Universita' degli Studi di Torino
Dipartimento di Scienze Economico-Sociali e Matematico-Statistiche

Tesi di Laurea Magistrale

Social Connections and Retirement Decision:

an agent-based model with NetLogo on the Italian case

Matricola:775452:
Piergaetano Battaglia

Thesis advisor:
Prof. P.Terna

Research supervisor:
Prof. S.Margarita

Thesis submitted in 2014

Contents

1 Introduction

Our world is composed by singular bodies included in a society interacting between each others, this is very clear in our modern view, what is less clear is that the society as a whole seems not only be the results of the interactions of the agents but also a character that influences the agents themself.

The events that take place in the world are composed by the individual behaviors and movements of the agents, but when this behaviors became common to the whole, or at least a big part, of the society, they influence the individual agent behavior in taking one way instead of another, for this reason the relation between personal behavior and society behavior is very thin and sensible, in other words when a big number of agents adopt a specific behavior we can say that society takes this behavior and influences back the single agent using it. The aim of this work is understand how social connections between agents can influence the retirement decision of the agents themself, a lot of work have been done on the personal incentives of the agents and how they respond if we assume that they act in a fully rational way, but less work have been made in understand how this response change in a framework of connected people with bounded rationality.

In this project we work a lot on this two main points:

- The network structure between the agents, and
- the idea of bounded rationality.

For the first element we try to create a social network structure that is a simple but effective approximation reflecting the relations between the agents in the real world, it has to hold some features like:

- low density coefficient, because in a social network structure contest is very rare to find network structure that have a number of connections near to the theorically possible of the system,
- high clustering coefficient, because in a social network structure the probabilty of having a relation increase if we have relations in common with other agents,
- more than one dimension, for this reason we allow the model for more than one network class.

For the second element we interpret bounded rationality in contrast with the standard assumption that we find in the litterature, in this sense our agents:

- look at the social sphere when they have to take their decisions, so their social situations influence directly their behaviors, and
- do not take necessarly the same decision when facing the same condition, in our simulation the response of the agents depends on some random selections that enlarge the complexity of the system.

In chapter one we make an overview over the pension system subject,

- first, we analyse what is happened inside the OECD member states in the last years and we find that if is true that they use different measures, they all try to obtain the same things. In other words the objectives of all the governments converge.
- Second, we present a summary of the last ten years of the Italian pension system in order to familiarize with the framework.
- Finally, we study tha last pension reform that is the one that is nowadays implemented and that we will have to deal with in the next years, in particullary the innovation of the flexibility of retirement and the adequacy to life expectancy principle.

Moreover, always in chapter one, we present the theorical framework of agent-based computational models with which we built our model, we define it also by the comparison with other methods, similar but not equal, such as Microsimulation and Multi-Agent System. We also analyse how this theorical framework is young and often subject to critiques, we try to help the reader to not fall in the common interpretation errors over the Agent-Based simulations.

In particullary we present two works, Aaron (2010) and Borella and Moscarola (2010), that have been inspiring during the building of our model and the drawing up of this document. They contains the inspiration that guide us to focus our efforts on this topic and with this methodology.

Finally we deal with the topic of network structure that is crucial in the model. We comment the importance that such a structures can have inside the society, their elements and the classical models that the litterature presents.

In the second chapter we present five versions of the model, each version represent a development of the previous one. From version A to version E we try to create a model that study the choice of retirement of agents endowed with social relations living inside a society, despite the fact that there are still a lot of possible improvements we think that the final results give us some usefull intuitions keeping the rules of the game simple as the agent-based tradition teach.

- Version A: we begin to built the network strucure thinking abount the concept of network classes.
- Version B: we change how we create the network structure and we begin to work on the procedures that let the network structure influences the behavior of the agents.
- Version C: this version create links that represent the social relations allowing the user to see the connections, moreover to obtain a dynamic network structure we start to develop a procedure that regulates the evolution of the network structure.

- Version D: here we deal implementation of the last pension reform, in particulary we define the idea replacement rate and its influence on the model.
- Version E: we increase the differentiation between the agents and we change the procedure that guide the evolution of the network structure.

In the third chapter we use the last version of the model described in chapter two, we try to understand the effects of the different variables of the model on the behavior of the agents with and without the social network structure. The usual proceeding that we use in the experiments is take a variable and see what happens if we modify its value ceteris paribus, the last experiment differ from the others because do not run the code with a specific setting, as the others do, but is a theoric discussion on the concept and implementation of the network structure in the model. The experiments are:

1. Population of the Agents, where we change the size of the population of the agents,
2. Additional Network Classes, where we add one at a time network classes to the agents,
3. More Connected Networks, where we increse the probability of create social relations,
4. Replacement Rate, where we modify the replacement rate guaranteed to the agents after retirement,
5. Hetereogeneity, where we increase the differences between the agents,
6. Network Structure, where we discuss about how we have built the network structure.

Now we want to spend some words on why we have an interest in this subject instead of another. The theme of pension system is very important in this very moment of history, in the last fifty years the life expectancy grow very much and the composition of the society change a lot. If we make reference to the OECD countries we see the change from growing countries in terms of population to a slack situation with a stagnant rate of population growth. This change impact very much the pension system but it does not change correspondingly, governments fail to adjust their pension system to the structural changes of their society and in the last years the problem explodes. Moreover our society does not change only in terms of life expectancy and composition of society but also in terms of complexity, the informations available increase exponentially (both the right and the wrong ones), a rivolution of the concepts of connectivity and speed of information exchange occur. In order to deal with this new fast-changing and highly connected society we need new instruments that help us to isolate the rules that govern the dynamics of its evolution and allow us to

improve our comprehension of the world. Luckily in my master degree i faced two courses that open my eyes on this topics, "Simulation Models for Economics" by professor P.Terna and "Economics of Savings and Pensions" by professor M.Rossi, during this courses born an interest over this theme and little by little grow in myself the awerness that this personal interest can be use in a fruitfull way in this work. Ultimately this study wants to contribute to the agent-based modelling litterature and show how we are far away from a full comprehension of a subject like the retirement behavior of agents included in a connected society.

2 Analysis: literature part

2.1 An overview on the Pension System theme

2.1.1 OECD situation from 2005-2013

We analyse the pension system characterisation inside the OECD (Organization for Economic Co-operation and Development) comunity; we find that the states members of this comunity are a good comparison with the Italian case, because they share a comparable level of economic development.

In 2005 the OECD report on pension systems underlines the existence of numerous typologies of retirement-income systems and note that the world bank's "three pillar" classification that divides between a publicly managed system with mandatory participation and the limited goal of reducing poverty among the old (first pillar), a privately managed mandatory savings system (second pillar) and voluntary savings (third pillar) has become confusing, in order to solve this problem they develop a "taxonomy that avoids the concept of pillar altogether. It aims, instead, for a global classification for pension plans, pension funds and pension entities that is descriptive and consistent over a range of countries with different retirement-income systems" Queisser and Whitehouse (2005). The approach followed is based on the roles and objectives of the different parts of the pension system. There are two mandatory tiers:

- a redistributive part and,
- an insurance part,

the first element ensures that pensioners achieve some absolute, minimum standard of living, while the objective of the second one is to achieve some target standard of living in retirement in relation to what they enjoy during working life. Voluntary provisions are considered the third tier.

All OECD countries have a first-tier scheme whose objective is prevent poverty of the elderly and, as we can see in the Figure 1, they divide into 4 categories.

- Social assistance.
- Basic pension schemes, where the same amount is paid to every pensioners or at least depend on the years of work.
- Targeted plans, that pay less to the rich retiree and more to poorer pensioners.
- Minimum pensions, that are similar to targeted plans excepts for the fact that pensions do not fall below a certain level and both the institutional set-up and the conditions of eligibility are different.

Moreover there are five country in which there are not precise targeted program for older people and Italy is one of these country. Half of the OECD countries

Tier: function	First tier: universal coverage, redistributive				Second tier: mandatory, insurance		
Provision	Public				Public	Private	
Type	Social assistance	Targeted	Basic	Minimum	Type	DB	DC
Australia		✓					✓
Austria		✓			DB		
Belgium		✓		✓	DB		
Canada		✓	✓		DB		
Czech Republic	✓		✓	✓	DB		
Denmark		✓	✓		DB/DC		✓
Finland		✓			DB		
France		✓		✓	DB + points		
Germany	✓				Points		
Greece		✓		✓	DB		
Hungary				✓	DB		✓
Iceland		✓				✓	
Ireland		✓	✓				
Italy	✓				Notional ac		
Japan			✓		DB		
Korea			✓		DB		
Luxembourg	✓		✓	✓	DB		
Mexico		✓					✓
Netherlands	✓		✓			✓	
New Zealand			✓				
Norway		✓	✓		Points		
Poland				✓	Notional ac		✓
Portugal		✓		✓	DB		
Slovak Republic				✓	Points		
Spain				✓	DB		
Sweden		✓			Notional ac	✓	✓
Switzerland		✓		✓	DB	Defined credit	
Turkey		✓		✓	DB		
United Kingdom		✓	✓	✓	DB		
United States		✓			DB		

Figure 1: Structure of pension systems in OECD countries, Queisser and Whitehouse (2005).

rely on one primary single instrument to prevent poverty, while the others have combination of two or three.

The aims of the second tier is to ensure that when retire people have an adequate replacement rate and not only a managing absolute standard of living. Like the first-tier is mandatory, for all but Ireland and New Zealand. Speaking about second-tier there are two possibilities:

- Define Benefit plans, the pension for each retiree depend on the number of year of contribution and on some measure of individual earnings from work,
- Define Contribution plans, where each worker has an individual account in which his contributions are stored and invested and which, after the retirement, will be converted in a pension income stream. There are also variations from the classic Define-Contribution model like the French and German cases, where workers earn pension points during their working life that will be converted in a stream of pension payments at retirement.
- Notional Account schemes, used by Italy, Poland and Sweden. Here each worker has an individual account where are stored the individual contributions and a rate of return applied on that sum. The word "notional" is due to the fact that both the contributions and the interest rate exist on the book of managing institution.

In the years after 2005 many governments took pension reform as important topic, "population ageing and declining fertility rates require reforms which also need to pre-empt, where possible, adverse social and economic effects of making pension systems more financially sustainable... it is important to consider the long term scenarios rather than short-term views." Reilly (2013)

The expenditure of the pension systems in the majority of the OECD countries are forecast to increase because of the rise in the life expectancy. This development is unsurprising and, for this reason, is a common conclusion that rules and pension systems have to change effectively. During the January 2009 and September 2013 a wind of change in the pension systems spread over and all countries that change follow some common key elements.

Pension system coverage in both mandatory and voluntary scheme, OECD countries have set up plans, either public or private, in order to achieve a deeper coverage. Policy acts in three main forms: private pension provisions in addition to public schemes, introduction or extension of mandatory occupational pensions, and automatic enrollment in voluntary schemes.

- Adequacy of retirement benefits, in order to address income replacement and redistribution.
- The financial sustainability and affordability of pension promises to taxpayers and contributors, to obtain a better situation regard the state budgets. A common measure has been the reform of pension indexation mechanism.

- Incentives that encourage people to work for longer parts of their lifetimes and to save more while in employment, there are three measures used by OECD countries to obtain that: increases in the statutory retirement age, improved provision of financial incentives to work beyond retirement age, and less or no early retirement scheme.
- Administrative efficiency to minimize pension system running cost.
- The diversification of retirement income sources across providers (public and private), the three pillars, and financing forms, there are four ways to diversify and secure savings: push voluntary pension plans, apply a regulation that allow the individuals with a greater choice over the way they can invest their retirement savings in private plans, relaxing the restrictions on investment options to obtain a greater diversification of pension fund's portfolios, and improve pension fund's solvency rates.

Trade-off and synergies among this elements are, of course, frequent; we can see the different elements of intervention by all countries members of the OECD in Figure 2

2.1.2 History of the Italian system

In 1992 the political crisis due to "Tangentopoli" and the crisis of the public accounts give the birth to a reform of the pension system, in particular in the 90's three different important phases can be underlined.

- Reform by Amato in 1992, the legislative decree 30 december 1992 n.503[1], try to balance social security spending and gross domestic product (GDP), introduce supplementary pension funds and financial statements, maintain and ensure an adequate mandatory pension for all. The retirement age is raised from 60 to 65 years for men and from 55 to 60 years for women. The minimum contribution for the retirement pension is increased from 15 to 20 years of contributions. The indexation of pensions is disconnected from the movement of the wage and related to the index of consumer prices (inflation) provided by ISTAT.
- Dini reform in 1995, law n.335[2] 8 August 1995, reflects an agreement signed between the Government and the social partners in 1995, the pension calculation system passes from the salary criterion (average earnings over the last 10 years of work) to the contributory system, the latter based on the actual amount of employee contributions during their working lives. The supplementary pension is governed by the start of the pension funds.
- Prodi reform, law n.449[3] of 27 December 1997, modify the Amato reform of 1992, adjusting with the agreements established between the government and the unions and with the need to reorder public finances, in

[1] Full official document of the legislative decree available at http://www.normattiva.it/
[2] Full official document of the law available at http://www.normattiva.it/
[3] Full official document of the law available at http://www.normattiva.it/

	Coverage	Adequacy	Sustainability	Work incentives	Administrative efficiency	Diversification/ security	Other
Australia	x	x	x	x	x		x
Austria	x	x	x				x
Belgium				x			
Canada	x		x	x		x	x
Chile	x	x			x	x	x
Czech Republic			x	x		x	
Denmark				x	x		
Estonia		x	x	x	x	x	
Finland	x	x	x	x		x	
France	x	x	x	x			x
Germany		x	x	x			
Greece		x	x	x	x		
Hungary		x	x	x		x	x
Iceland							x
Ireland	x		x	x		x	x
Israel	x	x				x	
Italy		x	x	x	x		
Japan	x	x	x		x		x
Korea	x		x		x		
Luxembourg	x		x	x			
Mexico		x			x	x	
Netherlands						x	
New Zealand		x	x				x
Norway		x	x	x			
Poland	x		x	x		x	
Portugal	x	x	x	x		x	
Slovak Republic			x		x	x	
Slovenia			x	x			
Slovenia	x	x	x	x	x	x	x
Spain		x	x	x			
Sweden		x	x	x	x	x	
Switzerland			x			x	
Turkey				x		x	x
United Kingdom	x	x	x	x	x	x	x
United States	x	x	x				

Figure 2: Overview of pension reform measures in 34 OECD countries, 2009-2013, Reilly (2013).

order to ensure the entry of Italy into the European Union. The Prodi reform is characterized by a tightening of the requirements for obtaining the age of retirement, for the increase of the burden of the contribution of self-employed, for the equalization of contribution rates of special funds to pension and for the elimination of certain conditions recognized to workers during the transition period to the contributory system.

So the Maroni reform of 2004, law n.243[4], has set in this context, raising the chronological age for seniority pensions: in particular, the age required to gain access to this form of retirement rises to 60 for all from 2008, without prejudice to the contribution requirement of 35 years. In 2010, the requirement will be raised to 61 years, in 2014 to 62. Alternative requirement, starting in 2008, as established by Law n.335 of 1995, for access to retirement is 40 years of contributions, regardless of age registry. For the self-employed the age requirements are higher than a year to those set out in the various deadlines for employees. The law n.243 also provides for the reduction from 4 to 2 of the windows of the requirements for those that reach old age of retirement, resulting in further raising the retirement age. Raising the age limit is not just about the salary sytem or mixed, but also for the contributory sytem. For workers whose pension is paid exclusively with this system, the minimum age requirement is expected to be raised to 60 for women and 65 for men. The men will also have access to retirement if the worker has made contributions for at least 35 years to 60, 61 or 62 years of age respectively in 2008, 2010 and 2014. Access to retirement will be possible regardless of the age requirement in the presence of a requirement of contributions equal to 40 years.

At the end of 2007 was adopted the Law of 24 December 2007 n.247[5], which modify the provisions of law no.243/2004 that have been brought into force from 1 January 2008 and which would have resulted in an immediate increase from 57 to 60 age of the person. The law n. 247/2007 has, however, provided for a modification of the requirements for entitlement to a retirement pension, but more gradually, and introduced, as from 1 July 2009, the "quota system". In particular, with regard to the requirements for employees to have access to a retirement pension are as follows: from 1 January 2008 to 30 June 2009 shall require at least 58 years of age and 35 years of contributions; from 1 July 2009 to 31 December 2010, the target position is 95 with a minimum age of 59 years and a minimum contribution of 35 years; from 1 January 2011 to 31 December 2012, the share to reach is 96 with an age of at least 60 years old and a minimum contribution of 35 years; with effect from 1 January 2013, the target position is 97 with a minimum age of 61 years and a minimum contribution of 35 years. At the same time, it is confirmed that the attainment of 40 years of seniority can be accessed with the requirements regardless of age registry.

So in the last twenty years the Italian pension system has faced a number of structural reforms, summarized in Figure 3, all actuated to obtain long-term sustainability. The pre-reform system (pre-1992) was characterized by a Defined-

[4]Full official document of the law available at http://www.normattiva.it/
[5]Full official document of the law available at http://www.normattiva.it/

Benefit pension formula based on the last few years of earnings combined with soft eligibility rule, without any actuarial correction for the age of retirement, the first reform maintain the Defined-Benefit system but reinforce the eligibility rule. The second reform marks the passage to a Define-Contribution system, and the 2008 reform modify the eligibility rule of existing system.

