cui sono descritte le procedure di raccolta dei depositi effettuati dai soci ed erogazione dei finanziamenti da parte della banca. In particolare quest'ultima porzione di codice è stata suddivisa in due parti: quella riferita ai prestiti base e quella relativa ai prestiti con post-risparmio.

Per quanto riguarda i depositi si richiama la procedura *make-deposit* (si noti che tale procedura è presente all'interno del comando *go* e viene perciò attivata ogni qualvolta siano premuti i pulsanti *step* oppure *go*).

```
to   make-deposit
        ask families[
            if (pcolor = yellow and borrower? = false and endowment > 0
              and shocked? = false and dead? = false)
           [let n [endowment * 30 / 100]
            of one-of families-on patches with [pcolor = yellow]      ; families deposit the 30% of their
endowment
           set endowment (endowment - n)
           set personal_deposit (personal_deposit + n)
                 ask bank (number_of_families)[
                       set deposit_account (deposit_account + n)
                       set reserve (reserve + (n * (20 / 100)))   ; the 20% of deposits is accounted
as reserve
                       set liquidity (liquidity + (n * (80 / 100))))]
      ] ]
end
```

Il codice sopra riportato produce effetti su entrambe le categorie di agenti, infatti è utilizzato due volte il commando *ask*. Tramite il comando *if* si definisce la serie di condizioni necessarie affinchè l'agente effettui il deposito: l'agente deve trovarsi su una delle *patches* di colore giallo, deve possedere un *endowment* positivo, deve aver completato il periodo di ammortamento di prestiti eventualmente ottenuti in precedenza, non deve essere stato colpito dallo shock e non deve aver abbandonato la banca. Si noti la definizione della variabile locale "n" attraverso il comando *let*. Tale variabile individua l'ammontare depositato da ciascun agente in quell'istante ed è pari al 30 per cento dell'*endowment* accumulato. Il valore della variabile locale sarà

pertanto diverso per ciascun agente in quanto diversi sono la ricchezza iniziale e il reddito netto assegnati.
Formalmente, grazie all'impiego del comando *with*, è stato possibile creare un nuovo *agentset* contenente solo quegli agenti che si trovano sulle *patches* gialle e che soddisfano perciò il reporter indicato all'interno delle parentesi quadre. Più in generale *with* richiede due input:

***agentset* with [*reporter*]**

Sulla sinistra c'è l'insieme di agenti a cui viene impartito l'ordine; sulla destra un *reporter* booleano. Il risultato è un nuovo *agentset* contenente solo quegli agenti che soddisfano le condizioni espresse dal *reporter*.
Vengono inoltre utilizzati i comandi *of* e *one-of* i quali seguono i seguenti schemi generali:

[*reporter*] of *agentset*

one-of *agentset*

Il primo implica che per l'*agentset* indicato sulla destra, venga riportata una lista contenente il valore del *reporter* per ciascun agente compreso in quel determinato *agentset* (seguendo un ordine casuale). Il secondo invece stabilisce che venga considerato un agente casuale all'interno dell'*agentset* specificato (se l'*agentset* è vuoto non viene considerato nessun agente).
Una volta effettuato il deposito l'agente vedrà il proprio *endowment* decurtato di n e il *personal_deposit* incrementato dello stesso valore. Anche le variabili proprietarie della banca subiranno una variazione: il *deposit_account* si incrementa di n, la *reserve* del 20 per cento di n e la *liquidity* dell'80 per cento di n.
Nel momento in cui un socio effettua un deposito presso la Jak Bank è previsto che gli siano accreditati un certo numero di punti di risparmio che continuano a generare ulteriori punti per tutto il periodo in cui il denaro rimane presso la banca. Pertanto il codice prevederà che, solo nel caso in cui l'agente sia di colore blu (non sia in fase di ripagamento di un prestito) o arancione (sia in fase di ripagamento di un prestito base), la variabile *saving_points* sia incrementata del *personal_deposit* moltiplicato

per il *saving_factor* (fattore utilizzato proprio per convertire i depositi in euro in equivalenti punti di risparmio, modificabile attraverso lo slider presente nell'interfaccia). La procedura risulta perciò essere la seguente:

```
to get-saving-points                          ; every month the member accumulates saving points
      ask families[
        if (borrower? = false and (color = blue or color = orange))[
        set saving_points (saving_points + personal_deposit * saving_factor)
 ]  ]
end
```