	Private employees			Self-employed		
Reform	Pension formula	Seniority pension	Old-age pension	Pension formula	Seniority pension	Old-age pension
Pre-1992	DB: average of last 5 years earnings × 0.02 × years of seniority	Men and women: min 35 years; max 40 years; no age requirements	Men: age 60+, seniority 15 years Women: age 55+, seniority 15 years	DB: average of last 10 years income × 0.02 × years of seniority (since the 1990 reform)	Same as private employees	Men: age 65+, seniority 15 years Women: age 60+, seniority 15 years
1992	DB: average of indexed lifetime earnings × 0.02 × years of seniority	Same as pre-1992	Men: age 65+, seniority 20 years Women: age 60+, seniority 20 years	DB: same as private employees	Same as private employees	Same as private employees
1995	NDC: lifetime payroll taxes capitalised at the GDP rate of growth; converted into annuity with actuarially fair coefficients	Men and women: possibility of retirement between ages 57 and 65, with a min of 5 years of seniority; accrued pension > 1.2 times the minimum old age allowance to claim pension before 65		NDC: same as private employees	Same as private employees	
2008	NDC: same as 1995	Men and women: min age 61+, min seniority 35 years + sum(age; seniority) >= 97	Same as 1992	NDC: same as 1995	Men and Women: min age 62+, min seniority 35 years + sum(age; seniority) >= 98	Same as private employees

Figure 3: Main characteristics of pension reforms in Italy, Borella and Moscarola (2010)

Finally, the Decree Law no.201[6] in 2011 (the famous Decree "save Italy", which contains the reform Fornero), on one hand there is a switch for all to the contribution system pro-rata from 1 January 2012; secondly, it further raises the minimum retirement age: the minimum retirement age is changed from 60 to 62 years for employees (which will become 64 in 2014, 65 in 2016 and 66 in 2018, the self-employed will have to work one more year), and 66 years for men. It introduces a flexible band of retirement, differentiated between women (63-70 years) and men (66-70 years): the worker who chooses to stay at work longer, get a pension increase. Finally, the length of service requirement is raised to 42 years and 1 month for men and 41 years and 1 month for women; since 2012, who will has gained such a requirement but not the minimum number of years required will suffer a penalty of 2 percent per year of less work.

2.1.3 The last reform by Monti-Fornero

The average age for admission to employment is today around 26 years, up to this age people are typically students, remaining in charge of the their fam-

[6] Full official document of the legislative decree available at http://www.normattiva.it/

ily. Once landed on the labor market we can expect a period between various temporary occupations, lasting a few years and then finally get a stable and permanent employment: it will be a place for self-employed or employee, will represent their main source of livelihood, as well as the bank for the old age pension.

The reform by Fornero in addition to holding full operation of the adjustment mechanism of the requirements for retirement to the "life expectancy", has introduced two novelties. The first is its extension to unique contribution requirement for early retirement; the second is the exchange step, starting from the adjustment after the year 2019, because since then the individual appointments with the life expectancy will no longer be held every three years, but two years.

Pensions and life expectancy The mechanism has been introduced in 2009, by the legislative decree n.78[7] of 2009 (ratified by Law no.102/2009) had introduced a general intervention aimed at all workers, both public and private, which provided that with effect from 1 January 2015, the requirements registrations to access to the Italian pension system should be adjusted to the increase in life expectancy established by Istat and validated by Eurostat, with reference to the previous five years. Subsequently, the legislative decree n.78[8] of 2010 (ratified by Law no.122/2010) gave effect to the provisions of the n. 78/2009, with some modification. That is providing the adjustment every three years (not five years as before) in order to adapt to the increase in life expectancy observed annually by ISTAT (no more validation Eurostat) no later than June 30, starting in 2015. Finally, it is arrived the n.98/2011, which brought backward to January 1, 2013 (instead of January 1, 2015) the date of the first adjustment to life expectancy. At the same time, has anticipated in 2011 (instead of 2014) the obligation for ISTAT to make available the data to the change in life expectancy. Moreover, it has postponed to December 31 of each year (instead of 30 June) the obligation for ISTAT tomake available the figure for the variation in the previous three years of life expectancy at the age corresponding to 65 years.

The mechanism, therefore, today is this: every three years is measured the change in the probability that a man and a woman aged 65 will survive (this is the "life expectancy"); if the probability increases (that is, if increase more years of "expected" life), although the age of retirement moves away of the same measure, otherwise the requirements remain unchanged. In this way, therefore, in the future we will know for sure when you start to work but not when you can stop (age), the uncertainty stems from the reform age for retirement. A variable, this age, which has been object of many changes in the last 30 years. With the Amato reform in 1992, there was a gradual increase from 55 to 60 years for women and from 60 to 65 for men for retirement in old age; with the Dini reform, in 1996, and with the Maroni reform later, in 2004, the age for retirement has increased gradually from 52 to 62 years, finally, the reform

[7] Full official document of the legislative decree available at http://www.normattiva.it/

[8] Full official document of the legislative decree available at http://www.normattiva.it/

Damiano (Welfare Protocol of 2007) introduced the so-called "quotas". Even the last reform put his hand the age requirements; however, the uncertainty on the actual age of retirement comes from having connected, automatically, all age groups of all pensions to the increase in the life expectancy that is established by Istat.

Life expactancy: a reform forever The mechanism of adaptation to the "life expectancy" presents repetitive effects during time. Every three years, in other words, it checks to see if there was variation in life expectancy calculated by ISTAT and, consequently and automatically, follow the upgrade requirements for retirement. This law, in particular, provides that with effect from 1 January 2013 age requirements and values of the sum of age and years of contributions to (the famous "quotas", deleted by the reform, except in some cases), the age requirements 65 years and 60 years for the attainment of the retirement pension, the requirement for the retirement of women in the public sector, the age requirement 65 years for social pension and contribution requirement for the attainment of right of access to retirement regardless of age, should be updated every three years by Decree of the Ministry of Economy and Finance in consultation with the Ministry of Labour, to be issued at least 12 months before the effective date of each update. From 2011 Istat is required to make available annually within December 31 of the same year, the figure for the change in the three years preceding life expectancy at age 65 years in the corresponding reference to the average of the resident population in Italy. In the event of a fraction of a month, the update is done with rounding off to the nearest.

In year 2012 the new reform state two different form of pension:

- old-age pension, that is what we want to study and,
- anticipated pension, that is less interesting for our analysis because we doesn't take it in consideration in the the agent-based model.

As in the old discipline, the old-age pension is awarded in the presence of a minimum contribution (20 years) and age is not less, when fully implemented, 66 years. Age, however remain subject to the adjustments to life expectancy.

The condition of age The age requirements, in details, for the achievement of the entitled to old-age pension for men and women, workers and self-employed, with effect from 1 January 2012:

- 62 years for employees (female), this age requirement is set at 63 years and six months with effect from the January 2014, 65 years with effect from 1 January 2016 and 66 years from 1 January 2018. In any case, firm the discipline of adaptation of the requirements for access to the system pensions to increases in life expectancy; therefore, following the publication of the Decree of 6 December 2011, the above requirements shall be increased by three months starting from 1 January 2013;

- 63 years and 6 months for self-employed workers (female), such age requirement is set at 64 years and six months with effect from 1 January 2014, to 65 years and 6 months with effect from 1 January 2016 and 66 years with effect from 1 January 2018, It is in any case firm discipline adaptation of the requirements for access to pension system to increases in life expectancy; therefore, following the publication of the Decree of 6 December 2011, the above requirements shall be increased by three months starting from 1 January 2013;

- for employees male in general and employees female of the public sector the age requirement for access to old-age pension is determined in 66 years,

- for self-employed workers (male) the age requirement for access to old-age pension contribution is calculated in 66 years.

The contribution requirement The right to a retirement pension is earned in presence of insurance records a minimum of 20 years, provided that the amount of pension of the workers who fully belong to the contributory scheme turns out to be no less than 1.5 times the amount of the social allowance. The predicted threshold amount equal to the year 2012, 1.5 times the amount of the social allowance, is re-evaluated annually on the basis of the variation of the five-year average of nominal gross domestic product (GDP), calculated by the National Institute of Statistics (Istat), with reference to the five-year period preceding the year to be reassessed. This minimum requirement doesn't hold if the worker has an age equal to 70 years, subject to insurance records of minimum 5 years.

Flexibility An absolute novelty of the new old-age pension is the flexibility which takes the form of a reward mechanism in favor of those who delay the access to pension up to 70 years. Who continues the work beyond the age of retirement, in other words, is awarded of the application of a "transformation coefficient" of measurement more convenient. These coefficients (which are the rates that applied to the total contribution give the measure of pension) will be predetermined until the age of 70 years (except subsequent adjustments to life expectancy). The limit of 70 is subject to the adjustment to life expectancy.

Art.18 up to 70 years To guarantee employees the possibility to use the new flexibility, the law is binding on the protection of stability of up to 70 years. In fact, states that the effectiveness of the provisions of Article 18 of Law no. 300/1970 (Workers's Statute) work until the attainment of the predicted maximum flexibility, which starts with the limit value of 70 years but then will be adjusted to life expectancy. The measure, of course, is to prevent that the company can to dismiss for reasons of age, once the employee has reached the minimum age of retirement.

2.2 Conputational Social Science

In a world in continuous change the understanding of the structure and function of society and the nature of its changes is crucial for governance and for the well being of people. The modern technologies and methods give us the possibility of acting in order to change the world and transform it in a better place, however to change the world we first have to understand the dynamics that govern it's development over the time. Humanity is currently facing great challenges Conte *et al.* (2012):

- change of the population structure (change of birth rate, migration);
- financial and economic instability (trust, consumption and investments, sovereign debt, taxation, and inflation/deflation, sustainability of social welfare systems, and so on);
- social, economic and political divide (among people of different gender, age, education, income, religion, culture, language, preferences);
- Threats against health (due to the spreading of epidemics, but also to unhealthy diets and habits);
- unbalance of power in a multi-polar world;
- organized crime, including cyber-crime, social unrest and war;
- uncertainty in institutional design and dynamics (regarding regulations, authority, corruption, balance between global and local, central and decentralized systems);
- unethical usage of communication and information systems (cyber risks, violation of privacy, misuse of sensitive data, spam).

The study of what are called Complex social systems, their features and the interactions between them are a new challenges of this century and represent our attempt in understanding the real world structure. The traditional tools of social sciences are not able to search deep in the mechanism that govern what happens, in this century when the speed of the change is very high. Complex social systems are characterized by multiple ontological levels with multidirectional connections, proceeding not only from the micro to the macroscopic levels but also back from the macro to the micro-levels.

> The new ICT (information and communication technologies) enabled study of society has been named computational social science. This is a truly interdisciplinary approach, where social and behavioral scientists, cognitive scientists, agent theorists, computer scientists, mathematicians and physicists cooperate side-by-side to come up with innovative and theory-grounded models of the target phenomena. Computational social scientists strongly believe that a new era has started in the understanding of the structure and function of our society at different levels, Conte *et al.* (2012).

The study of this field is focused on the emergence of all the collective phenomena and behaviors that spread out from individuals systems in interaction, there are three direction of the research: emergent social behavior, emergent social aggregates and emergent institutions.

Probably the most important features of computational social science is the "learning" property, this is an incredible fruitful way of approach, the idea that the agents can adjust their behavior, modify their expected conditions facing the reality of the facts,

> ... despite the conceptual and theoretical weaknesses of the models and techniques used, learning systems have had a strongly innovative effect on the study of social influence, Conte *et al.* (2012).

For example the works of the last years has revealed how learning dynamics can lead to solutions of the problem of cooperation that are out-of-equilibrium from a rational actor perspective, but may be robust when agents are boundedly rational. The non-equilibrium phenomena generated by learning dynamics are a great area of work for computational social science.

In the first few years computational social sciences had been used as a qualitative tools that help to understand social phenomena, but recently the achieving of a more quantitative orientation has gained importance to fully develop the potential of the technologies. This evolution is build on three directions: the assessment of the validity of simulations, use of quantitative massive data analysis, data driven simulations, made to compare with, understand, and predict real-life phenomena. Researchers are strongly working on the field of the validity of the simulation and on the use of quantitative massive data analysis. A branch that is particularly interesting what is called data-driven dynamical application system, that support interactions between computer agents and real world entities in a logic of input-output stream (ABMs are data-driven dynamical application system in this sense).

In what ICT characterized by social computational science can help?

- First it useful to deal with BigData, i.e. new type of massive data, for addressing the big problem that hurt our complex and fast changing society.
- Second provide a good and deep instrument in inspiring, formalizing and implementing the core scientific concepts, principles, and ideas of computational social science, giving a new way of âĂIJthinkingâĂİ about the problems.

This ICT infrastructure had favored the exchange of data between people and had allowed every person to be not only a receiver but also a provider of various type of information, the ICT infrastructure not only provide data exchange but also outsource productive tasks. So computational social science became

> ... a powerful tool for fostering our understanding of the complexities of real socio-economic systems, by building "virtual computational

> social worlds" that we can analyze, experiment with, feed with and test against empirical data on a hitherto unprecedented scale. A range of excellent papers has been written to make this point, Conte *et al.* (2012).

Another very important property is "adaptation", social complexity as an emergent phenomenon is caused by successful (or not) adaptation to challenging environments, can be caused also by uncertainty, probability and other scientific theories of uncertainty are important notions needed for understand social complexity (theory-driven models of opinion formation, revision and dynamics in interplay with other mental constructs, like beliefs). "Scaling" is another crucial aspect of computational social science, computational social theories, so also ABMs, can easily reproduce power law distribution, income distribution for example, but not everythings scale the same.

Computational social science can develop an important role in understanding the level and the directions of interaction, the classical view presented two levels of interaction, macro-level and micro-level but this environment is too simple, real-world society has several levels of complexity: an individual interact with other people but also with entities of all the other levels, social levels emerge from one another and retro-act to one another. In this environment emergence is studied, and is defined as phenomenon Conte *et al.* (2012):

- for which the conditions for its occurence are well defined,
- non-deliberately but spontaneously,
- modifying the entities involved, in particular, interfering with their fates but unlike learning, without modifying the internal states of the producing entities and,
- unlike evolutionary effects, non-transmissible.

Anyway isn't well understood because of:

- insufficient analysis of the emergence process,
- inadequate models of the micro-level interacting units from which emergence is supposed to proceed,
- unsatisfactory account of the coupling of emergence and downward causation.

The interconnection among different levels of social phenomena, cannot be fully accounted for unless multidirectional processes are modelled, including the downward process from higher-level properties to lower level entities, called Micro-Macro link. More precisely, the Micro-Macro link is the loop process by which behavior at the individual level generates higher-level structures (bottom-up process), which feedback to the lower level (top-down), sometimes reinforcing the producing behavior either directly or indirectly.

Another top-importance focus is the research for a model of good collective behavior, but a precondition to obtain that is a model of individual behavior. Usually in the literature is used the notion of "rational agent" but now is clear the rationality theory isn't sufficient. The study of this subject is focused on the importance of networks in the choice of the individual, and also in how and when different networks spread from individuals with different characteristics.

Models are used for three reasons:

- to make predictions about present system and so understand what will happens in the future, these model are output-oriented, so are primarily involve optimizing the models to make their output as accurate and precise as we need it to be,
- to explain properties or behavior of a target system, basically show as the component parts of the system create the behavior that can be observed, or in other words "show how things gone", and
- understand how hypothetical system works.

2.2.1 ABMs, Microsimulation and Multi-Agent System

At this point we can take a beautiful definition of ABMs:

> Compactly, in agent-based computational models a population of data structures representing individual agents is instantiated and permitted to interact. One then looks for systematic regularities, often at the macro-level, to emerge, that is, arise from the local interactions of the agents. The shorthand for this is that macroscopic regularities "grow" from the bottom-up. No equations governing the overall social structure are stipulated in multi-agent computational modeling, thus avoiding any aggregation or misspecification bias. Typically, the only equations present are those used by individual agents for decision-making. Different agents may have different decision rules and different information; usually, no agents have global information, and the behavioral rules involve bounded computational capacities, the agents are "simple". This relatively new methodology facilitates modeling agent heterogeneity, boundedly rational behavior, non-equilibrium dynamics, and spatial processes, Axtell (2006).

ABMs and Microsimulation The official definition of the International Microsimulation Association defined microsimulation as a modelling technique that operates at the level of individual units such a persons, households, vehicles or firms. All the units of the model are endowed with a record containing a unique identifier and a set of characteristics, moreover the development of the models is dominated by a set of rules that influence and change the behaviour of the units. These rules can be deterministic (with probability $= 1$) or stochastic (with probability < 1), the result is an estimate of the outcomes of applying

these rules. Over many steps an important results are the overall aggregate change of the model's state of being but more important is the way this change is distributed in the population that is being modelled.

Agent-Based is a class of models that simulate the actions and interactions between autonomous agents, taking both as individual or as collective entities (groups and organizations), with the view to understand their effects on the system as a whole trying to predict the appereance of complex phenomena. We take emersions from the lower lever of the system and see the effects on the higher level (micro to macro), in this sense the key notions are simple behaviour rules that generate complex behaviour (K.I.S.S. principle "keep it simple, stupid!") and that the whole is greater than the sum of the parts. Agents from this model are designed to be rational, acting perceiving their own interest using heuristics or simple decision rule, they experience "learning" and adaptation.

We can define microsimulation a methodology used in a large variety of scientific fields to stimulate the states and behaviours of different units (for examples individuals or firms) as they evolve in a given environment (for examples a market or a state). This view is generated by the works of Orcutt in the late 1950s. He pointed out the weakness of the macroeconomic models of the time in handling with subject as income redistribution or poverty, situation due to a lack of detailed information about the micro relationships of the behaviour and interaction of the elemental decision making unit. The innovation of microsimulation is use representative distribution of individuals and put emphasis on their heterogeneous decision making. Richiardi (2014) argues that Agent-Based Models (ABMs) are characterized by three features:

- there are a multitude of objects that interact with each other and with the environment
- these objects are autonomous, i.e. there is no central, or "top-down" control over their behaviour and more generally on the dynamics of the system
- the outcome of their interaction is numerically computed.

ABM are generally identified as theoretical exercises aimed at investigating the (unexpected) macro effects arising from the interaction of many individuals, each following possibly simple rules of behavior, or the (unknown) individual routines/strategies underlying some observed macro phenomenon. Microsimulation are more policy-oriented while ABM are more theory-oriented. Moreover microsimulation generally rely on partial equilibrium approach and so live under an economic equilibrium which takes in consideration only a part of the market, ceteris paribus, to attain equilibrium while ABM are often closed models.

Two dynamic microsimulation models are very important because in some sense represent the connection between the standard version of microsimulation models and the ABM: Bergmann's model of US Economy and Eliasson's model of the Swedish Economy. The former departs from the Orcutt standpoint because modelled the behaviour of the actors in a structural sense and not only in terms of transition probabilities between different states. The latter also had

a complete and structural model of economy, in particular of the behaviour of the firms and the workers.

So two approaches can be identified. The first one, which is standard in the dynamic microsimulation (DMS) literature, is to separately estimate the different processes (education, household formation, labor market participation, etc.) as reduced-form equations,

> ...this requires to assume that choices are made sequentially, so that all the covariates in every process can be considered as predetermined, Richiardi (2014), and

this assumption being often untenable, the practical solution is to keep the estimates separate and "adjust" the estimates (via alignment algorithms) in order to keep the evolution of some macro-variables of interest in line with exogenously given targets. This also takes care of specification errors, and of the fact that microsimulations generally lack general equilibrium feedback effects. However, the procedure is of dubious validity, from a theoretical point of view, always from Richiardi (2014):

> It is considered as a "quick and dirty" solution to deal with complicated models and inadequate data; it can (by construction) succeed in tracking real time series but it has no structural backbone and thus it is likely to fail to predict the effects of policy changes; moreover, it is not able to provide out-of-sample guidance when no external targets are available.

The second approach is the one followed by DSGE (Dynamic stochastic general equilibrium) modelling, which has evolved from rough calibration to structural estimation. Notwithstanding the fact that DSGE models are packed with simplifying assumptions, their estimation is by no means straightforward. The most standard technique is ML (Maximum Likelihood) and requires to linearize the model in order to find a local approximation of the steady state solution, then express this solution as a Markov chain, then apply filtering theory in order to obtain the likelihood function. ABMs are relatively young related to DMS and nowadays we don't have empirical validations of their results, the hypothetical good properties of a large scale ABM will decide if they substitute DMS standard view (sequential approach) and establish their validity, also over and beyond DSGE models.

ABMs and Multi-Agent System Martin Sewell define MAS (Multi-Agent Systems),

> A multi-agent system is a system in which several interacting, autonomous, intelligent agents pursue some set of goals or perform some set of tasks.

Comparing this definition with the concept of AMBs may look that, in fact, there is not a difference but this is only partly true, obviously the two concepts overlap each others but we can say that they are not exactly the same for some reason.

- We have to define the meaning of intelligent, in ABMs for examples we can create agents that are not totally rationa, rational but that have limited informations or that decide using some simply rules, while in MAS the mainstream interpretation of "intelligent" is rational and self-interested.
- hystorically ABMs and MAS are used in different areas of research, the terminology of ABMs tends to be used more often in the sciences, and MAS in engineering and technology.

Generally speaking we can say MAS are not always the same as an AMBs, but there are many points in common and they share the idea of the additional complexity deriving from the presence and the interaction of many agents.

2.2.2 ABMs

Another nice definition of simulation that fit perfectly for ABMs:

> Simulation in general, and ABM in particular, is a third way of doing science in addition to deduction and induction. Scientists use deduction to derive theorems from assumptions, and induction to find patterns in empirical data. Simulation, like deduction, starts with a set of explicit assumptions. But unlike deduction, simulation does not prove theorems with generality. Instead, simulation generates data suitable for analysis by induction. Nevertheless, unlike typical induction, the simulated data come from a rigorously specified set of assumptions regarding an actual or proposed system of interest rather than direct measurements of the real world. Consequently, simulation differs from standard deduction and induction in both its implementation and its goals. Simulation permits increased understanding of systems through controlled computational experiments, Axelrod and Tesfatsion (2006).

Understand a political/economic/social system is more than understand the behavior of individuals in isolation; in some sense the big picture is more than the sum of the forms and colors and this difference is made by the interaction of the agents (people) and the effect of this interaction back on the single agent behavior, the so called micro-macro effect. Axelrod and Tesfatsion (2006) indicate that ABMs are very good for this type of study because they exhibit the following two properties:

- the system is composed of interacting agents and,

- the system exhibits emergent properties, that is, properties arising from the interactions of the agents that cannot be deduced simply by aggregating the properties of the agents.

When the interaction of the agents is contingent on past experience, and especially when the agents continually adapt to that experience, mathematical analysis is typically very limited in its ability to derive the dynamic consequences Researchers pursuit four goals during their work:

- Empirical - Question: why have large regularity that born, grow and persist without central control? try to find the grounded explanations in the repeated interactions of agents operating in a defined environment.
- Normative - Question: how we can use ABMs to discover the good design hidden in the reality of what happens? Create an environment where the design is applied, put inside agents motivated by self interest and characterized by learning capabilities and possibility of develop, and capture the salient aspects of the system, checking if the outcome is efficient.
- Heuristic - Question: how can greater insight be attained about the fundamental causal mechanism in social science?
- Methodological - Question: what are the best methods and tools that we must use to obtain the best from ABMs technology?

So, social science and computer science provide concepts and tools that ABMs use when they have to face social processes, this new methodological approach can be very useful in order to enrich and refine existing theories in the areas where math and statistics methods fail and also reach a deeper understanding of the mechanism of multi-agent systems.

2.2.3 Skeptical Minds

This is not an easy problem, but unfortunately is crucial for the future of ABMs approach and, probably, if in the last years the population and notorious of this field isn't grow at high rate is also "our" fault.

In Waldherr and Wijermans (2013) are showed the ten more common critiques received by non-modelling peers.

- Your model is too complex: normally this problem show up because the counterpart compare ABMS with an equation-based approach and perceive the first like something complicated and not clear with a huge number of variable without a definite meaning, while the second as a logical and comprensible line of reasoning.
- Your model is too simple: ironically, this is exactly the opposite of the previous point, the modelers are often accused of over-simplify the reality. This is a strange critic because every model by definition simplify something (many times doesn't make sense built-up a model equal to the real world, because we want to isolate the effect we want to study).

- You chose the wrong focus: this often happens when modelers talk to scientists that are that are experts on their modeling targets, in their vision if the model doesn't include every single factor it is not valuable.
- Your model is not theory-based: every ABMs have theory behind that is represent by the assumptions that are explicity implement in the source code, but this is not theory in common sense so often this cause rejection of the entire model.
- Your model is not realistic: "empirical validation" represent another big problem that sometimes appears, instead of concentrate on the fact that ABMs show some evidences of non-observable process the critique is build on the fact that the result of the model isn't comparing to the empirical data.
- Your assumption and parameters are arbitrary: the criticism concerning theory and empirical data both boil down the question where the model's assumptions come from.
- Your results are built into the model: sometimes the results of the model are strongly related to some parameters choosing by the modelers, since they can easily modify these parameters the point is that you can (theorically) obtain every results you want (or you need) only changing few numbers.
- Your model is a black-box: this reaction is normally due to lack of understanding in the audience or to a high degree of complexity of the model, or in the worst case to both.
- This is not science!: this implicate normally a full rejection of the model from an ideological point of view, the core of the scepticism is due to the fact that ABMs don't follow an orthodox mathematical approach.
- Your model is not useful: it is a variation of "this-is-not-science".

Feedback can be good to improve the quality of the work, it may be helpful when addressing substantial weakness of the work. Anyways often criticism is regarded not directly to our model but try to reject the whole methodology and puts the presenter in the position of defending not his model, but all the agent-based approach and theory.

There are two source for rejection, Waldherr and Wijermans (2013):

- lack of understanding and,
- academic territorialism.

In the authors view the first case is open to good conclusion, the presenter has to improve direct communication and preparation with his counterpart. The audience can think that the huge number of variable is due to a bad comprension of the subject and moreover a clash of research goals also might produce

misunderstanding, a scholar expecting prediction from a model that was designed to contribute to understanding, or demanding empirical validation from a model meant to explore future scenarios. Operation like being more clear, understanding what is not understood, relating the subject to your audience and being more transparent about your choice in the model can really help to create constructive communication that now can yield to a constructive criticism. In the second case the paper presents a completely different approach, when the counterpart refuse the listen about your work only because he is not interest about the method or more he doesn't think that your way is valuable at all, you have to switch the conversation to a more abstract level, push the conversation to a meta-level communication and refuse to enter in the cycle of mutual rejection. It is important in this types of situation make explicit what is going on in the conversation, the question is not whether one method is better than another but which one is the best in the peculiar situation of that research, a method can be better without delegitimized the other (that is more suited for different situations). In the end good discussions (the meaning of good is that can yield to a constructive feedback) are the ones that presented high level of understanding and low level of academic territorialism.

2.3 Core Lecture

2.3.1 Behavioral Dimension of Retirement Economics

The methods used in this book are drawn from the field of Behavioral Economics, that put together efforts by economists, phsycologists, sociologists and legal scholars that try to mix their research findings from their disciplines and at the same time retain the rigor and statistical precision common of economic theory and econometrics. This book is a concentration of notions that overcome the standard assumption of classical economy in order to obtain instruments more useful in order to understand the mechanisms that influence retirement. The book proposes many innovative point of view on the retirement subject, one of this, Axtell (2006), Became crucial in the inspiration of our work.

Discussion of classical assumption The classical economists framework assume that people make lifetime plans regarding the timing of work and retirement and that when a new information became available they modify their plans respect to these new conditions, so they modify their behavior immediately. The evidences show that this is not the case, normally the adjustment process of people to new informations is slower and gradual. People normally do not have a clear and defined set of preferences, they lack information and mental capacity to analyze the data available to them and to get a full analysis of all the implications for their decisions. Life-cycle model sustain that that people plan consumption using their estimates of future income flows that they expect to earn in their life, but, again, evidence reject this idea indicating that people saves much less that what the model forecast and a large minority saves nothing all without even think about retirement.

O'Donoghue and Rabin study the idea of "procrastination", this phenomenon is not considered in the standard economic view but could explain the failure of people to save adequately. In short they fail in understand that today's incentives for delay savings will happens every day and underrate the cost of waiting to start retirement savings.

Axtell and Epstein explore the possibility that a big part of the people do not understand totally the rules (lack of information or other reasons) and so do not respond directly to their changes but are influenced indirectly throught the interactions with other individuals or other random events. They prove that also if a small percent of people are "rational", i.e. respond to the rules directly, all the society as a whole reach the point where everyone acting like if he is "rational", but time is needed to complete the change behavior process. These delay is due to the fact that the non-rational agents retire random or when a fraction of the agents included in their social networks retire.

Fetherstonhaugh and Ross explain how people's responses to public policies depend on how public policies are presented and whether the effects of retirement policies are subject to "wealth illusion". In other words if the reference point with whom the agents compare their decisions have an impact on the decisions themselves, while wealth illusion concern the tendency of agents to prefers lump sums to streams of payment of equal (sometimes greater) present value.

Lowewenstein, Prelec, and Weber find evidences indicating that person's sense of well-being does not deteriorate at retirement, in addition the not-yet-retired do not forecast the effect of retirement on their well-being and so do not choose accurately the right time to effectly retire. These insights contrast keys assumption of standard economic analysis.

Finally, Shelly Lundberg find that the behaviors of married workers in retirement's subject is the result of the interaction of the actions of both spouses and explain how game-theory can be applied to to the analysis of this types of retirement decision.

Coordination in Transient Social Networks: An Agent-Based Computational Model of the Timing of Retirement Axtell and Epstein We find inspiration for our model from the chapter 5 from Aaron (2010) that propose an agent-based model by Axtell and Epstein.

The chapter has two areas of theoretic interest: first investigates the relationship between individual rationality and aggregate efficiency, and in addition studies the role of social interactions and social networks in determining the macro outputs and dynamics. Axtell and Epstein start from the evidence, supported by experimental economics and psychology that individual rationality is bounded and try to compute the influence of this approach building a model in which imitation has an important role in reach high aggregate level of optimal behavior, in other words try to study how "imitative behavior and social interactions... may be fundamental in explaining the sluggish response policy" Axtell (2006).

The model has three categories of agents, one minority, we can call them the

"rational", respond optimally to the policy and its changes, another minority that retire with a fix probability, we can call them "random" agents, and finally the last category, the majority of the agents, that mimic the behaviors of the members of their social networks, we can call them "imitators". The population of the model is divided into 81 cohorts from age twenty to age one hundred, every cohorts contains a number C of agents and to each agents is assign a random age of death drawn from U[60,100], every time an agent die is immediately replaced by a new twenty years old agent. Every period the agents that can retire decide whether they do it or not. Moreover the agents are endowed with heterogeneous social networks, the network size, a value S, is drawn from U[a,b] and the extent of the social network (how far the network's agent extends up and down the ages), a value E, is drawn from U[o,c]. This concept of the network have a strong influence on the behavior of one category of agents, the imitators. Every imitators is endowed by a unique social network, the model define f the fraction of the agents in a single network that have retired, obviously this is heterogeneous in every moments for all the imitators, in addition an imitation threshold, t, is assigned too. The rule that govern the behavior of imitators is comparing f and t, when f is bigger they retire otherway do not.

Two experiments are presented using the model, one with a large fraction of rational agents and another with few ones, all the other parameters are kept fix: C =100, the size of the networks is drawn from U[10,25], E =5, t = 0.5 for all the agents and random agents retire with probability p =0.5 each period. In the first experiments 15 percent of agents are rational, 80 percent are imitators and 5 percent are randoms, while in the second experiments the proportion are 5 percent rationals, 90 percent imitators, 5 percent randoms. In Figure 4 can be seen a time series plot of the fraction of agents eligible for retirement who actually are retired, within the first six period essentially all the eligible population retires.

In the second experiments change the composition of population and retirement age show a much longer fluctuations before full converge to age sixty-five, as show Figure 5.

Obviously in order to characterize quantitatively the model many realizations for every "set" of parameters are needed, but with a sufficiently large number of realizations the intrinsic sthochasticity of the model can be approximate.

Moreover Axtell and Epstein modified each parameter ceteris paribus in order to understand their effect on the time required for the retirement norm to emerge, defined transition time, and discover some interesting findings. First, the number of agents per cohort have no effect on the average transition time for $C > 100$. Second, reducing the proportion of rationals, while holding constant the proportion of randoms, increases transition time, however for a given fractions of rationals, the transition time decreases as the proportion of randoms increases, Figure 6.

Third, increasing the variance in the threshold decreases the average transition time, this is due to the fact that the low threshold agents retire quickly and influence the rest of the population to do the same, Figure 7.

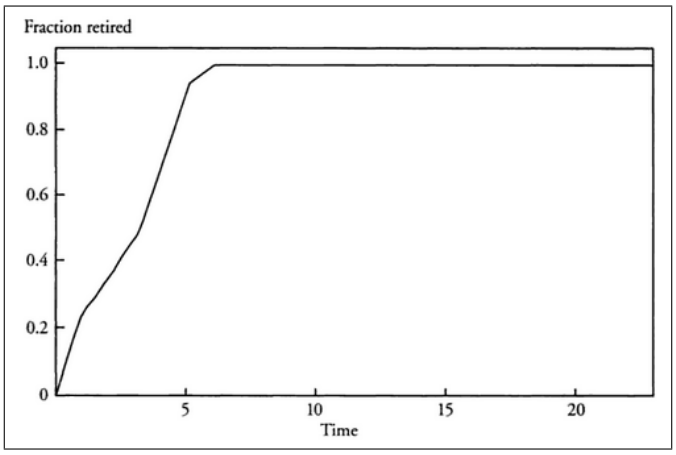

Figure 4: Fraction of eligible agents retired over time, typical realization, 15 percent rational agents; Aaron (2010).

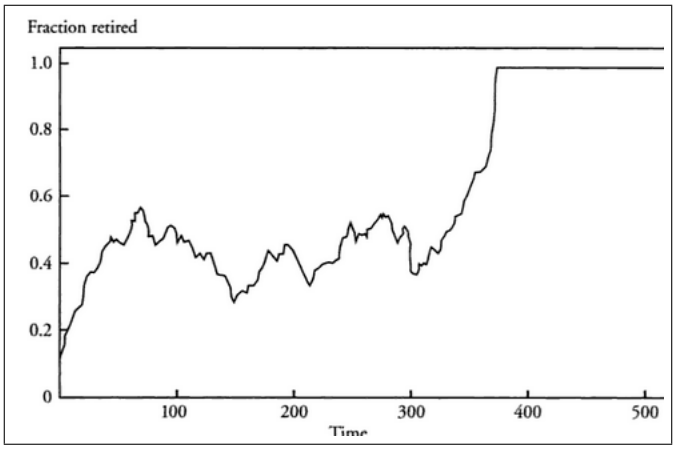

Figure 5: Fraction of eligible agents retired over time, typical realization, 5 percent rational agents; Aaron (2010).

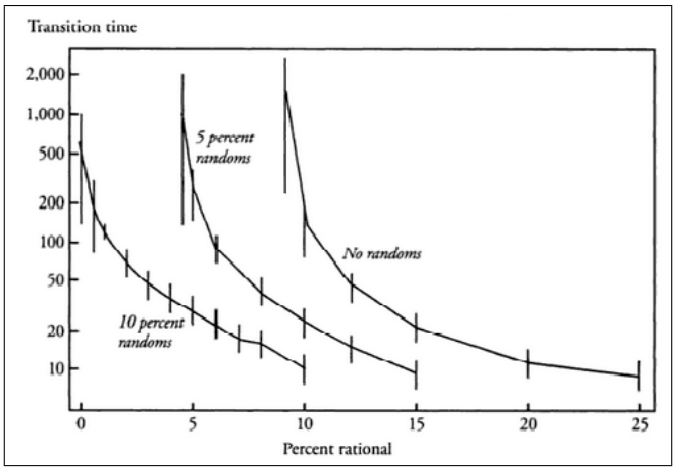

Figure 6: Effect of fractions of rational agents on transition time to age 65 retirement norm, by fraction of randomly behaving agents; Aaron (2010).

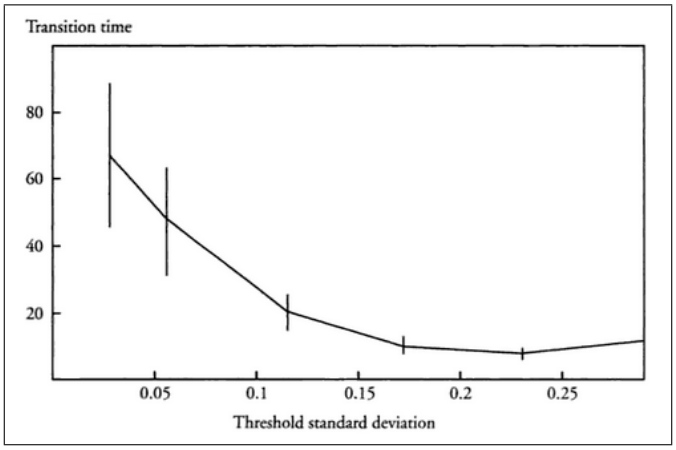

Figure 7: Effect of standard deviation in imitation threshold on transition time to age 65 retirement norm; Aaron (2010).

Fourth, transition time increase with the size of agent networks but this is the resulting flow generate by two contrasting effects: change in average size of networks increase transition time, while higher variance in the size of networks reduce transition time, anyway the first effect is stronger so the size of networks increase transition time, Figure 8 and Figure 9.