La sezione successiva del codice si occupa dell'ottenimento da parte dei soci di un prestito base. Come già spiegato in precedenza questo è il caso in cui il socio abbia già accumulato, grazie a depositi effettuati precedentemente, il numero sufficiente di punti di risparmio al fine di ottenere il prestito di medio importo.

```
to  get-basicLoan
          let z 16000 * ((132 / 12) + (132 / 12) ^ 2) / 2                    ; necessary points for a loan
          let p 16000 * ((number_of_families * estimated_%_of_borrowers))
         ask families[
             if (pcolor = white and borrower? = false and z <= saving_points and liquidity > p
               and shocked? = false and dead? = false)   ; in this case the family doesn't need to
make post-saving
                    [set borrower? true
                     set color orange
                     set saving_points (saving_points - z)
                    set endowment (endowment + 16000)
                              ask bank (number_of_families)[
                                        set liquidity (liquidity - 16000)]
          repay-basicLoan]
   ]
end
```

Il codice definisce due variabili locali, "z" e "p" che si riferiscono rispettivamente al numero di punti necessario per l'ottenimento di un prestito di 16.000 euro

restituibile nell'arco di 11 anni[3] e all'importo totale di prestiti da erogare nel periodo successivo. Tale valore, che in genere viene stimato dagli operatori della banca, nel modello può essere modificato dall'interfaccia del programma attraverso uno slider denominato "*estimated_%_of_borrowers*" (il valore può variare da 0 a 1 con incrementi multipli di 0.1). Ancora tramite il comando *if* è esplicitato l'insieme di condizioni che devono essere contemporaneamente soddisfatte affinchè possano essere applicate le variazioni successivamente indicate in parentesi quadre. In particolare, l'agente potrà usufruire di un finanziamento di tipo base se la *patch* su cui si trova è di colore bianco (come si è già detto in precedenza si è scelto di concretizzare l'interazione tra la banca e il socio al passaggio di quest'ultimo su una delle *patches* occupate dall'istituto finanziario), se non ha altre posizioni aperte con la banca, se ha accumulato un numero di punti di risparmio pari ad almeno il valore di z, se la liquidità della banca è superiore a p, se non ha subito shock e se non ha lasciato la banca.

In tal caso l'agente vedrà la sua variabile booleana *borrower?* assumere il valore vero, diventerà di colore arancione, consumerà z punti di risparmio e incrementerà la propria ricchezza di 16.000 euro. La liquidità della banca subirà invece un decremento di 16.000 euro. Viene a questo punto, richiamata la procedura *repay-basicLoan* che presenta il seguente schema:

```
to   repay-basicLoan
        ask families[
            if (borrower? = true and color = orange)[
              set endowment (endowment - ((16000 / 132) + ((loan_fee / 12) * 16000)))
              set repayments (repayments + ((16000 / 132) + ((loan_fee / 12) * 16000)))
                  ask bank (number_of_families)[
                        set liquidity (liquidity + ((16000 / 132) + ((loan_fee / 12) * 16000)))]
       stop-repay-basicLoan]
   ]
end
```

[3] Le caratteristiche del prestito rispecchiano le condizioni medie a cui vengono erogati i prestiti presso la Jak Bank operante in Svezia. Tali informazioni sono state reperite sul sito ufficiale della banca cooperativa (www.jak.se).

Tale procedura regola il pagamento mensile delle rate di restituzione del prestito con conseguenti diminuzione dell'*endowment* e aumento dei *repayments*. Si noti che il comando *if* subordina l'avvio di questo piano d'ammortamento al solo caso in cui l'agente abbia assunto colore arancione ed abbia ottenuto un prestito.

La loan fee che compare nella formula per il calcolo della rata rappresenta il tasso d'interesse applicato sul prestito. Tale dato è stabilito periodicamente dalla banca, è pertanto fisso e uguale per tutti i soci. Nel modello può essere variato attraverso uno slider.

Il pagamento delle rate viene fermato in due casi: quando la variabile *repayments* raggiunge o supera l'ammontare totale del debito (dato dalla somma del capitale e degli interessi e definito con "c" nel modello) oppure quando la ricchezza della famiglia si esaurisce. Ciò provoca l'azzeramento dei *repayments*, il ritorno al colore blu e lo stop della procedura *repay-basicLoan*.

```
to   stop-repay-basicLoan
        let c 16000 +  ((loan_fee / 12) * 16000) * 132          ; total amount of the debt: capital +
interests
        ask families[
            if ((repayments >= c or endowment <= 0.001)
                and borrower? = true and color = orange)[      ; condition of amortization end
        set borrower? false
        set color blue
        set repayments 0
        stop (repay-basicLoan)]
 ]
end
```