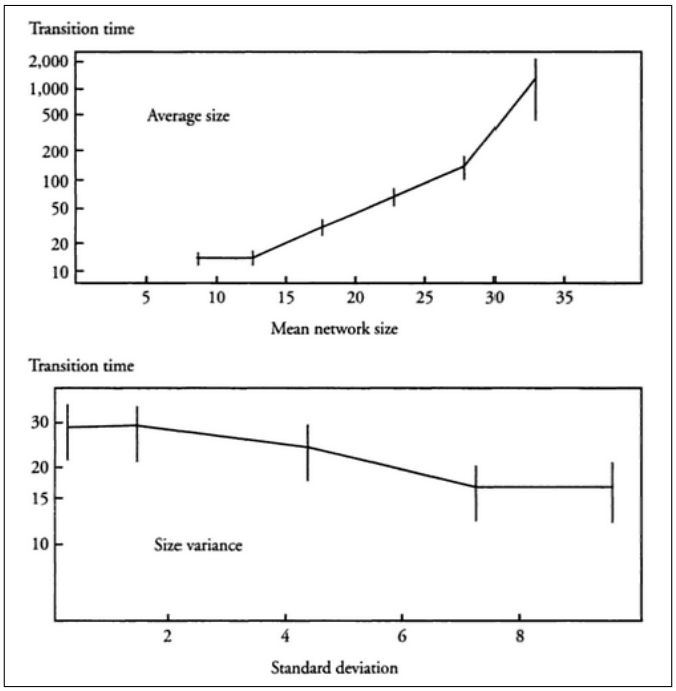

Figure 8: Effect of average and variance of the size of social network on transition time to age 65 retirement norm; Aaron (2010).

Finally, the effect of an increase in the extent of the agents (E) decrease transition time, because including older agents in the networks make more likely to retire the population.

As a conclusion of his model Axtell and Epstein in the book Aaron (2010) argues that in a society composed by a mayor part of imitative agents the rationality of all the individuals is not important and in some sense can be considered unnecessary, a small group of rational agents can guide the whole society to face the changing conditions of the environment. This argument is supported by the fact that in real life we experience a period of adaptation of the society to the social norms, that may indicate the time needed to the rational people to convert the others. From this model we take some the ideas:

- every individual has a network group, composed by people with whom he has a relation,

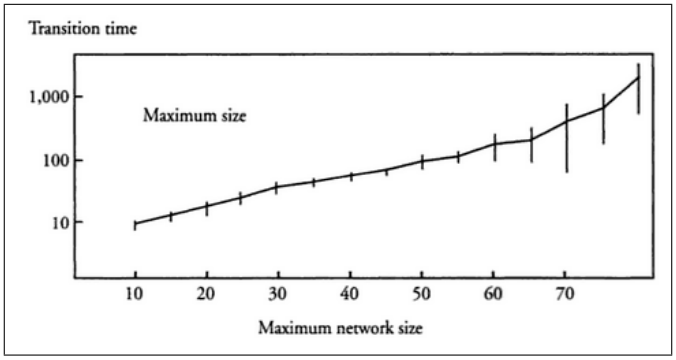

Figure 9: Effect of maximun size of social network on transition time to age 65 retirement norm; Aaron (2010)-

- the state of the art of these people on some subject, in our case retirement, influence the decision of the individual,
- the individual takes people of his network group and count how many of them for each states of the art about the subject he has and decides using a thresholds.

2.3.2 Microsimulation of pension reforms: behavioural versus non-behavioural approach,

The second lecture that in fact inspire this work is the paper "Microsimulation of pension reforms: behavioural versus non-behavioural approach" by Borella and Moscarola (2010), in this work they analyze the impact of various pension regimes on retirement age, adequacy issue, and redistribution.

In their analysis they use CeRPSIM2 that is a "dynamic partial-equilibrium microsimulation model of the social security system which relies on a behavioural rule for modelling retirement decisions", Borella and Moscarola (2010). We have already discussed about the difference between microsimulation model and agent-based model, and also about their points in common, so now we breafly present their result as well as the elements have been inspiring for our model.

The result of this work is based on the simulation of cohort 1995, all the four pension system is applied in turn so we can evaluate the results and look at the passage from a DB system to a NDC system. In the non-behavioural scenario the individuals retire as soon as they can while in the behavioural one they decide in a probabilistic way, they use the retirement decision rule estimated by Alessie and Belloni 2009, that estimates well the key role played by the economic incentives to retirement implicit in the pension scheme.

The effect of the reforms on the retirement age Clearly the reform process lead to an increase in average retirement age, as you can see in Figure 10,

this trend is due to two effect: first the increase of the minimum age requirement (nonbehavioral scenario)and second the decrease of financial incentives to retire early (behavioral scenario). Taking in account the nonbehavioral scenario men retire around 57.9 years and women around 55.4 years in pre-1992 system while after 2008 reform men retire at 62.2 years and women at 61.4. Instead when individuals can choose their retirement ages rise more reaching 62.8 years for men, start from 59, and 61.5 for women, start from 56.2.

		Nonbehavioural		Behavioural	
Reform		**Men**	**Women**	**Men**	**Women**
Pre-1992	Age	57.9	55.4	59.0	56.2
	Freq.	*3612*	*2216*	*3580*	*2214*
1992	Age	59.0	57.6	60.1	57.9
	Freq.	*3588*	*2208*	*3552*	*2207*
1995	Age	58.4	59.9	60.2	60.3
	Freq.	*3599*	*2196*	*3540*	*2194*
2008	Age	62.2	61.4	62.8	61.5
	Freq.	*3486*	*2186*	*3448*	*2185*

Figure 10: Average age of retirement by reform, gender, and retirement rule; Borella and Moscarola (2010)

However the results are less easy to understand if we allow for the difference between self-employed and private workers as show in the Figure 11.This is due to the fact that retirement decisions depend on many factors, individual preferences for non-working time, marginal financial incentives to retire early, and expectations about future consumption needs. The first two factors make convenient for the individual to retire early the third has the opposite effect.

In order to understand if the conditions imposed by the 2008 reform are binding they compare the non-behavioural scenario of 2008 with the behavioural one by 1995; the results indicate that the reform of 2008 is binding for private employees but not for self-employed.

The effect of the reform on adequacy This is the most debate points in the NDC reform because the common perception is that while the DB system provide an adequate level of benefits; the 2008 reforms does not. Figure 12 below show the results, the RRs that are associated with every reforms, types of work, schemes and gender.

The situation in pre-1992 system find a RRs in a range of 63-70 percent in the nonbehavioral scenario and of 65-73 in the behavioral one. After 1992 all the percents rise both for the minimum value and the optimal, this is due to the increase of the minimum retirement age and to the higher revaluation coefficients. When NDC system happens (1995 reform) it was followed by a decrease of the RRs granted, in particularly self-employed show a big reduction which

Reform		Nonbehavioural				Behavioural			
		Men		Women		Men		Women	
		Private	Self	Private	Self	Private	Self	Private	Self
Pre-1992	Age	58.0	57.4	55.1	57.4	58.8	59.9	55.9	58.1
	Freq.	*2983*	*629*	*1888*	*328*	*2967*	*613*	*1886*	*328*
1992	Age	59.3	57.4	57.7	57.4	60.5	58.0	58.0	57.6
	Freq.	*2959*	*629*	*1880*	*328*	*2931*	*621*	*1879*	*328*
1995	Age	57.7	61.6	59.5	62.0	59.7	62.7	59.9	62.2
	Freq.	*2999*	*600*	*1868*	*328*	*2954*	*586*	*1866*	*328*
2008	Age	62.2	62.5	61.2	62.4	62.8	63.0	61.4	62.4
	Freq.	*2888*	*598*	*1858*	*328*	*2864*	*584*	*1857*	*328*

Figure 11: Average age of retirement by reform, gender, working scheme, and retirement rule; Borella and Moscarola (2010).

		Nonbehavioural				Behavioural			
		Men		Women		Men		Women	
		Private	Self	Private	Self	Private	Self	Private	Self
Pre-1992	**Median**	**66.38**	**69.85**	**63.25**	**67.19**	**67.25**	**72.67**	**64.59**	**67.55**
	Mean	66.93	69.4	66.97	67.7	67.78	72.82	68.54	68.22
	Freq	*2983*	*629*	*1888*	*328*	*2967*	*613*	*1886*	*328*
1992	**Median**	**68.86**	**72.98**	**69.05**	**69.95**	**70.24**	**73.48**	**69.21**	**69.77**
	Mean	69.85	72.44	72.67	70.07	71.38	74.04	73.05	70.07
	Freq	*2959*	*629*	*1880*	*328*	*2931*	*621*	*1879*	*328*
1995	**Median**	**48.17**	**38.81**	**53.92**	**37.42**	**53.65**	**41.04**	**54.64**	**37.9**
	Mean	50.57	38.19	58.68	37.33	57.66	40.56	60.09	37.61
	Freq	*2999*	*600*	*1868*	*328*	*2954*	*586*	*1866*	*328*
2008	**Median**	**63.82**	**41.98**	**59.67**	**39.35**	**65.99**	**43.26**	**59.82**	**39.38**
	Mean	66.6	41.43	65.65	39.36	69.07	43.02	66.31	39.53
	Freq	*2888*	*598*	*1858*	*328*	*2864*	*584*	*1857*	*328*
2008	**Median**	**76.35**	**55.35**	**70.92**	**51.25**	**78.91**	**57.16**	**71.11**	**51.28**
I+II pillar	Mean	79.78	54.79	77.90	51.32	82.80	56.96	78.68	51.55
	Freq	*2888*	*598*	*1858*	*328*	*2864*	*584*	*1857*	*328*

Figure 12: RR by reform, scheme, gnder, and retirement rule; Borella and Moscarola (2010).

cause is the lower payroll taxes (20 versus 33 percent for employes) which was unaccounted for DB system, the 2008 reform increase the minimum retirement age and so raises all the RRs. The introduction of NDC system obliged the legislator to stimulate the contribution to the second pillar, and this can help to maintain adequate pension benefits, of course with the risk related to the second pillar.

Effect of reform on system's distributive properties Here the analysis use present value ratio (PVR) to measure the intergenerational distributive impact of the pension system. The pre-1992 system was highly generous and highly redistributive, but such a redistribution is both towards lower income and higher income individuals, the PVR of private employes is around 1.6-1.9 and for self-employers 3.2-3,6, so DB system provide benefits to workers two-three times higher than the value of contributions. Instead the NDC system will be actuarial fair and canceling out almost all the redistribution, the value of PVR inside the range 0.99-1.2 so very close to equality between contributions paid and benefits received. This is show in Figure 13 below for what concern the behavioral scenario, the table also provide the difference between the average PVR in the lowest and in the highest income quartile. The 2008 system makes a general reduction in redistributive impact but is also slightly progressive thanks to the minimum old-age allowance to which low-income pension people are entitled.

	PVR				PVR(25)-PVR(75)			
	Men		Women		Men		Women	
	Private	Self	Private	Self	Private	Self	Private	Self
Pre-1992	1.595	3.192	1.856	3.559	0.181	0.333	0.234	0.047
1992	1.522	3.146	1.719	3.270	0.236	0.089	0.261	0.046
1995	0.999	1.057	1.032	1.086	0.016	0.004	0.017	0.006
2008	0.990	0.996	1.019	1.020	0.012	-0.006	0.013	0.008

Figure 13: mediam PVR by gender, scheme, and incom quartiles - behavioural scenario; Borella and Moscarola (2010).

Conclusions The shift from a generous DB system to a NDC system provoke a postponement of the retirement of the individual in order to preserve the adequacy of the pension system. Using behavioral and non behavioral approach allow to find many consequences of the various pension system deeper hidden in the data, this model discover two less-know facts: first, also the pre-reform scenario contained incentives for some categories to postpone retirement, and second women tend to postpone less than man, showing a higher preference for non-working time.

2.4 Network Structure

> Relationships among individuals have a number of different dimensions. The first task is therefore to find a conceptual framework within which they can be described and then measured in a meaningful manner. We also need a language within which variations in relationships can be naturally expressed. The concept of a network addresses these requirements nicely: a network describes a collection of nodes and the links between them. The notion of nodes is fairly general: they may be individuals or firms or countries, or even collections of such entities. A link between two nodes signifies a direct relation between them; for mutual defense pact, Goyal (2012)

So a social network is a structure made up by a set of actors that establish dyadic connections between them, this structure is both emergent from and determinant of the actions of the agents.

We use social networks to simulate the social relations existing in the life of human being, we do not expect to be comprehensive of all the network classes that composed the portrait of a real person but we simplify their number and complexity in order to obtain indications about their power in explaining social behaviors and choices, Goyal (2012) said,

> Casual observation as well as introspection suggests that our behavior is influenced significantly by the actions of our neighbors, friends, and acquaintances. The behavior of our friends is influenced by their friends, whose behavior is in turn influenced by the behavior of their friends, and so on. We are thus led to the view that individual behavior is shaped by the entire structure of relationships that obtain in a social or economic context.

In other words there is a macro effect that simultaneously born from the union of the behaviors of many agents interacting each others with simply rules and no vision of the whole structure, and the back influence of this macro element on the single agent's behaviors. Social network structures are big sources of complexity, in fact is very difficult that exactly the same structure appear twice and if we add features in the nodes we can easily say that every personal network is different from another because the probability of find the same neightboruds in the same position building the same composition are very little. In addition if we enlarge the point of view we can see as, in a big network, every node faces a unique network environment, also if we assume that there are two nodes having the same position and facing two identical networks with nodes endowed by equal features to obtain exactly the same situation we also have to assume other two things: first that the two nodes in exam have the same identical characteristics and preferences and that the neightbourods of the neighbors have, again, equal characteristics (if they have not their influence will be different).

As we can read in Hamill and Gilbert (2009) the elements that composed a social network are two: nodes and links, which combine to give birth to paths. The main characteristics of node are his degree of connectivity and his clustering coefficient,

- the first represent the number of links to or from the node,
- the second indicates the extent to which the nodes connected to a given node are in turn linked to each other, that is measured by the ratio of the actual number to the maximum possible number of links.

Moreover the basic characteristics of the network as a whole are size, paths length, and density:

- size is simply measured by the number of nodes or links,
- path length indicates the distance between a pair of nodes measured by the number of links between the pair, links and nodes can appear only one time in each path and,
- the network density is the ration of the existing links to the total possible.

2.4.1 Classical Network models

The literature present four types of basic network model, they are regular lattice, random, small-world and preferential attachment (also called scale-free).

Regular lattice Regular networks are "regular" because each node has exactly the same number of links. Regular networks are highly ordered. No statistical rule is needed to define the lattice's degree distribution because the number of degrees is the same for each node, for example as in Figure 14 each node is "clustered" (connected) to four near neighbor nodes, The circular form makes it easy to demonstrate the effects of short path lengths and clustering on connectivity. Two nodes placed on opposite sides of the circle lattice are members of separate clusters that do not intersect, there are relatively long path lengths (low connectivity) between these remote nodes. Two nodes that are within three nodes of each other on the circle belong to separate but intersecting clusters, this is equivalent to two groups of friends where some people belong to both groups (this kind of intersecting clustering enhances connectivity within the network). A "regular" network architecture, such as the lattice, does not offer high connectivity in many cases because a long and circuitous route is required to reach many nodes.

Random A random network, is a theoretical construct which contains links that are chosen completely at random with equal probability, random network is considered to be highly disordered. Using a random number generator, one assigns links from one node to a second node, this random links typically result in shortcuts to remote nodes, thus shortening the path length to otherwise distant

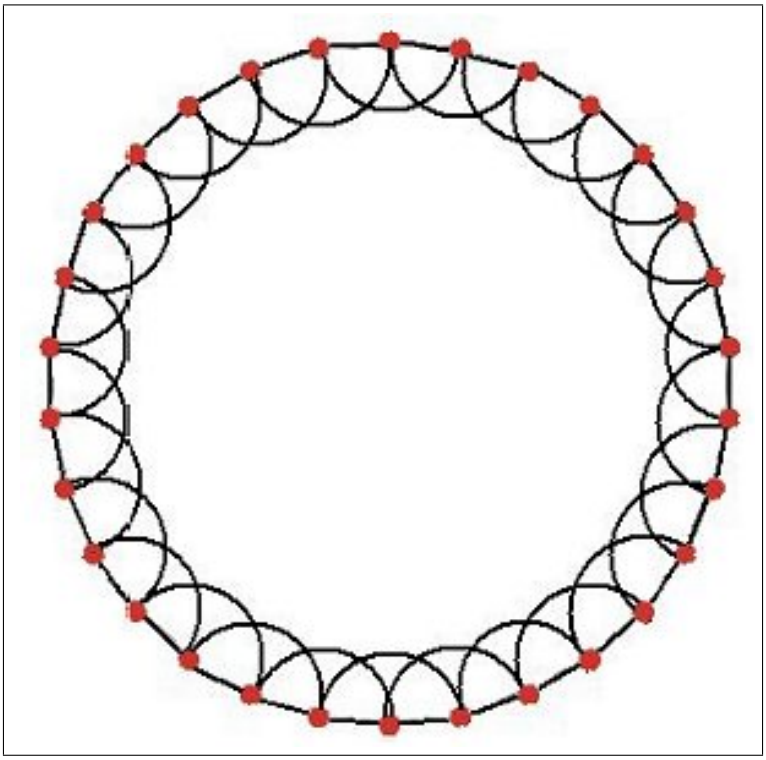

Figure 14: regular network with four degrees, Hamill and Gilbert (2009)

nodes result in an increasing connectivity. Unlike real world networks, there is low clustering in random networks. Therefore, the resulting network very rarely contains highly connected nodes, consequently, a random network is not a good candidate model for the highly connected architecture that characterize emergent patterns in nature. The clique is a highly connected random network where each node has a 100 percent probability of being connected to every other node, this means that you can get from any node to any other node in a single step because each node is connected to every other node. The model is equivalent to a group of people who all know each other. The clique diagram is useful for portraying clusters but rarely exists by itself in nature. Real world networks in nature are "sparse", meaning that they have far fewer links than what exists in a clique. In a random graph, unlike the "regular" architecture, the rule is that the node degrees have not to be all equal. Instead, the degrees are distributed according to a Poisson or normal distribution because it is assumed that any linking between nodes can happen with equal probability.

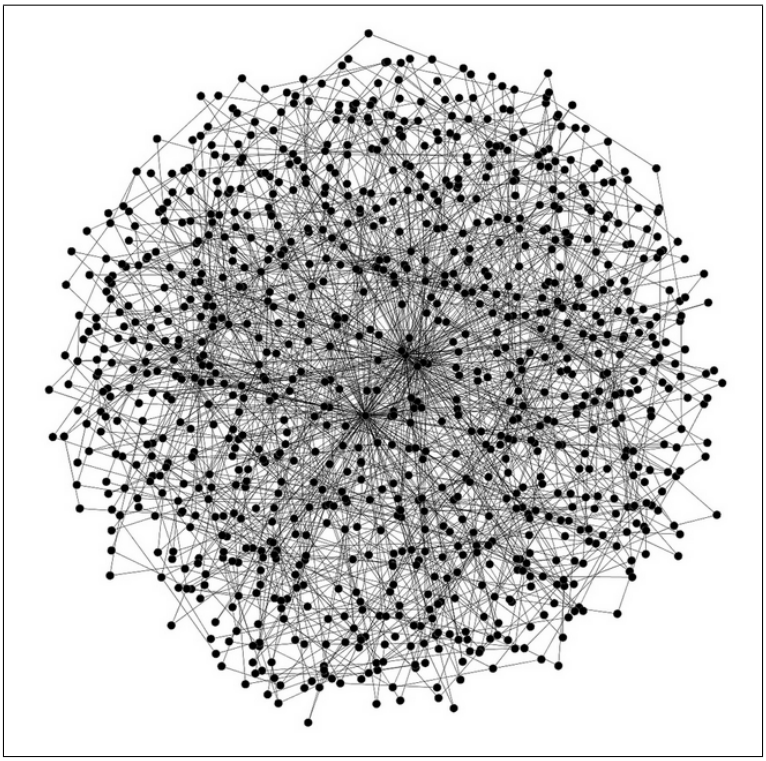

Figure 15: Random network, Patrick J. McSweeney.

Small-world AÂăsmall-world networkÂăis a network graph that can be summarized in two sentences; most nodes are not neighbors of one another, but most

nodes can be reached from every other node by a small number of links. A small world network, where nodes represent people and edges connect people who are acquainted in some way, captures the "small world" phenomenon of strangers being linked by a small number of acquaintances. One such small-world type is often called the Watts-Strogatz (WS) model. Here, only a few random links are needed to generate a short average path length which results in high connectivity between remote nodes and clusters. So short overall path lengths and high clustering are the characteristic signatures of small world networks. These characteristics result in a graph that is in between of the extremes of ordered (regular) and disordered (random). Watts and Strogatz ran a series of computer simulations where they started with a regular network. In each step of the simulation, an existing link was randomly chosen and rewired between one node and another, the network's connectivity was assessed at each step and, surprisingly, it increases dramatically with only a small amount of randomization. Thereafter, any further increase in connectivity was minimal. Early on in the simulation, the network became a "small world" and did not get significantly smaller. Watts and Strogatz showed that randomly rewiring only a few links in such a network dramatically reduced the number of links required to reach distant nodes. This dramatically increased connectivity within the network.

Preferential attachment The "scale-free" network compared to a random network has a very different kind of connectivity because the degree distribution is defined by a power law distribution instead of the Poisson distribution associated with the random network. In a power law distribution, most nodes have relatively few links but a few nodes, called hubs, have a high number of links. The contribution of the hubs to the overall connectivity is very high, while connectivity contribution of the nodes with fewer links is much lower. A famous example is the Internet in which most web pages have only a few links connecting to them, but sites like Google and Yahoo have a very large number of hyperlinks pointing to them. Scale-free networks feature link clustering around certain hubs based on preferential attachments that emerge due either to merit or legacy. Clustered small world network architectures like the Watts-Strogatz model can also be described as scale-free with the characteristic power law distribution of links. Many natural networks are well approximated by scale-free network models.

In summary there are features that the scale-free network contains that are lacking in the random network.

- In a scale free network, a small number of nodes contribute heavily to connectivity.
- In a scale free network, any two arbitrarily chosen nodes, even in a very large network, can be connected via few other intermediary nodes.
- A power law has a characteristic (constant) exponent, no matter what size the network, the dimension stays the same. Thus the term "scale-free".

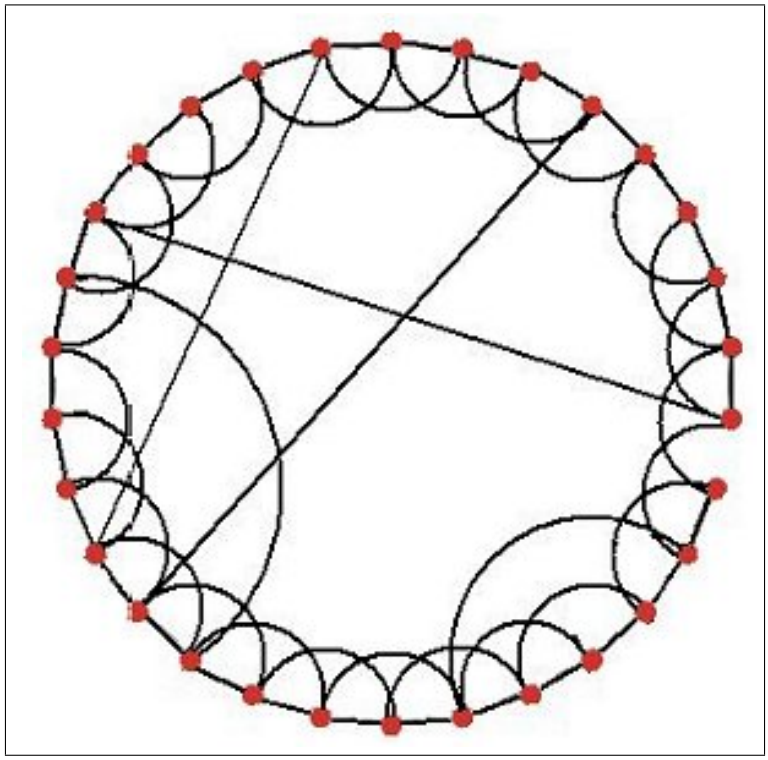

Figure 16: Small world network, Hamill and Gilbert (2009)

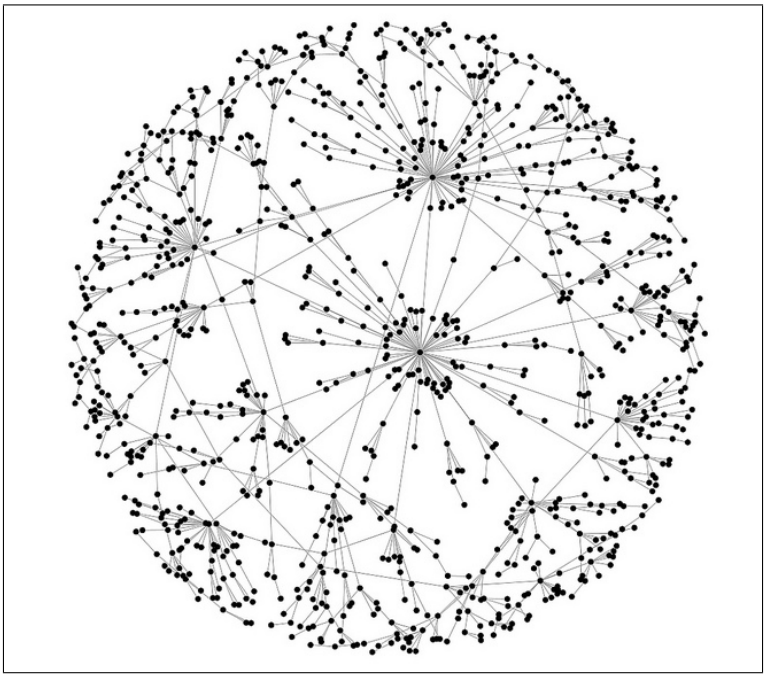

Figure 17: preferential attachment network, Jamil Alioui

- Scale-free networks are "self similar". Any part of the network is statistically similar to the whole network.
- A scale-free network is "robust". It can still operate with a random removal of a few nodes. But, the failure of a hub can fragment the system into a number of smaller islands, causing connectivity failure.
- Scale-free networks tend to promote high speed transfer of information. Their hubs have a combination of high global connectivity with highly developed local clustering. Consequently, there is a rapid synchronization of distant nodes.

3 NetLogo model

In this chapter we want to present five versions of the model, first we present the simply starting version and then the developments until the last version, the one that we use in our experiments. In each step we add new features and refine the model trying to give simple rules to the agents and simulate their behaviors.

In the version A we begin to built the network strucure thinking abount the concept of network classes, we try to implement them inside a collection of agents taking care of the different feature of the family class in terms of reciprocity. In version B we change completly how we create the network structure and we begin to work on the procedures that let the network sctructure influence the behavior of the agents. In version C the network structure is not only in the list of the agents but is showed in the interface too, this version create links that represent the social relations allowing the user to see the connections, moreover to obtain a dynamic network structure we start to develop a procedure that regulates the evolution of the network structure. We also start to differenciate the agets between each others, allowing the agents to have different weights for the network classes in the decision of retirement and increasing the complexity of the system, moreover we leave the assumption of infinitely living agents. In version D every improvement is related to the implementation of the last pension reform, in particulary we define the idea replacement rate and its influence on the model. In version E we increase the differentiation between the agents building a situation where some agents are more connected and so more important in the network structure, we change the procedure that guide the evolution of the network structure too. Finally we create procedures that help us in the study of the network structure as a whole.

3.1 Version A

We start from a simple idea of a world divided into two areas, in the light-blue one (on the right of the screen) we have the working population and in the grey one (the left) we find the retired population, at the beginning of each experiments all the agents work. Each agents receive a personal probability to retire called `myretireprob` that is independent from the network structure and the working status of all the other agents.

```
set retiredzone patches with [pxcor < 0] ask retiredzone [set pcolor 8]
set workzone patches with [pxcor >= 0] ask workzone [set pcolor 98]

create-turtles population
ask turtles [set shape "person" set color 0]
ask turtles [move-to one-of workzone set retired 0
set myretiredprob random 15 set myretiredprob myretiredprob / 100
set mynetworkretired 0 set mynetworkfamilyretired 0 ]
```

The procedure `BeSocial` we create the social network in the model, the basic idea is that each turtle has two lists, called

- `mylistfamily` and
- `mylistfriends` ,

that contains numbers and that indicates two different types of relation.

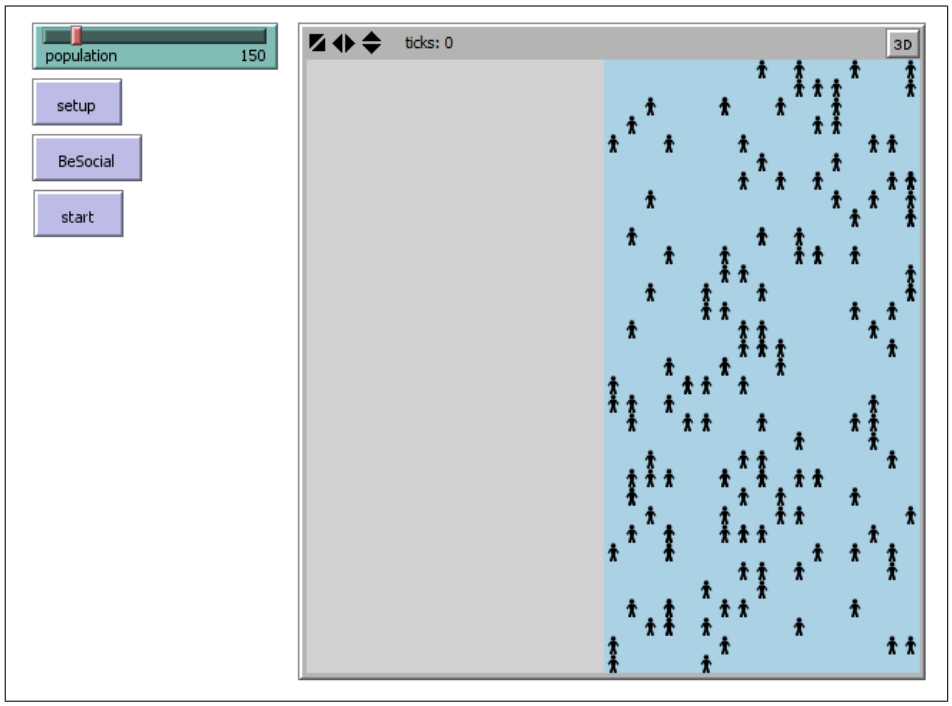

Figure 18: Interface of the model, verion A.

There are many agents as the value of a variable `population`, since every turtles is perfectly identified by his turtles-own `who` and it is a value that the turtles receive automatically while we create them, we said that a turtle is friend with the other turtles that have their numbers `who` inside the `mylistfriends` of the turtle and the same rule applies for the family group using the list `mylistfamily`.

At first we assign a random number to all agents that indicates the number of people inside their family group, then we insert many numbers as the value of this random number inside the list `mylistfamily`. As we said before these numbers that we put inside the list are random numbers, random from 0 to the value of the variable `population`, that create a relation with other turtles.

```
ask turtles [set numfamilies random 5 set mylistfamily []
```

```
set countfamily 0]
ask turtles [while [numfamilies > 0 and countfamily < numfamilies]
[set mylistfamily fput (0 + random population) mylistfamily
set countfamily countfamily + 1]
```

For the family case we have to make some necessary adjustment: each turtle remove the duplicates inside his list`mylistfamily`, they remove the number population (there isn't a turtle with turtle-own `who` equal to the value of variable population) and finally remove their own `who` number (doesn't make sense that a turtle is in his own family group).

```
set mylistfamily remove-duplicates mylistfamily
set mylistfamily remove population mylistfamily
set mylistfamily remove who mylistfamily
set numfamilies length mylistfamily ]
```

Moreover if a turtle is in the family group of another the reverse has to be true too, so we ask to each turtle to control if his number `who` appear in the others `mylistfamily` and if it is the case insert in his `mylistfamily` the new number to satisfy the reciprocity (of course if the number is already present, so the reciprocity is already true the turtle doesn't add a new number).

```
ask turtles [ set n 0 while[n <= population - 1]
[ set m who ask turtle n [set a mylistfamily] if member? m a
[  ask turtle m [ set numfamilies numfamilies + 1
set mylistfamily lput n  mylistfamily
set mylistfamily remove-duplicates mylistfamily
set numfamilies length mylistfamily]] set n n + 1 ] ]
```

The same process is applied to the friends group using the list `mylistfriends` but since we suppose that for the friend relation the reciprocity property is not needed we don't need to implement the reciprocity control. The second procedure, called `start`, count for each turtle how many agents of his family are retired and compare this group with the group composed by all the member of the list `mylistfamily`, if the ratio is bigger than a fixed exogenous value (that we decide) the influence of the family group will lead the turtle to retire, in the other case to not retire. As usual the same process is applied for the friend group of each turtles and give the influence from the friend network to the agent's decision.

3.2 Version B

Using our first version of the model we find out that if the variable `population` takes a big number, for example one thousand agents, the code needs a big amount of time to process one single tick. This is due to a part of the procedure `BeSocial` where every turtle asks to all the others to control for the reciprocity in the family relation and add or not new numbers to his `mylistfamily`. With

big numbers this doesn't fit our needs, one thousand agents generates one million controls every tick that is too heavy. So we decide to use another rule to create the social network between the agents, we change the procedure `BeSocial`.

The basic idea change completely, now we want to create a single list, called `ClassOfNetworkList`, with five numbers inside, each position of the list indicates a different class of network:

- the first position indicates the family,
- the second the friends,
- the third the colleagues,
- the fourth the bar's friends and
- the last indicates the soccer's friends.

So, speaking about family network, a turtle is in a family relation with all the other turtles with the same number in the first position, and, speaking about friends network, is in a friends relation with all the other turtles with the same number in the second position, and so on.

```
let family random 30 let pfa random 100
ifelse pfa > PNoFamily [set ClassOfNetworksList lput
(family) ClassOfNetworksList]
[set ClassOfNetworksList lput 0 ClassOfNetworksList ]

let friends random 15 let pfr random 100
ifelse pfr > PNoFriends [set ClassOfNetworksList lput
(friends) ClassOfNetworksList]
[set ClassOfNetworksList lput 0 ClassOfNetworksList]

let collegues random 20 let pco random 100
ifelse pco > PNoCollegues [set ClassOfNetworksList lput
(collegues) ClassOfNetworksList]
[set ClassOfNetworksList lput 0 ClassOfNetworksList]

let Barfriends random 25 let pbfr random 100
ifelse pbfr > PNoBarfriends [set ClassOfNetworksList lput
(Barfriends) ClassOfNetworksList]
[set ClassOfNetworksList lput 0 ClassOfNetworksList]

let Soccerfriends random 25 let psfr random 100
ifelse psfr > PNoSoccerfriends [set ClassOfNetworksList lput
(Soccerfriends) ClassOfNetworksList]
[set ClassOfNetworksList lput 0 ClassOfNetworksList]
```

Moreover the code allow for the possibility of not having all netwokclasses, for example a turtle can have family, colleagues, bar's friends relations but not have friends and soccer's friend relations. With 5 sliders we set the probability of not having each network classes, then for five times we chose a random 100 number and if it is smaller than the value in the corresponding slider the turtle takes a zero in the list, otherwise the turtle take a random number that, as we said before, indicates the other turtles with which he has a relation.

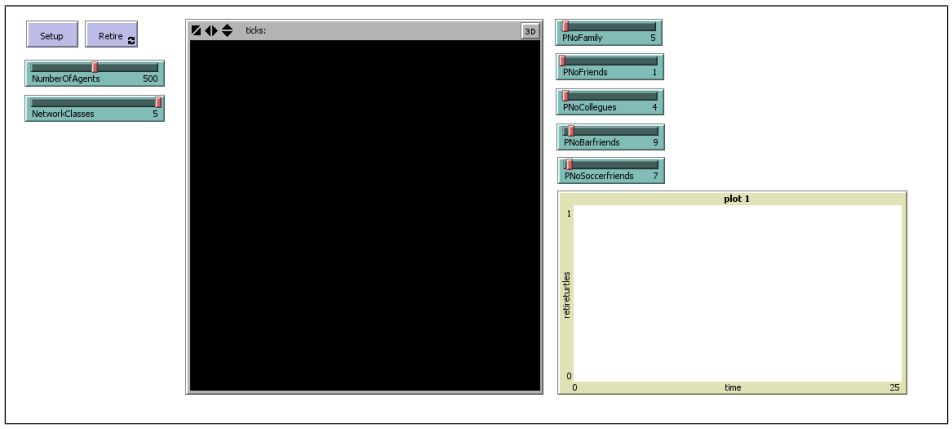

Figure 19: Interface of the model, verion B.