La parte successiva del codice riguarda invece l'ottenimento di un prestito con post risparmio. La parte iniziale della procedura è molto simile a quella riguardante la tipologia di prestito base; le uniche differenze consistono nel fatto che in questo caso l'agente assume colore rosso, anziché arancione e nel fatto che il comando *if* prevede come condizione di validità che la famiglia non abbia ancora accumulato il numero sufficiente di punti di risparmio, anche qui definito con la variabile locale "z".

```
to  get-postSavingLoan
        let z 16000 * ((132 / 12) + (132 / 12) ^ 2) / 2                ; necessary points for a loan
        let p 16000 * ((number_of_families * estimated_%_of_borrowers))
        ask families[
            if (pcolor = white and borrower? = false and z > saving_points and liquidity > p and
            shocked? = false and dead? = false)[      ; in this case the family must make post-saving
                set borrower? true
                set color red
                set endowment (endowment + 16000)
                ask bank (number_of_families)[
                        set liquidity (liquidity - 16000)]
        repay-postSavingLoan]
    ]
end
```

Il passo successivo consiste nel definire il piano di ammortamento di un prestito con post-risparmio. La rata da pagare mensilmente risulta essere composta da tre elementi: la prima parte costituisce la quota capitale (ottenuta semplicemente dividendo l'importo ricevuto per la durata dell'ammortamento in mesi), la seconda è la quota interessi e la parte finale è la quota aggiuntiva di risparmio necessaria per l'accumulo dei punti di risparmio mancanti. Nella procedura l'ammontare totale della rata mensile è indicato con la variabile locale "s".

```
to repay-postSavingLoan
      ask families[
        let s ((((16000 / 132) + ((loan_fee / 12) * 16000))) + ((16000 / 132) –
            ((2 * saving_points / (132 + 132 ^ 2)))))
        if (borrower? = true and color = red)[
                set endowment (endowment - s)
                set repayments (repayments + ((16000 / 132) + ((loan_fee / 12) * 16000)))
                    ask bank (number_of_families)[
                        set liquidity (liquidity + s)]]
    stop-repay-postSavingLoan]
end
```

Come nella situazione precedente, relativa al prestito base, quando il piano di ammortamento viene completato si ferma la procedura *repay-postSavingLoan.* Qui la banca provvede a restituire al cliente le quote di post-risparmio depositate durante il periodo di estinzione del debito; pertanto l'*endowment* delle famiglie subirà un aumento pari a t e la *liquidity* della banca un decremento di pari valore.

```
to   stop-repay-postSavingLoan
      let c (16000 +  ((loan_fee / 12) * 16000) * 132)     ; total amount of the debt: capital + interests
      ask families[
      let t ((16000 / 132) - ((2 * saving_points / (132 + 132 ^ 2))))   ; amount of post-saving deposit
      if ((repayments >= c or endowment <= 0.001)
          and borrower? = true and color = red)[        ; condition of amortization end
          set borrower? false
          set color blue
          set repayments 0
          set endowment (endowment + (132 * t))
          set liquidity (liquidity - (132 * t))
          set saving_points 0
      stop (repay-postSavingLoan)]
 ]
end
```

Il sistema Jak per garantire la sua sostenibilità richiede ai soci il pagamento di una quota associativa annua pari a 23 euro. Tale commissione viene utilizzata come parziale copertura dei costi di gestione e deve essere versata da ciascun componente del nucleo familiare. Si noti nella procedura l'utilizzo del comando *mod* che segue il seguente schema:

number1 mod number2

Il commando deve perciò essere compreso tra due inputs e riporta il resto ottenuto facendo il rapporto tra l'input di sinistra e quello di destra. In tal modo si stabilisce che ogni 12 *ticks*, perciò ogni anno, tale importo venga detratto dall'ammontare di ricchezza accumulata da ciascuna famiglia. Non è stato previsto che la quota incrementi la liquidità della banca poiché, come spiegato prima, si suppone che tale

disponibilità liquida venga utilizzata a copertura di spese strutturali e non abbia pertanto rilevanza all'interno del nostro modello.

```
to pay-membership-fee
        ask families[
        if (ticks mod 12 = 0 and shocked? = false)[
           set endowment (endowment - 23 * family_members)     ; at the beginning of every year
each family's member pays a membership fee
  ]
  ]
        tick
end
```

Nella porzione finale del codice si determina quando le famiglie sono considerate insolventi: ogni qualvolta la loro ricchezza si annulla gli agenti assumono la forma"x" e la variabile booleana *dead?* assume valore *true*. Ciò indica che la famiglia ha abbandonato la banca e non potrà più ricevere altri prestiti, effettuare nuovi depositi o subire gli effetti di un eventuale shock.

```
to default_families
          ask families with [endowment <= 0.001][
                    set shape "x"
                    set dead? true
                    set borrower? false
                    set shocked? false
]
end
```

Le successive procedure sono associate ai due bottoni presenti nell'interfaccia del programma *shock* e *stop-shock*. Con tali comandi è possibile rispettivamente attivare e disattivare la presenza di uno shock economico nel mondo rappresentato dal modello. Tale shock provoca l'annullamento del reddito netto mensile, una riduzione dell'*endowment* pari all'80 per cento, l'azzeramento del *personal_deposit* e l'assunzione da parte dell'agente colpito della forma "*ghost*". E' possibile, modificando lo *slider* denominato *shock_intensity*, variare l'intensità dello shock da

un minimo di 0.1 ad un massimo di 1.0. Tale valore indica la percentuale di famiglie che subiscono l'impatto provocato dalla crisi. Da notare che il comando *n-of* permette di creare un nuovo *agentset* di dimensione pari a quella indicata nella posizione *size* casualmente scelto all'interno dell'input indicato, senza ripetizioni. Lo schema seguito è il seguente:

n-of *size agentset*

Il pulsante *stop-shock*, invece, fa sì che l'agente riassuma la forma "*person*" e ritorni a ricevere un reddito netto mensile. Si è scelto di fornire alla famiglia appena uscita dalla situazione di shock una quota di risparmio più limitata (assegnata casualmente tra 50 e 100 euro), pensando che presumibilmente le famiglie colpite dalla crisi si troveranno in condizioni più precarie anche nella fase di ripresa dell'economia.

```
to shock
        ask n-of (number_of_families * shock_intensity) families[
                set shocked?  true
                set netIncome 0
                set endowment (endowment * 0.2)
                set personal_deposit 0
                set shape "ghost"]
end

to stop-shock
        ask families[
        if (shocked? = true)[
          set shocked? false
          set netIncome (50 + random 51)
          set shape "person"
          set color blue]
]
end
```

La procedura *do-plots* sviluppa un grafico: "*Bank situation*". Ogni rappresentazione grafica delle variabili richiamate è attuata mediante i comandi *set-current-plot-pen*,

che crea la "penna" con cui tracciare il grafico, e *plot*, il quale disegna l'andamento della variabile. E'possibile, cliccando con il tasto destro del mouse sul grafico nell'interfaccia, accedere, attraverso "*edit*", ad una finestra di dialogo con cui personalizzare ciò che è stato appena creato (assegnando, per esempio, i valori massimi e i valori minimi all'asse delle ascisse e a quello delle ordinate, oppure cambiando colore alle "penne" utilizzate e così via).

```
to do-plots
        set-current-plot "Bank situation"
        set-current-plot-pen "deposit_account"
        plot deposit_account
        set-current-plot-pen "reserve"
        plot reserve
        set-current-plot-pen "liquidity"
        plot liquidity
end
```

Infine, si evidenzia la possibilità di osservare nell'interfaccia il numero di *post-saving families*, basic loan families, families without a loan, shocked families e default families (grazie agli appositi *monitor*), oltre che l'andamento delle variabili proprie di ogni *turtle* (cliccando con il tasto destro sulla *turtle* di interesse).

PIANO DEGLI ESPERIMENTI

Per effettuare i nostri esperimenti abbiamo analizzato casi estremi in modo da mostrare chiaramente l'effetto delle variazioni dei parametri. Inoltre abbiamo considerato un orizzonte temporale di 80 anni, cioè 960 cicli del programma.

ESPERIMENTO N. 1:

Ipotesi:

Saving factor = 0.7 contro il valore di partenza posto pari a 1

Tesi:

Un *saving factor* più basso comporta un accumulo di punti più lento da parte delle famiglie; di conseguenza si verifica un maggior ricorso al *post-saving loan* e un minor numero di *basic loans*.

La banca quindi potrebbe aver la necessità di abbassare il *saving factor* in caso di scarsa liquidità, in modo da incrementare le entrate derivanti dai *post-saving loans*.

Inoltre un maggior utilizzo del sistema del post-risparmio determina una maggior esposizione delle famiglie al default.

Esecuzione esperimento:

Dopo aver modificato lo slider *saving_factor* passando dal valore iniziale 1 al valore 0.7, si clicchi il bottone *setup* per inizializzare il mondo e poi il pulsante *go* per eseguire l'esperimento.