The procedure `BeSocial` concludes his work creating a new list for each turtles, called `MemberOfMyGroups`, that reflects the dimension of the network groups, in other words the procedure counts, from the perspective of one single agent, how many turtles of my network groups exist and stored these numbers in the new list. The length of this list is due to the value of a slider called `NetworkClasses`, that can assume integers values from one to five, the user can decide how many network classes have influence on the agents behavior, so if the value is three, the length of the list `ClassOfNetworksList` is three and only the first three network classes have a weight in the decision of the turtle.

```
ask turtles [
let poslist 0 repeat NetworkClasses [
ifelse item poslist ClassOfNetworksList > 0 [
let countmember count other turtles with
[item poslist ClassOfNetworksList =
[item poslist ClassOfNetworksList] of myself]
set MemberOfMyGroups lput (countmember) MemberOfMyGroups
set poslist poslist + 1]
[set MemberOfMyGroups lput 0 MemberOfMyGroups
set poslist poslist + 1]  ]]
```

We change the name of the second procedure, it is no longer call `Start` but now is called `Retire`, we want to count for each class how many other turtles not only have the same number in a defined position, so they are member of the same group of class of network, but also they have the variable `Retired` equal to one, means that their working life is finished. We compare the ratio obtained from this numbers and the numbers corresponding numbers from the list `MemberOfMyGroups` to a fixed quantity (1/2), if the ratio is bigger than 0.5 we put a one in a new list, called `InfoMyGroups`, in the other case we put a zero.

```
ask turtles [ repeat NetworkClasses
[ set InfOfMyGroups remove-item 0 InfOfMyGroups]
let poslist 0 repeat NetWorkClasses [
ifelse item poslist ClassOfNetworksList > 0 [
let retiredmember count other turtles with [item poslist
ClassOfNetworksList = [item poslist ClassOfNetworksList]
of myself and Retired = 1 ]

ifelse retiredmember > (1 / 2) * (item poslist MemberOfMygroups)
[set InfOfMyGroups lput 1 InfOfMyGroups ]
[set InfOfMyGroups lput 0 InfOfMyGroups]
set poslist poslist + 1 ]
[set InfOfMyGroups lput 0 InfOfMyGroups set poslist poslist + 1] ]]
```

Finally we calculate the total probability to retire of each turtle, for each class of network we take the corresponding one or zero in the list `InfoMyGroups` and multiply it for the weights of the class in the decision of the agents, for simplicity in this version we assume equal weights for all the network classes and the personal probability to retire of the individuals, so the weights are all the same and equal to the ratio of one over the number of network classes plus one. We sum all the influence from the network classes and the personal probability to retire.

We obtain a number between zero and one and compare it with a random number, always between zero and one, if the is bigger the agent doesn't retire, otherwise if is smaller or equal he retires, sets his variable `Retired` equal to one and sets his color equal to red.

```
ask turtles [let poslist 0 set NetworkProbRetire 0
repeat NetWorkClasses [ let PlusProbRetire
((item poslist InfOfMyGroups ) / (NetworkClasses + 1))
set NetworkProbRetire NetworkProbRetire + PlusProbRetire
set poslist poslist + 1 ]
set TotalProbRetire
(NetworkProbRetire + (PProbRetire / (NetworkClasses + 1)))
```

```
let retselection (random 100) / 100
if retselection <= TotalProbRetire
[set Retired 1 set color red]]
```

Last thing we change is the interface, there is no more a two areas world one with retired agents and one with working agents, now the turtles appear random in the screen.

3.3 Version C

Here in our version three we introduce some important ideas in the model in order to create a more realistic environment and develop the complexity, the interface change very much, links appear between the agents and the disposition of the turtles is ordered while before is random. We abandon the idea of infinitely living agents, modify the existing procedure like `BeSocial` and `Retire`, and create new procedures like, for example, `FindTheWeights`.

The first thing important to note is that we change again the procedure `BeSocial` allowing the agents to have more than one group of relations for each network class. We maintain the structure that relates the position in the list `ClassOfNetworkList` with the type relation between agents but instead of a list containing numbers, now the list contains other lists, in this way the first list inside the sequence contains numbers that identify the turtles with whom a family relation happens, while the list in the second position identifies the turtles for the friends relation and soo on. All the process of course is from the perspective of a single agent, so each agent has his own `ClassOfNetworkList` containing others lists containing numbers.

```
to BeSocial

if NetworkClasses = 0 [ ask turtles [ repeat 5 [NoSocial ]]    ]
if NetworkClasses = 1 [ ask turtles [ FamilySocial repeat 4 [ NoSocial ]] ]
if NetworkClasses = 2 [ ask turtles [ FamilySocial
FriendsSocial repeat 3 [ NoSocial ]]]
if NetworkClasses = 3 [ ask turtles [ FamilySocial FriendsSocial
ColleaguesSocial repeat 2 [ NoSocial ]]]
if NetworkClasses = 4 [ ask turtles [ FamilySocial FriendsSocial
ColleaguesSocial BarfriendsSocial NoSocial ]]
if NetworkClasses = 5 [ ask turtles [ FamilySocial FriendsSocial
ColleaguesSocial BarfriendsSocial SoccerfriendsSocial]]

end
```

In the code we define five short procedures, one for each network classes, called `FamilySocial`, `FriendsSocial`, `ColleaguesSocial`, `BarsFeriendsSocial`

and SoccerFriendsSocial, these procedures do exactly the same thing but each one for a different network class. What they do is simply chose a random one hundred number and if it is smaller or equal than the corresponding slider they insert a list containing a random number in the list ClassOfNetworkList of the turtle. The sliders using in the comparison are five, one for each network classes, and are called PFamily, PFriends, PColleagues, PBarfriends and PSoccerfriends while the random numbers inside the lists identify the other turtles with whom a relation happens and they are random from zero to a value decide by the user from five inputs elements in the interface, called G.family, G.Friends, G.Colleagues, G.Barfriends and G.Soccerfriends, this allow the user to decide about the avarage size of the groups for each network class.

```
to FamilySocial

let family random G.Family let pfa random 100
ifelse pfa < PFamily [ let FamilyList []
set FamilyList lput (family) FamilyList
set ClassOfNetworksList lput FamilyList  ClassOfNetworksList ]
[let FamilyList []
set ClassOfNetworksList lput FamilyList  ClassOfNetworksList ]

end

to FriendsSocial

let friends random G.Friends let pfr random 100
ifelse pfr < PFriends [ let FriendsList []
set FriendsList lput (friends) FriendsList
set ClassOfNetworksList lput FriendsList ClassOfNetworksList]
[let FriendsList []
set ClassOfNetworksList lput FriendsList ClassOfNetworksList]

end
```

Note a difference from the previous version: there the sliders indicates the probability of not having a relation, here indicates the probability of having one, we make this change because we think that the new formulation is more intuitive.

The procedure CountMemberP count ask to each turtle to count how many turtles in the various lists inside the list ClassOfNetworkList and create a new list, called MemberOfMyGroups, with this numbers inside, also the previous version does it but this time the code is different because of the fact that now the agents have in each positions a list that can contain more than one value reflecting the fact that they can be member of more than one group of relation for each network class.

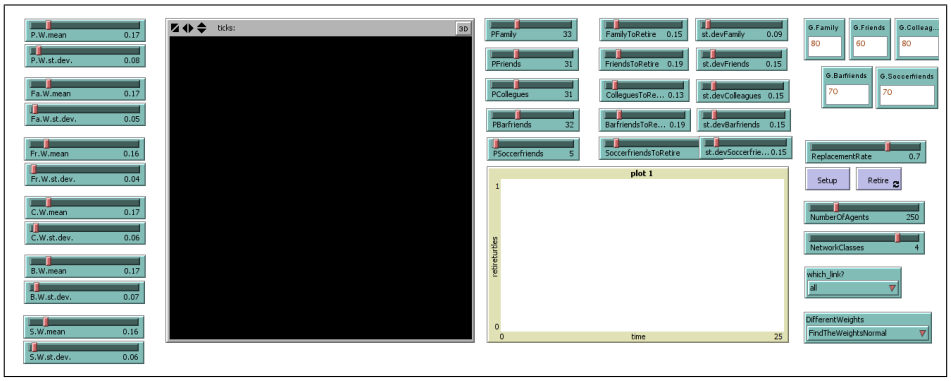

Figure 20: Interface of the model, version C.

```
to CountMemberP

ask turtles [ set MemberOfMyGroups []
let poslist 0 repeat NetworkClasses
[ ifelse empty? item poslist ClassOfNetworksList = false
[ let list1 item poslist ClassOfNetworksList
let LList length item poslist ClassOfNetworksList let countmembertot 0
repeat LList [let pos2list 0 let countmember count other turtles with
[member? item pos2list list1 item poslist ClassOfNetworksList = true  ]
set countmembertot (countmembertot + countmember)
set pos2list (pos2list + 1)  ]
set MemberOfMyGroups lput (countmembertot) MemberOfMyGroups
set poslist poslist + 1]
[set MemberOfMyGroups lput 0 MemberOfMyGroups set poslist poslist + 1]  ]]
```

Also the procedure `Retire` changes, in the previous version every turtles count with how many other agents a social relations exist and how many of these turtles are also retired, they compare the ration formed by this two numbers with a fix quantity and insert a one in the list `InfoMyGRoups` if the ratio is bigger otherwise zero if the ratio is smaller. This indicates if each network class influences the behavior of the turtles or not.

The same happens in this version with the difference that there isn't a fix quantity but every turtle faces a random normal selection from each network class, in other words the quantity is chosen from a normal distribution with mean and standard deviation select by the user in the interface from some sliders. These sliders, one for the mean and one for the standard deviation and one couple for each network class, are called `FamilyToRetire`, `st.devFamily`, `FriendsToRetire`, `st.devFriends`, `ColleguesToRetire`, `st.devColleagues`,

BarfriendsToRetire, st.devBarfriends, SoccerfriendsToRetire and st.devSoccerfriends.

```
ask turtles [
repeat NetworkClasses [ set InfOfMyGroups remove-item 0 InfOfMyGroups]
let poslist 0
repeat NetWorkClasses [ ifelse empty? item poslist ClassOfNetworksList = false
[ let list1 item poslist ClassOfNetworksList
let LList length item poslist ClassOfNetworksList
let retiredmembertot 0 let pos2list 0
repeat LList [ let retiredmember count other turtles with
[ member? item pos2list list1 item poslist ClassOfNetworksList = true
and Retired = 1]
set retiredmembertot (retiredmembertot + retiredmember)
set pos2list (pos2list + 1) ]
let ValueAboveWhich 0
if poslist = 0
[set ValueAboveWhich random-normal FamilyToRetire st.devFamily]
if poslist = 1
[set ValueAboveWhich random-normal FriendsToRetire st.devFriends ]
if poslist = 2
[set ValueAboveWhich random-normal ColleguesToRetire st.devColleagues]
if poslist = 3
[set ValueAboveWhich random-normal BarfriendsToRetire st.devBarfriends]
if poslist = 4
[set ValueAboveWhich random-normal SoccerfriendsToRetire st.devSoccerfriends]
ifelse retiredmembertot > ((ValueAboveWhich) * (item poslist MemberOfMygroups))
[set InfOfMyGroups lput 1 InfOfMyGroups ]
[set InfOfMyGroups lput 0 InfOfMyGroups]
set poslist poslist + 1 ]
[set InfOfMyGroups lput 0 InfOfMyGroups set poslist poslist + 1]]]
```

Another novelty in the procedure `Retire` is the death of the agent, in the procedure `Setup` the turtles obtain two variables, one represents the age and the other represents the age of the death of the turtle. Initially we write a code where every round of the procedure `Retire` corresponds to a tick of the model and a year in the life of the agents, in other words the variable `Age` grow by one each round, and when the value of the variable `Age` reach the same value of the variable `AgeOfDeath` the turtle die, in this view in each round some turtles die and other born.

This create a problem with the memory store of the program because of the fact that NetLogo never really forget about the die agents and round by round became slower. So we decide that when the variable `Age` reach the variable `AgeOfDeath` the turtle doesn't really die but simply change all his characteristics and his social relations with the others, in other words the total number of agents

inside the model remain always the same and the death of a turtle is the change of all his variables and lists. So in this way we have created a life-cycle for the agents, they born, each round decide if they want to retire or not until their death and reborn as a new turtle.

```
ask turtles [ifelse Age >= AgeOfDeath [
set PProbRetire (random 10) / 100 set TotalProbRetire 0
set NetworkProbRetire 0
set ClassOfNetworksList [] set MemberOfMyGroups [] set InfOfMyGroups []
set Age random-normal 70 5 set AgeOfDeath random-normal 75 5 set retired 0
set color white set Salary random-poisson 4.9375 set Salary (Salary * 4000)
if Salary < 11700 [ set Salary 11700]
let poslist 0 repeat NetWorkClasses [ set InfOfMyGroups lput 0 InfOfMyGroups
set poslist poslist + 1 ]
if NetworkClasses = 0
[repeat 5 [NoSocial ]]
if NetworkClasses = 1
[ FamilySocial repeat 4 [ NoSocial ]]
if NetworkClasses = 2
[ FamilySocial FriendsSocial repeat 3 [ NoSocial ]]
if NetworkClasses = 3
[FamilySocial FriendsSocial ColleaguesSocial repeat 2 [ NoSocial ]]
if NetworkClasses = 4
[FamilySocial FriendsSocial ColleaguesSocial BarfriendsSocial NoSocial ]
if NetworkClasses = 5
[FamilySocial FriendsSocial ColleaguesSocial
BarfriendsSocial SoccerfriendsSocial]]
[set Age (Age + 1 )    ]]
```

New procedures apper that are really important in the program, the first is the procedure `FindTheWeights` the second is `CreateLinks`.

In the previous version the weights in the decision of the agent of each network class and the personal probability to retire are fix and for simplicity they are all equal, but in this version they aren't. The user can select three different ways for the assignment of the weights, they are

- `FindTheWeightsFix`,
- `FindTheWeightsRandom` and,
- `FindTheWeightsNormal`.

If the user select `FindTheWeightsFix` he can use sliders to chose how much the personal probability to retire and the network classes influence the behavior, these sliders are called `P.W.mean`, `Fa.W.mean`, `Fr.W.mean`, `C.W.mean`, `B.W.mean`

and S.W.mean, of course the sum of this sliders can't exceed one so the program normalize the sum to one. If the user select FindTheWeightsNormal the program picks the value of the weights from normal distributions, one for each network class and one for the personal probability to retire, the means and the standard deviations for the normal distributions are selected by the user with sliders in the interface, the same ones used also in the procedure FindTheWeightsFix contain the means while the sliders called P.W.st.dev, Fa.W.st.dev, Fr.W.st.dev, C.W.st.dev, B.W.st.dev and S.W.st.dev, contain the standard deviations, of course also here the final sum of the weights is normalized to one. Finally if the user select FindTheWeightsRandom the wights are assigned by some random selections, one for each network classes and one for the personal probability to retire, make in such a way that the sum of the weights are one. In this case we suppose that the personal weights is decided before the family weight that in turn is decide before the friends weight and so on for all the network classes, in other words the random selection of the personal probability to retire is the first to happens while the family selection is the second, the friends selection is the third, the colleagues selection is the fourth, the bar's friends is the fifth and the soccer's friend selection is the last.

```
to FindTheWeightsRandom
ask turtles [
set WeightsList []

if NetworkClasses = 0 [ set WeightsList lput 1 WeightsList
repeat 5 [set WeightsList lput 0 WeightsList] ]

if NetworkClasses = 1 [ Let PersonalWeight (random-float 0.6 )
set WeightsList lput PersonalWeight WeightsList
set WeightsList lput (1 - PersonalWeight) WeightsList
repeat 4[ set WeightsList lput 0 WeightsList ] ]

if NetworkClasses = 2 [ Let PersonalWeight (random-float 0.5)
set WeightsList lput PersonalWeight WeightsList
let FamilyWeight random-float (1 - (PersonalWeight))
set WeightsList lput FamilyWeight WeightsList
set WeightsList lput (1 - (PersonalWeight) - (FamilyWeight)) WeightsList
repeat 3 [set WeightsList lput 0 WeightsList ]]

if NetworkClasses = 3 [ let PersonalWeight (random-float 0.4)
set WeightsList lput PersonalWeight WeightsList
let FamilyWeight random-float (1 - (PersonalWeight))
set WeightsList lput FamilyWeight WeightsList
let FriendsWeight random-float (1 - (PersonalWeight) - (FamilyWeight))
set WeightsList lput FriendsWeight WeightsList
set WeightsList lput (1 - (PersonalWeight) - (FamilyWeight)
```

```
- (FriendsWeight)) WeightsList
repeat 2 [set WeightsList lput 0 WeightsList]]

if NetworkClasses = 4 [ let PersonalWeight (random-float 0.3)
set WeightsList lput PersonalWeight WeightsList
let FamilyWeight random-float (1 - (PersonalWeight))
set WeightsList lput FamilyWeight WeightsList
let FriendsWeight random-float (1 - (PersonalWeight) - (FamilyWeight))
set WeightsList lput FriendsWeight WeightsList
let ColleguesWeight random-float (1 - (PersonalWeight)
- (FamilyWeight) - (FriendsWeight))
set WeightsList lput ColleguesWeight WeightsList
set WeightsList lput ( 1 - (PersonalWeight) - (FamilyWeight)
- (FriendsWeight) - (ColleguesWeight))
WeightsList set WeightsList lput 0 WeightsList]

if NetworkClasses = 5 [ let PersonalWeight (random-float 0.2)
set WeightsList lput PersonalWeight WeightsList
let FamilyWeight random (1 - (PersonalWeight))
set WeightsList lput FamilyWeight WeightsList
let FriendsWeight random-float (1 - (PersonalWeight) - (FamilyWeight))
set WeightsList lput FriendsWeight WeightsList
let ColleguesWeight random-float (1 - (PersonalWeight) - (FamilyWeight)
- (FriendsWeight))
set WeightsList lput ColleguesWeight WeightsList
let BarfriendsWeight random-float (1 - (PersonalWeight) - (FamilyWeight)
- (FriendsWeight) - (ColleguesWeight))
set WeightsList lput BarfriendsWeight WeightsList
set WeightsList lput (1 - (PersonalWeight) - (FamilyWeight)
- (FriendsWeight) - (ColleguesWeight) - (BarfriendsWeight)) WeightsList]]

end

to FindTheWeightsNormal

ask turtles [

let temporarylist [] set WeightsList []
let PersonalWeight random-normal P.W.mean P.W.st.dev.
set temporarylist lput PersonalWeight temporarylist
let FamilyWeight random-normal Fa.W.mean Fa.W.st.dev.
set temporarylist lput FamilyWeight temporarylist
let FriendsWeight random-normal Fr.W.mean Fr.W.st.dev.
set temporarylist lput FriendsWeight temporarylist
let ColleaguesWeight random-normal C.W.mean C.W.st.dev.
set temporarylist lput ColleaguesWeight temporarylist
```

```
let BarfriendsWeight random-normal B.W.mean B.W.st.dev.
set temporarylist lput BarfriendsWeight temporarylist
let SoccerWeight random-normal S.W.mean S.W.st.dev.
set temporarylist lput SoccerWeight temporarylist

let poslist 0 let sum1 0 let LList length temporarylist repeat LList
[ set sum1 (sum1 + item poslist temporarylist) set poslist (poslist + 1) ]

if sum1 = 1 [ set poslist 0 repeat LList[ let b item poslist temporarylist
set WeightsList lput b WeightsList set poslist (poslist + 1) ] ]
if sum1 > 1 [ set poslist 0 repeat LList[ let b item poslist temporarylist
set b (b / sum1) set WeightsList lput b WeightsList  set poslist (poslist + 1 ) ] ]
if sum1 < 1 [ set poslist 0 repeat LList[ let b item poslist temporarylist
set b (b / sum1) set WeightsList lput b WeightsList  set poslist (poslist + 1 ) ] ]]

end

to FindTheWeightsFix
ask turtles [
set WeightsList [] let temporarylist []
let PersonalWeight P.W.mean
set temporarylist lput PersonalWeight temporarylist
let FamilyWeight Fa.W.mean
set temporarylist lput FamilyWeight temporarylist
let FriendsWeight Fr.W.mean
set temporarylist lput FriendsWeight temporarylist
let ColleaguesWeight C.W.mean
set temporarylist lput ColleaguesWeight temporarylist
let BarfriendsWeight B.W.mean
set temporarylist lput BarfriendsWeight temporarylist
let SoccerWeight S.W.mean
set temporarylist lput SoccerWeight temporarylist

let poslist 0 let sum1 0 let LList length temporarylist
repeat LList [ set sum1 (sum1 + item poslist temporarylist) set poslist (poslist + 1) ]

if sum1 = 1 [ set poslist 0 repeat LList[ let b item poslist temporarylist
set WeightsList lput b WeightsList set poslist (poslist + 1) ] ]
if sum1 > 1 [ set poslist 0 repeat LList[ let b item poslist temporarylist
set b (b / sum1) set WeightsList lput b WeightsList  set poslist (poslist + 1 ) ] ]
if sum1 < 1 [ set poslist 0 repeat LList[ let b item poslist temporarylist
set b (b / sum1) set WeightsList lput b WeightsList  set poslist (poslist + 1 ) ] ]]

end
```