Per rendere più significativi i risultati abbiamo reiterato la procedura dieci volte e calcolato la media delle diverse realizzazioni ottenute.

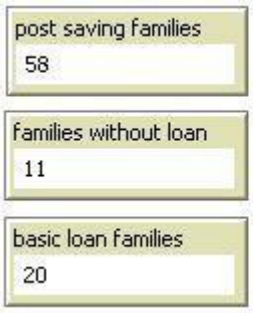

Fig. 1.1: *SF* = 1

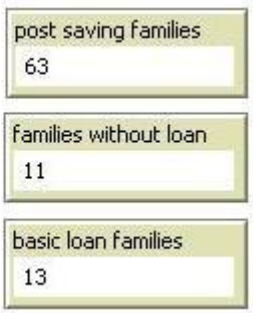

Fig. 1.2: *SF* = 0.7

Come mostrano le figure 1.1 e 1.2, nel caso di *saving factor* pari a 0.7 le famiglie che han fatto ricorso al *post-saving loan* sono maggiori rispetto al caso di SF pari a 1; al contrario scende il numero di famiglie che utilizzano un *basic loan*.

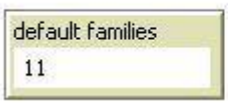

Fig. 1.3: SF = 1

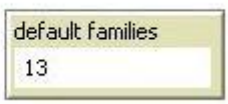

Fig. 1.4: SF = 0.7

Inoltre come si nota dalla figura 1.3, con un *saving factor* uguale a 0.7 aumenta il numero di soci che non riesce a ripagare le rate.

Conclusioni:

L'esperimento eseguito ha confermato le nostre aspettative, quindi è valida la tesi secondo cui se la banca abbassa il *saving factor* il numero di famiglie che fa ricorso al post-risparmio aumenta, a fronte di un lieve incremento del numero delle famiglie in default.

ESPERIMENTO N. 2

Ipotesi:

a) loan fee uguale a 0.020 (2%) contro il valore di partenza posto pari a 0.013

b) loan fee = 0.020 e saving factor = 0.7

Tesi:

a) una loan fee più alta comporta una maggiorazione nell' importo delle rate e quindi un aumento della liquidità disponibile per la banca. Per contro le famiglie sono più esposte al rischio di non riuscire a ripagare i loro debiti, dunque il numero di soci in default aumenta.

b) l'effetto combinato di un aumento della loan fee e di una contemporanea riduzione del saving factor causa un ulteriore incremento della liquidità della banca, a fronte di un maggior numero di soci in default rispetto al punto a).

Esecuzione esperimento:

a) Dopo aver modificato lo slider *loan fee* passando dal valore iniziale pari a 0.013 al valore 0.020, si clicchi il bottone *setup* per inizializzare il mondo e poi il pulsante *go* per eseguire l'esperimento.

Anche in questo caso per rendere più significativi i risultati abbiamo ripetuto la procedura dieci volte e calcolato la media delle diverse realizzazioni ottenute.

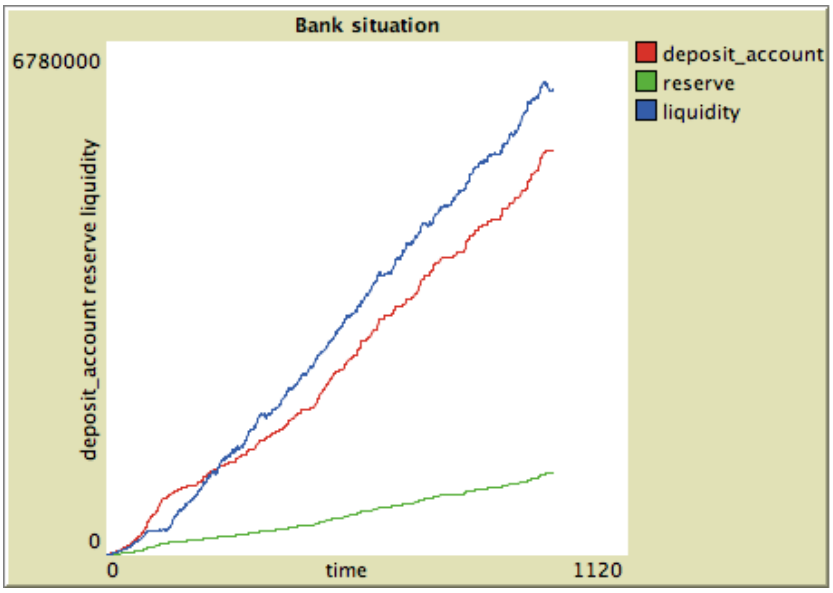

Fig. 2.1: loan fee = 0.013

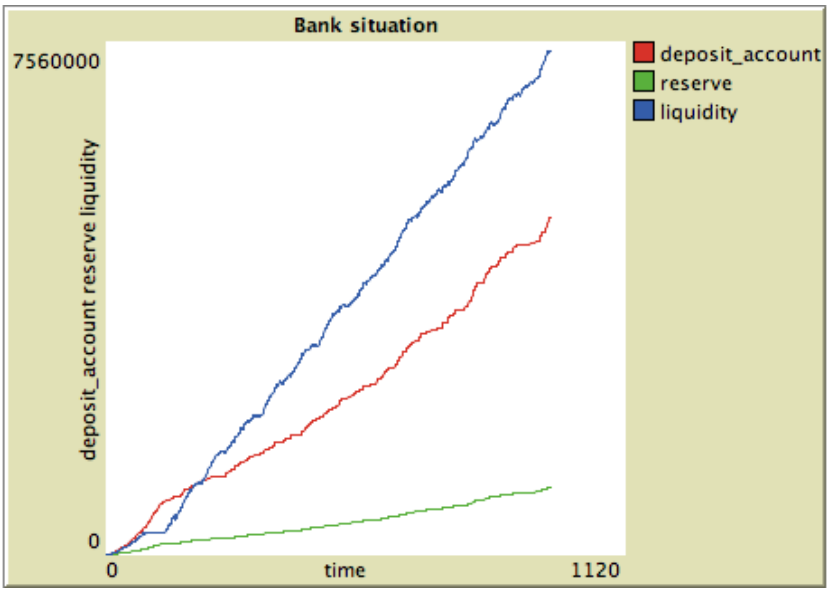

Fig. 2.2: loan fee = 0.020

Fig. 2.3: loan fee = 0.013

Fig. 2.4: loan fee = 0.020

La liquidità della banca aumenta, passando da un valore medio di 6.700.000 a 7.600.000. Le figure 2.1 e 2.2, che rappresentano i risultati della prova n.° 4, confermano i dati medi.

I monitor riguardanti il default delle famiglie mostrano un incremento del 3% (con un campione di 100 famiglie) nel numero di soci che non riesce a saldare il proprio debito.

b) Si modifichi anche lo slider *saving factor* e si avvii la procedura.

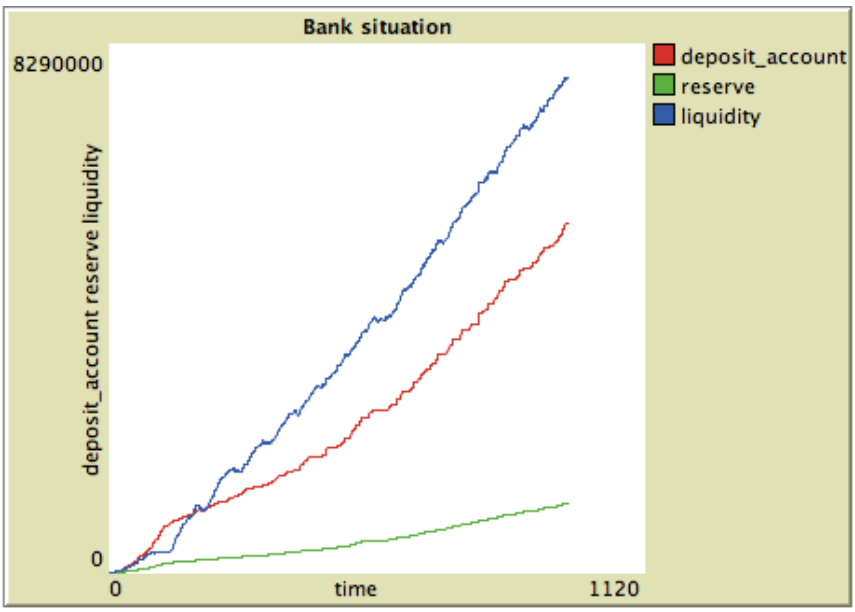

Fig. 2.5: loan fee = 0.020, saving factor = 0.7

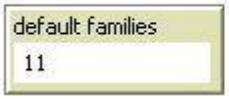

Fig. 2.6: loan fee = 0.013
SF = 1

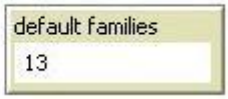

Fig. 2.7: loan fee = 0.020
SF = 0.7

La liquidità della banca passa dal valore medio del punto a) 7.600.000 al nuovo valore medio di 8.350.000; il grafico 2.5 mostra i risultati della prova n.°4.