Another innovation of this version are the links, they are very important in particular for the interface. Now when a social relation happens between the agents links appear connecting turtles and we can understand which type of relation a link represent because of the color that a link has: if a link is red means that there is a family relation, blue for friends, yellow for colleagues, green for bar's friends and orange for soccer's friends. The user can decide by a chooser in the interface which links he wants to observe in the interface, the options allow the user to examine one category of links at time or all links together.

The procedure involved in the creation of the links is called `CreateLinks` but it does more than only creates links, in fact first it order in a circle the turtles by their variable who, then create links and finally use a command call layout-string to move in the space the turtles that have links showing hte network structure in the population.

```
to CreateLinks
layout-circle (sort turtles) max-pxcor - 1

if which_link? = "Family" [
ask turtles [ let list1 item 0 ClassOfNetworksList
let LList length item 0 ClassOfNetworksList let pos2list 0
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 0 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ]]
ask links [ set typelink 1 set color red]]

if which_link? = "Friends" [
ask turtles [ let list1 item 1 ClassOfNetworksList
let LList length item 1 ClassOfNetworksList let pos2list 0
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 1 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ] ]
ask links [ set typelink 2 set color blue]]

if which_link? = "Collegues" [
ask turtles [ let list1 item 2 ClassOfNetworksList
let LList length item 2 ClassOfNetworksList let pos2list 0
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 2 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ] ]
ask links [ set typelink 3 set color yellow]]

if which_link? = "Barfriends" [
ask turtles [ let list1 item 3 ClassOfNetworksList
let LList length item 3 ClassOfNetworksList let pos2list 0
```

```
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 3 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ] ]
ask links [ set typelink 4 set color green]]

if which_link? = "Soccerfriends" [
ask turtles [ let list1 item 4 ClassOfNetworksList
let LList length item 4 ClassOfNetworksList let pos2list 0
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 4 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ]]
ask links [ set typelink 5 set color orange]]

if which_link? = "all" [
ask turtles [ let list1 item 0 ClassOfNetworksList
let LList length item 0 ClassOfNetworksList let pos2list 0
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 0 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ] ]
ask links with [typelink < 1] [ set typelink 1 set color red]
ask turtles [ let list1 item 1 ClassOfNetworksList
let LList length item 1 ClassOfNetworksList let pos2list 0
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 1 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ]]
ask links with [typelink < 1][ set typelink 2 set color blue]
ask turtles [ let list1 item 2 ClassOfNetworksList
let LList length item 2 ClassOfNetworksList let pos2list 0
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 2 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ] ]
ask links with [typelink < 1] [ set typelink 3 set color yellow]
ask turtles [ let list1 item 3 ClassOfNetworksList
let LList length item 3 ClassOfNetworksList let pos2list 0
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 3 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ] ]
ask links with [typelink < 1][ set typelink 4 set color green]
ask turtles [ let list1 item 4 ClassOfNetworksList
let LList length item 4 ClassOfNetworksList let pos2list 0
repeat LList [  create-links-with other turtles with
[ member? item pos2list list1 item 4 ClassOfNetworksList = true ]
set pos2list (pos2list + 1) ]]
ask links with [typelink < 1][ set typelink 5 set color orange]]

repeat 50 [layout-spring (turtles with [any? link-neighbors])
```

```
links 0.4 7 1 display ]

end
```

The last procedure in the code is called `Modify/AddLinks`, the presence of this procedure is justify by the fact that during their life people may change their social groups or add new groups, we need to transfer this property to our model. At first the turtles receive a new variable called `SocialStrenght` from a normal distribution, this variable is normalized to one and the agents are divided in three groups: the first is the group of agents with the variable `SocialStrenght` bigger than 0.7, the second group has the variable between 0.4 and 0.7, while the last group is composed by the ones that have the variable `SocialStrenght` less than 0.4 .

Now, what happens is similar to the procedure `BeSocial` but is different for each groups. The first group select some random numbers from the input elements in the interface (one for each network class using in the experiment except for the family) and some corresponding random one hundred numbers, for each network class if the one hundred random number is less or equal than the corresponding slider (`Pfriends` for example) the turtle add the random number from the input element to one of his lists in the list `ClassOfNetworkList`. In other words the first group may add new groups of relation in their social network classes. The second group likewise may change groups in their social network classes, so in some sense the turtles don't became more social but only modify their groups. The third group instead may delete elements from the lists in the list `ClassOfNetworkList` and may became less social. Note that the family relation is not involved in this procedure because of the fact that you can't create, change or delete family network structure.

```
to Modify/AddLinks

let maxSS 0
ask turtles [ set SocialStrenght random-normal 8 2
if SocialStrenght > maxSS [set maxSS SocialStrenght]  ]

ask turtles [ set SocialStrenght (SocialStrenght / maxSS) ]

ask turtles with [SocialStrenght >= 0.7 ][
let friends random G.Friends let pfr random 100
let colleagues random G.Colleagues let pco random 100
let barfriends random G.Barfriends let pbfr random 100
let soccerfriends random G.Soccerfriends let psfr random 100

let FriendsList item 1 ClassOfNetworksList
```

```
let ColleaguesList item 2 ClassOfNetworksList
let BarfriendsList item 3 ClassOfNetworksList
let SoccerfriendsList item 4 ClassOfNetworksList

if NetworkClasses = 2 [ if pfr < PFriends
[set FriendsList lput friends FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]]

if NetworkClasses = 3 [ if pfr < PFriends
[set FriendsList lput friends  FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]
if pco < PCollegues [set ColleaguesList lput colleagues ColleaguesList
set ColleaguesList remove-duplicates ColleaguesList
set ClassOfNetworksList replace-item 2 ClassofNetworksLIst ColleaguesList]]

if NetworkClasses = 4 [ if pfr < PFriends
[set FriendsList lput friends  FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]
if pco < PCollegues [set ColleaguesList lput colleagues ColleaguesList
set ColleaguesList remove-duplicates ColleaguesList
set ClassOfNetworksList replace-item 2 ClassofNetworksLIst ColleaguesList]
if pbfr < PBarfriends [set BarfriendsList lput barfriends BarfriendsList
set BarfriendsList remove-duplicates BarfriendsList
set ClassOfNetworksList replace-item 3 ClassofNetworksLIst BarfriendsList]]

if NetworkClasses = 5 [ if pfr < PFriends
[set FriendsList lput friends  FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]
if pco < PCollegues [set ColleaguesList lput colleagues ColleaguesList
set ColleaguesList remove-duplicates ColleaguesList
set ClassOfNetworksList replace-item 2 ClassofNetworksLIst ColleaguesList]
if pbfr < PBarfriends [set BarfriendsList lput barfriends BarfriendsList
set BarfriendsList remove-duplicates BarfriendsList
set ClassOfNetworksList replace-item 3 ClassofNetworksLIst BarfriendsList]
if psfr < PSoccerfriends [set SoccerfriendsList lput soccerfriends SoccerfriendsList
set SoccerfriendsList remove-duplicates SoccerfriendsList
set ClassOfNetworksList replace-item 4 ClassofNetworksLIst SoccerfriendsList ]]]

ask turtles with [SocialStrenght >= 0.4 and SocialStrenght < 0.7 ][

let friends random G.Friends let pfr random 100
let colleagues random G.Colleagues let pco random 100
```

```
let barfriends random G.Barfriends let pbfr random 100
let soccerfriends random G.Soccerfriends let psfr random 100

let FriendsList item 1 ClassOfNetworksList
let ColleaguesList item 2 ClassOfNetworksList
let BarfriendsList item 3 ClassOfNetworksList
let SoccerfriendsList item 4 ClassOfNetworksList

if NetworkClasses = 2 [ if empty? FriendsList = false and  pfr < PFriends
[set FriendsList remove-item 0 FriendsList
set FriendsList lput friends FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList] ]

if NetworkClasses = 3 [ if empty? FriendsList = false and  pfr < PFriends
[ set FriendsList remove-item 0 FriendsList
set FriendsList lput friends FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]
if empty? ColleaguesList = false and pco < PCollegues
[set ColleaguesList remove-item 0 ColleaguesList
set ColleaguesList lput colleagues ColleaguesList
set ColleaguesList remove-duplicates ColleaguesList
set ClassOfNetworksList replace-item 2 ClassofNetworksLIst ColleaguesList]]

if  NetworkClasses = 4 [ if empty? FriendsList = false and  pfr < PFriends
[set FriendsList remove-item 0 FriendsList
set FriendsList lput friends FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]
if empty? ColleaguesList = false and pco < PCollegues
[set ColleaguesList remove-item 0 ColleaguesList
set ColleaguesList lput colleagues ColleaguesList
set ColleaguesList remove-duplicates ColleaguesList
set ClassOfNetworksList replace-item 2 ClassofNetworksLIst ColleaguesList]
if empty? BarfriendsList = false and pbfr < PBarfriends
[set BarfriendsList remove-item 0 BarfriendsList
set BarfriendsList lput barfriends BarfriendsList
set BarfriendsList remove-duplicates BarfriendsList
set ClassOfNetworksList replace-item 3 ClassofNetworksLIst BarfriendsList]]

if  NetworkClasses = 5 [ if empty? FriendsList = false and  pfr < PFriends
[set FriendsList remove-item 0 FriendsList  set FriendsList lput friends FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]
if empty? ColleaguesList = false and pco < PCollegues
```

```
[set ColleaguesList remove-item 0 ColleaguesList
set ColleaguesList lput colleagues ColleaguesList
set ColleaguesList remove-duplicates ColleaguesList
set ClassOfNetworksList replace-item 2 ClassofNetworksLIst ColleaguesList]
if empty? BarfriendsList = false and pbfr < PBarfriends
[set BarfriendsList remove-item 0 BarfriendsList
set BarfriendsList lput barfriends BarfriendsList
set BarfriendsList remove-duplicates BarfriendsList
set ClassOfNetworksList replace-item 3 ClassofNetworksLIst BarfriendsList]
if empty? SoccerfriendsList = false and psfr < PSoccerfriends
[set SoccerfriendsList remove-item 0 SoccerfriendsList
set SoccerfriendsList lput soccerfriends SoccerfriendsList
set SoccerfriendsList remove-duplicates SoccerfriendsList
set ClassOfNetworksList replace-item 4 ClassofNetworksLIst SoccerfriendsList]]]


ask turtles with [SocialStrenght < 0.4 ][
let friends random G.Friends let pfr random 100
let colleagues random G.Colleagues let pco random 100
let barfriends random G.Barfriends let pbfr random 100
let soccerfriends random G.Soccerfriends let psfr random 100

let FriendsList item 1 ClassOfNetworksList
let ColleaguesList item 2 ClassOfNetworksList
let BarfriendsList item 3 ClassOfNetworksList
let SoccerfriendsList item 4 ClassOfNetworksList

if NetworkClasses = 2 [ if empty? FriendsList = false and  pfr < PFriends
[set FriendsList remove-item 0 FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList] ]

if NetworkClasses = 3 [ if empty? FriendsList = false and  pfr < PFriends
[ set FriendsList remove-item 0 FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]
if empty? ColleaguesList = false and pco < PCollegues
[set ColleaguesList remove-item 0 ColleaguesList
set ClassOfNetworksList replace-item 2 ClassofNetworksLIst ColleaguesList]]

if  NetworkClasses = 4 [ if empty? FriendsList = false and  pfr < PFriends
[set FriendsList remove-item 0 FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]
if empty? ColleaguesList = false and pco < PCollegues
[set ColleaguesList remove-item 0 ColleaguesList
set ClassOfNetworksList replace-item 2 ClassofNetworksLIst ColleaguesList]
if empty? BarfriendsList = false and pbfr < PBarfriends
[set BarfriendsList remove-item 0 BarfriendsList
```

```
set ClassOfNetworksList replace-item 3 ClassofNetworksLIst BarfriendsList]]

if  NetworkClasses = 5 [ if empty? FriendsList = false and  pfr < PFriends
[set FriendsList remove-item 0 FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList]
if empty? ColleaguesList = false and pco < PCollegues
[set ColleaguesList remove-item 0 ColleaguesList
set ClassOfNetworksList replace-item 2 ClassofNetworksLIst ColleaguesList]
if empty? BarfriendsList = false and pbfr < PBarfriends
[set BarfriendsList remove-item 0 BarfriendsList
set ClassOfNetworksList replace-item 3 ClassofNetworksLIst BarfriendsList]
if empty? SoccerfriendsList = false and psfr < PSoccerfriends
[set SoccerfriendsList remove-item 0 SoccerfriendsList
set ClassOfNetworksList replace-item 4 ClassofNetworksLIst SoccerfriendsList]]]

end
```

3.4 Version D

I this fourth version we try to implement in the model some new specifications in order to obtain an estimation of the resources needed to comply with the pension system and implement a semplify version of the newest pension reform by minister Fornero.

In order to obtain such estimation we need agents endowed with a new variable called `Salary` that indicates the money earned by the agents in a year of work, for the moment we assume fully employment in the society so we don't take in consideration the unemployment possibility.

The distribution of `Salary` is a Poisson distribution, we consider that it is the best approximation for this type of variable reflectiong the fact that there are few people with hight salary and more people with low salary in the society, at least in the italian case this is the situation. In the code we set a random-poisson with value four in his paramenter and then multiply the value obtained for 4225, we do so to obtain an average value of the variable Salary of 16900, we cannot put directly 16900 as a mean because NetLogo require only one parameter using the command `random-poisson` and that parameter combines the concept of mean and of degrees of freedom, using directly 16900 the Poisson distribution approximates to a Normal distribution.

We insert also a slider called `ReplacementRate` that indicates how much the Agents receive of their working salary when they decide to retire. Every agent has a variable called `PReplacementRate` that indicates his personal replacement rate, when a turtle borns this variable simply assumes the value of the slider `ReplacementRate` but when the agents pass the age of 66 years his `PReplacementRate` grow every year if the turtle is not retire yet. This is made to simulate the real newest pension reform that contemplates the fact that if you are 66 you can retire or you can choose to continue to work with benefits

in your future pension, you can postpone your retirement until the age of 70.

In each tick of the model we compute the money that the pension system need, in the code we ask to all turtles with the variable `Retire` equal to one to multiply their variable `Salary` with their `PReplacementRate`, finally we sum all together this values to obtain the total resources needed.

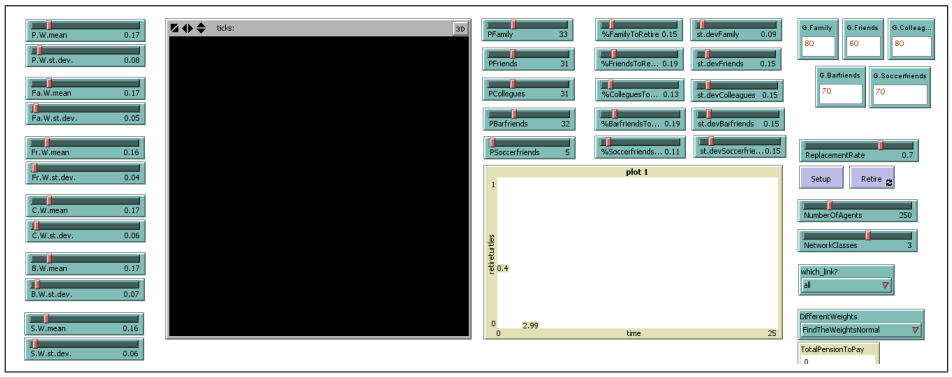

Figure 21: Interface of the model, version D.