Il numero di default diminuisce rispetto al punto a) con una variazione dell' 1%.

<u>Conclusioni:</u>

a) I risultati dell'esperimento confermano la nostra tesi. La banca potrebbe quindi alzare il tasso sui prestiti a causa di aumenti nei costi di gestione e struttura con lo scopo di incrementare la propria liquidità. Come previsto

però il numero di soci in default aumenta a causa di rate più alte; questa conseguenza potrebbe avere ripercussioni sulla situazione della banca che quindi si troverebbe di fronte a un trade-off.

b) Gli esiti ottenuti confermano in parte la tesi di partenza. Come previsto l'effetto combinato di *saving factor* e *loan fee* determina un incremento della liquidità della banca, mentre si registra un' inattesa diminuzione del numero di default. Quest'ultimo risultato potrebbe essere dovuto ad un numero troppo basso di prove realizzate.

In ogni caso sembrerebbe conveniente per la banca non modificare singolarmente i due parametri considerati, ma farli variare contemporaneamente.

ESPERIMENTO N. 3

In questa sezione analizziamo l'impatto di uno shock economico sulla situazione della banca.

Ipotesi:

a) - *shock intensity* = 1 rispetto ad un valore iniziale uguale a 0.4
- numero di famiglie: 100
- lo shock si verifica dopo 10 anni (120 cicli)

b) - *shock intensity* = 1 rispetto ad un valore iniziale uguale a 0.4
- numero di famiglie: 100
- trascorsi 10 anni (120 cicli) lo shock si verifica e termina ad intervalli regolari

c) - *shock intensity* = 1 rispetto ad un valore iniziale uguale a 0.4
- numero di famiglie: 20
- lo shock si verifica dopo 10 anni (120 cicli)

Tesi:

a) con una situazione così estrema in cui la totalità delle famiglie è colpita da uno shock, la banca si trova in forte difficoltà e la liquidità potrebbe scendere al di sotto del livello della riserva.

b) la banca non è in grado di sopportare una sequenza di shock perché non riesce ad acquisire sufficiente liquidità tra uno shock e l'altro.

c) diminuendo il numero di soci è più improbabile che la banca riesca ad affrontare le conseguenze di uno shock che colpisce tutte le famiglie, quindi in molti casi la liquidità è minore della riserva. Ciò è dovuto al fatto che ci siano meno soci che depositano e il livello della liquidità della banca pre-shock è inferiore a quello del punto a).

Esecuzione esperimento:

a) Dopo aver modificato il valore dello slider *shock_intensity*, si inizializzi il mondo e si clicchi su *go.* Dopo 120 cicli si clicchi il pulsante *shock.*

Effettuando 10 prove si rileva che nel 20% dei casi la liquidità della banca risulta inferiore alla riserva.

Osservazioni: analizzando il grafico 3.1 si può notare come a seguito di uno shock la liquidità della banca non scenda subito, ma la diminuzione inizi solo trascorsi circa 15 mesi, raggiungendo un livello stabile dopo circa 3 anni.

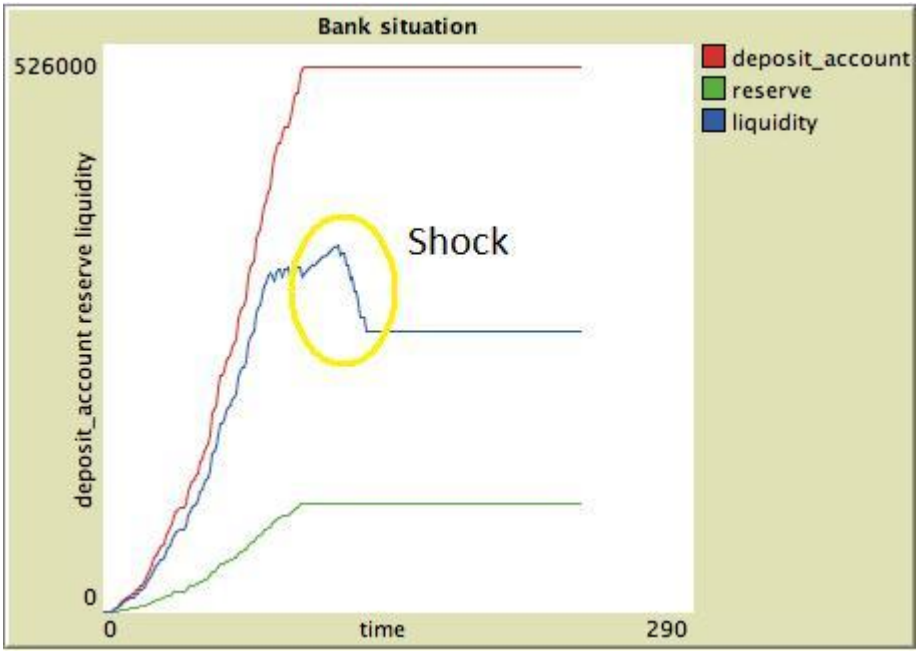

Fig. 3.1 : effetto di uno shock economico

b) Dopo aver modificato il valore dello slider *shock_intensity*, si inizializzi il mondo e si clicchi su *go.* Per generare ripetuti shock si clicchi sul pulsante *shock* e immediatamente dopo su *stop-shock* e si ripeta la procedura per più volte consecutive.

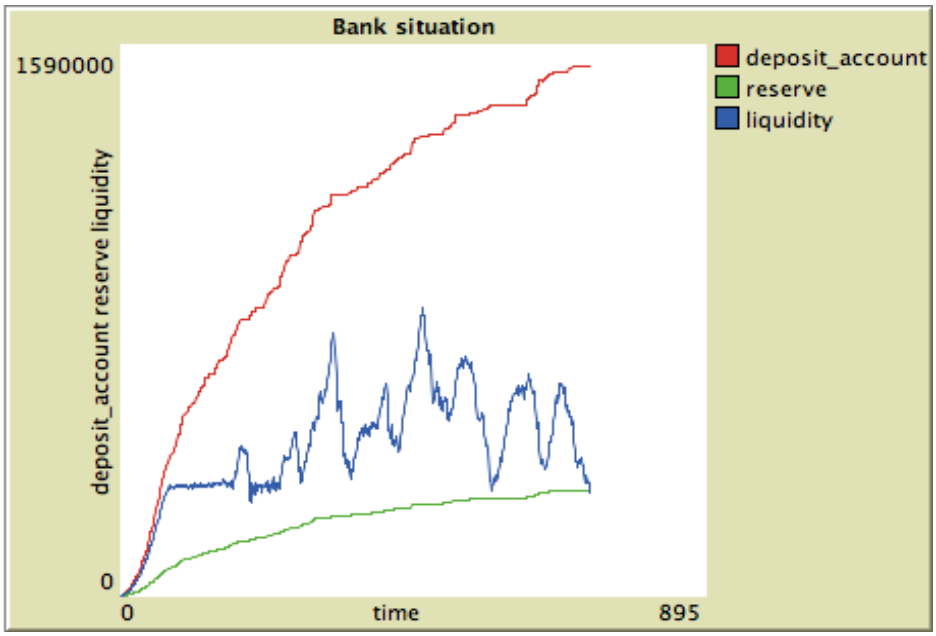

Fig. 3.2: effetto di shock consecutivi

Come mostra la figura, dopo ripetuti shock la liquidità della banca risulta inferiore alla riserva.

c) Dopo aver modificato gli slider *shock_intensity* e *number_of_families,* si clicchi il bottone *setup* per inizializzare il mondo e poi il pulsante *go* per eseguire l'esperimento.

Effettuando 10 prove si nota che in tutti i casi la liquidità della banca scende al di sotto del livello della riserva.

<u>Conclusioni:</u>

a) La nostra tesi non è verificata in quanto ci aspettavamo un numero maggiore di situazioni in cui la liquidità fosse al di sotto della riserva. Invece la banca dimostra di riuscire a sopportare eventi avversi nella maggior parte dei casi (80% circa).

b) Quanto da noi atteso è stato confermato dai risultati ottenuti, infatti la banca non riesce a sostenere più shock consecutivi. Comunque si può notare come siano necessari molti shock prima che la banca si trovi in difficoltà.

c) Le nostre aspettative sono confermate dall'esito dell' esperimento infatti, qualora la banca avesse un numero limitato di soci, non sarebbe in grado di reggere l'impatto dello shock poiché non avrebbe sufficiente liquidità.

In definitiva si può affermare che nonostante le condizioni limite imposte, il sistema Jak è sostenibile; questi risultati sono comunque subordinati al fatto che il numero di ripetizioni eseguite potrebbe essere non sufficiente.