Our second innovation in this version of the model is the implementation of a simplify version of the newest pension reform. We consider the conditions that will apply from 2018 and not the transition steps from years 2011 to, of course, 2018.

The conditions described in the reform indicate that people can retire if they are 66 and they have at least 20 years of contribution, but they can choose to not retire and obtain pension benefits for their future pensions until the age of 70. In this framework we do some assumptions, we assume that nobody will work after the age of 70, because there are no benefit to do it, we assume that an hight`"PReplacementRate"`induce the agents to retire, this means that the `PProbRetire` grow, because the benefits of working is little and pension is consistent with their usual standard of living, otherwise a low `PReplacementRate` induce the agents to not retire, and this means that the `PProbRetire` decrease. We introduce this ideas with three new procedures: `PensionCheck`, `ReplaceToRetire` and `LongWorkersBenefits`.

We create a procedure called `PensionCheck` that is recall by the procedure `Retire`, as in the previous version in the procedure `Retire` each agent get the information from their network classes and compute his personal `PProbRetire`, then decides if retire or not with a random selection. Here comes the`PensionCheck procedures`, it verifies if the turtles is not retired and if satisfies the years of contribution, if it is the case the procedure check the variable `Age`, if is bigger or equal to 66 the agent retire, if it is less than 66 the agent doesn't retire, becuase he can't, but set the variable `IWillRetire` equal to one, so he will retire for sure at the age of 66 and doesn't take in consideration the idea of working more from 66 to 70. The intuition is that if an agent wants to retire at age, for example, of 64

but he can't do it, he will do it at the very first moment possible.

```
to PensionCheck

if Retired = 0 and YearsContribution >= 20 [ if Age >= 66
[ set Retired 1 set color red set AgeOfRetire Age ]
if Age < 66  [ set IWillRetire 1 ]   ]

end
```

The procedure `ReplaceToRetire` contains the idea described before of the influence on the `PProbRetire` of the `PReplacementRate`, if the `PReplacementRate` is hight the agent is induced to retire otherwise is induced to work.

```
to ReplaceToRetire

if PReplacementRate <= 1 and PReplacementRate >= 0.9
[ set PProbRetire 1 ]

if PReplacementRate < 0.9 and PReplacementRate >= 0.7
[ set PProbRetire (PProbRetire + random-float 0.2) ]

if PReplacementRate < 0.7 and PReplacementRate >= 0.5
[ set PProbRetire PProbRetire ]

if PReplacementRate < 0.5 and PReplacementRate >= 0.3
[ set PProbRetire (PProbRetire - random-float 0.1) ]

if PReplacementRate < 0.3 and PReplacementRate >= 0
[ set PProbRetire (PProbRetire - random-float 0.2) ]

end
```

Finally the procedure `LongWorkerBenefits` reflect the idea of earning benefits for agents with Age more or equal to 66 that are not retired and continued to work, each year their `PReplacementRate` grow, this growing in the `PReplacementRate` influences their decisions as usual, hight `PReplacementRate` induce to retire, low `PReplacementRate` induce to work.

```
to LongWorkersBenefits

if Age >= 66 and Retired = 0
```

```
[ set PReplacementRate (PReplacementRate + 0.05)

if PReplacementRate <= 1 and PReplacementRate >= 0.9
[ set PProbRetire 1 ]

if PReplacementRate < 0.9 and PReplacementRate >= 0.7
[ set PProbRetire (PProbRetire + random-float 0.2) ]

if PReplacementRate < 0.7 and PReplacementRate >= 0.5
[ set PProbRetire PProbRetire ]

if PReplacementRate < 0.5 and PReplacementRate >= 0.3
[ set PProbRetire (PProbRetire - random-float 0.1) ]

if PReplacementRate < 0.3 and PReplacementRate >= 0
[ set PProbRetire (PProbRetire - random-float 0.2) ] ]

end
```

Another addition in this version is that the `PProbRetire` of the agents grow also by the simply fact that they became older, this is another assumption, maybe strong but reasonable, we think that the fact of becoming older generates a growth in the `PProbRetire` of the agents, but how much? the growth is generated by a random float of 0.1 .

Finally we also add a new graph, called plot 2, that show the average age of retirement, each turtle has a variable called `AgeOfRetire` that assume the value of the variable `Age` when, in fact, he retires. we sum all together the variables `AgeOfRetire` of the retired turtles and divide the sum by the number of the these turtles.

3.5 Version E

The Last development of our model is centered on the network subject, we do two different improvements:

- we refine some old procedures (`BeSocial` and `Modify/AddLinks`) to enrich the network structure,
- we create new procedure and elements that help us to study the network structure, otherwise we could not say much on it.

Finally we modify the procedures `Retire`, `longWorkersBenefit` and `ReplaceToRetire`.

The meaning of the procedure `BeSocial` remain the same, it is still composed by five subprocedures, one for each network class, that have the role to insert or not numbers into the lists of the list `ClassOfNetworksList` and so create the indications for the network structure but here in this version appear the variable

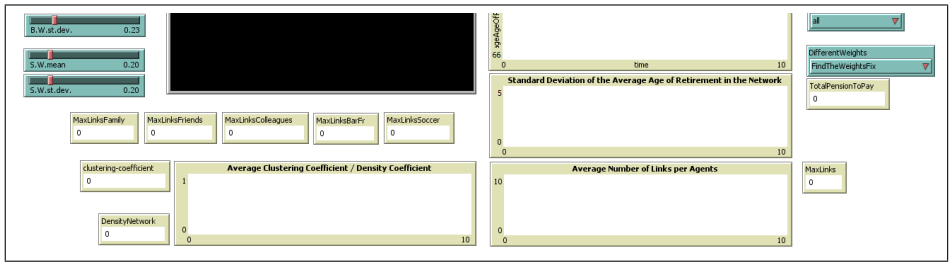

Figure 22: Interface of the model, version E.

`SocialStrenght`. The intuition is simple, for high values of this variable a turtle do the process many times (for each network class), in particular if the variable is more than 0.8 the turtle do other two additional rounds, while if the values is between 0.6 and 0.8 only one additional round. This not happens for the family class and the reason is always the uniqueness of the family relation.

```
to FriendsSocial

let friends random G.Friends let pfr random 100 let pfr2 0

ifelse pfr < PFriends [ let FriendsList []
set FriendsList lput (friends) FriendsList
set ClassOfNetworksList lput FriendsList ClassOfNetworksList]
[let FriendsList [] set ClassOfNetworksList lput FriendsList ClassOfNetworksList]

if SocialStrenght >= 0.8 [ repeat 2
[let FriendsList item 1 ClassOfNetworksList
set pfr2 random 100 let friends2 random G.Friends if pfr2 > PFriends
[  set FriendsList lput (friends2) FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList  ] ] ]

if SocialStrenght >= 0.6 and Socialstrenght < 0.8
[let FriendsList item 1 ClassOfNetworksList
set pfr2 random 100 let friends2 random G.Friends if pfr2 > PFriends
[  set FriendsList lput (friends2) FriendsList
set FriendsList remove-duplicates FriendsList
set ClassOfNetworksList replace-item 1 ClassofNetworksLIst FriendsList  ] ]

end
```

We also modify the procedure `Modify/AddLinks`, we do not change how the code operates but while before there are three categories of behavior of the turtles related to the value of the variable `SocialStrenght`, now there are four categories. If agents have the variable `SocialStrenght` greater or equal than 0.8 they do two round where they could add a new number to their list, and so enlarge their network, if the variable is between 0.8 and 0.6 they do only one round of potential addition, if it is between 0.6 and 0.4 they eliminate one number and add a new one, so in fact they do not enlarge the network but only change the composition, if is between 0.4 and 0.2 the turtles do one round where they could eliminate one number from their lists, and so restrict their social network, and finally if the variable `SocialStrenght` is less than 0.2 they do two rounds of potential elimination of social connections. So this part of the code represent the evolution of the social network of living turtles over time.

Now we want to define two new procedures very important in order to judge the social red of the agents:

- `find-clustering-coefficient`
- `DensityRatio`

We have explained this two measures in the first chapter, now we explain how we get them from Netlogo. To get the `DensityRatio` we simply take the ratio of the existing links over the theorically possible maximum number of links. Instead for the clustering coefficient we use a code with a procedure `find-clustering-coefficient` present in one example of the network unit of the library in NetLogo, Wilensky 2005, called Small-World; detailed clarification on the code are available in the website of NetLogo where the model Small-World is described.

```
to DensityRatio

let a count links
let b ( NumberOfAgents * (NumberOfAgents - 1) * NetworkClasses)
set DensityNetwork ( a / b )

end

to find-clustering-coefficient

ifelse all? turtles [count link-neighbors <= 1]
[set clustering-coefficient 0]

[let total 0
ask turtles with [ count link-neighbors <= 1]
[ set node-clustering-coefficient "undefined" ]
ask turtles with [ count link-neighbors > 1]
```

```
[let hood link-neighbors
set node-clustering-coefficient (2 * count links with [ in-neighborhood? hood ] /
((count hood) * (count hood - 1)) )
set total total + node-clustering-coefficient]

set clustering-coefficient total / count turtles with [count link-neighbors > 1]]

end

to-report in-neighborhood? [ hood ]

report ( member? end1 hood and member? end2 hood )

end
```

Moreover we deifine another new procedure that generate a plot and six monitors in the interface, this procedure is called `DofC`. This part of the code calculate for each turn of the model the maximum number of links in a sinlge node, the average number of links per node, and the maximum number of links in a single node for each category of links (that are related to each network classes). From this datas we can understand better the degree of connectivity of the nodes in the network, the average numbers of links is showed in a plot, called "Average Number of Links per Agents", while the others measures appear in six monitors called: `MaxLinksFamily`, `MaxLinksfriends`, `MaxLinksColleagues`, `MaxLinksBarFr`, and `MaxLinksSoccer`.

```
to  DofC

set MaxLinks 0
ask turtles [ set HmLinks count my-links
if Hmlinks > MaxLinks [ set Maxlinks HmLinks ] ]
set AvLinks mean [ HmLinks ] of turtles

set MaxLinksFamily 0

ask turtles with [empty? item 0 ClassOfNetworksList  = false ]
[ set HmFaLinks count my-links with [ typelink = 1 ]
if HmFalinks > MaxLinksFamily [ set MaxlinksFamily HmFaLinks ] ]
let AvFaLinks mean [ HmFaLinks ] of turtles

set MaxLinksFriends 0

ask turtles with [ empty? item 1 ClassOfNetworksList  = false ]
[ set HmFrLinks count my-links with [typelink = 2 ]
if HmFrlinks > MaxLinksFriends [ set MaxlinksFriends HmFrLinks ] ]
```

```
let AvFrLinks mean [ HmFrLinks ] of turtles

set MaxLinksColleagues 0

ask turtles with [  empty? item 2 ClassOfNetworksList = false ]
[ set HmCoLinks count my-links with [typelink = 3 ]
if HmColinks > MaxLinksColleagues [ set MaxlinksColleagues HmCoLinks ] ]
let AvCoLinks mean [ HmCoLinks ] of turtles

set MaxLinksBarFr 0

ask turtles with [ empty? item 3 ClassOfNetworksList = false ]
[ set HmBaLinks count my-links with [typelink = 4]
if HmBalinks > MaxLinksBarFr [ set MaxlinksBarFr HmBaLinks ] ]
let AvBaLinks mean [ HmBaLinks ] of turtles

set MaxLinksSoccer 0

ask turtles with [ empty? item 4 ClassOfNetworksList  = false ]
[ set HmSoLinks count my-links with [typelink = 5]
if HmSolinks > MaxLinksSoccer [ set MaxlinksSoccer HmSoLinks ] ]
let AvSoLinks mean [ HmSoLinks ] of turtles

end
```

In the interface we add also three new plots and two new monitors, they report to the user some informations:

- the average age of the agents when they retire,
- the standard deviation of the age when they retired,
- the average clustering coefficient and average density coefficient,
- the precise average clustering coefficient of the turn and the precise average density coefficient of the turn.

As we have said before we modify the procedure `Retire`, we do this in order to increase the complexity of the system. In the previous version the list `InfOfMyGroups` collect the influences of the five networks classes on the behavior of the agent but the possibility are only two: if there is a zero in the list the class influences the agent to not retire, if there is a 1 in the list the class influences the agent to retire. We decide to modify this situation for two classes, friends and colleagues, in order to avoid this on/off situation. So now family, bar's friends and soccer's friends influence the decision of the agent as usual but with friends and colleagues the situation change:

- in the class Friends everything depend on the variable `SocialStrengh`, if it is bigger or equal than 0.66 the code insert a random 0.4 number in the list `InfOfMyGroups`, if it is between 0.66 and 0.33 insert a random 0.4 number plus 0.3 and if it is less than 0.33 insert a random 0.4 number plus 0.6 . The intuition is that if the agent is very "social" he do not care much about the opinions of his friends because he probably changes or adds soon new friends to his network and so is, in some sense, free to decide because the relations are not bounding. But if the variable `SocialStrenght` has a low value this means that the relations with his friends is durable (it probably will not change over time), and so the influence of these relations became more important and this is reflect by the number in the list `InfOfMyGroups`.
- In the class Colleagues the intuition is similar but everything is related to the variable `StableWork`. If this variable assumes value 1 this means that the work relation is stable and the social relation is durable, so the list `InfOfMyGroups` receives a random 0.5 number plus 0.5 while if it assumes value 0 means that the work relation is temporary and the social relation is not durable, in this case the list `InfOfMyGroups` receives only a random 0.5 number. This modification push up the heterogeneity of the agents and enlarge the complexity, building in our view a more realistic and less predictable simulation.

```
ifelse retiredmembertot > ((ValueAboveWhich) * (item poslist MemberOfMygroups))

[if poslist = 0  [ set InfOfMyGroups lput 1 InfOfMyGroups ]

if poslist = 1  [ if SocialStrenght >= 0.66
[set InfOfMyGroups lput (random-float 0.4) InfOfMyGroups]
if SocialStrenght >= 0.33 and SocialStrenght < 0.66
[set InfOfMyGroups lput (random-float 0.4 + 0.3 ) InfOfMyGroups]
if SocialStrenght < 0.33
[set InfOfMyGroups lput (random-float 0.4 + 0.6) InfOfMyGroups]]

if poslist = 2  [ ifelse StableWork = 1
[ set InfOfMyGroups lput ( random-float 0.5 + 0.5) InfOfMyGroups ]
[ set InfOfMyGroups lput (random 0.5) InfOfMyGroups ]]

if poslist = 3  [set InfOfMyGroups lput 1 InfOfMyGroups]

if poslist = 4  [set InfOfMyGroups lput 1 InfOfMyGroups]]

[set InfOfMyGroups lput 0 InfOfMyGroups]

set poslist poslist + 1 ]

[set InfOfMyGroups lput 0 InfOfMyGroups set poslist poslist + 1] ]]
```

We modify the procedure `Retire` also from another point of view, in our experiments should be usefull to have the possibility to run the code with 0 network classes, so in other words without the network structure, in orther to compare the behavior with and without social influence, and with the previous version of the code this is not possible. So we change the situation and we write a code that taking in consideration two possible options at the beginning:

- if the slider `NetworkClasses` takes a value greater than 0, the model create the network structure and the agents are influenced by the network classes,
- if the slider `NetworkClasses` takes the value 0, the agents do not perceive the network classes and take their decision using only their `PProbRetire` to decide when retire or not.

Finally we modify the procedures `LongWorkersBenefits` and `ReplaceToRetire`. We eliminate the part of the procedure `LongWorkersBenefits` that let the `PReplacementRate` influence the `PProbRetire` because that part is recall in the procedure `ReplaceToRetire`. In the previous version this part of the code is presented in both procedures but this is no more than a repetition that creates a double not reasonable effect, so we delete it. Moreover now the procedure not only influence the `PProbRetire` but also could modify the influence of the Family class on the retirement decision. Everything depends on the variable `PReplacementRate` ,

- if it is bigger or equal than 0.9 the turtle set `PProbRetire` equal to 1 and if the influence of the Family in the list `InfOfMyGroups` is 0 set the influence to 0.5,
- if it is between 0.9 and 0.7 it increases the `PProbRetire` by a random 0.2 number and if the Family influence is 0 set the influence to 0.3,
- if it is between 0.7 and 0.5 nothing happens,
- if it is between 0.5 and 0.3 the procedure decrease the `PProbRetire` by a random 0.2 number and if the Family influence is 1 set the influence to 0.5,
- finally if it is between 0 and 0.3 the procedure decrease the `PProbRetire` to 0 and if the Family influence is 1 set the influence to 0.3 .

The intuition is that in the case of high `PReplacementRate` the agents have the incentives to retire and the Family support the decision of retirement, while in the case of low `PReplacementRate` the agents have the incentive to not retire and the Family obstruct retirement.

```
to  LongWorkersBenefits

if Age >= 66 and Retired = 0
[ set PReplacementRate (PReplacementRate + 0.05)]
```

```
end


to ReplaceToRetire

if PReplacementRate <= 1 and PReplacementRate >= 0.9
[ set PProbRetire 1 let z item 0 InfOfMyGroups if z = 0
[ set InfOfMyGroups replace-item 0 InfOfMyGroups 0.5 ]]

if PReplacementRate < 0.9 and PReplacementRate >= 0.7
[ set PProbRetire (PProbRetire + random-float 0.2)
let z item 0 InfOfMyGroups if z = 0
[ set InfOfMyGroups replace-item 0 InfOfMyGroups 0.3 ]]

if PReplacementRate < 0.7 and PReplacementRate >= 0.5
[ set PProbRetire PProbRetire ]

if PReplacementRate < 0.5 and PReplacementRate >= 0.3
[ set PProbRetire (PProbRetire - random-float 0.2)
if PProbRetire < 0
[set PProbRetire 0]
let z item 0 InfOfMyGroups if z = 1
[ set InfOfMyGroups replace-item 0 InfOfMyGroups 0.5 ]]

if PReplacementRate < 0.3 and PReplacementRate >= 0
[ set PProbRetire 0 let z item 0 InfOfMyGroups if z = 1
[ set InfOfMyGroups replace-item 0 InfOfMyGroups 0.3 ]]


end
```

4 Simulations

We create a schedule in this way: first we identified a "question" and then we try to answer using the model, building some simulations that help us to understand the phenomenon.

1. Large population instead of little population change the result of the simulation?

   Experiments: we do 9 simulations in groups of 3, in each group all the condition are fixed except of the slider population that assumes values 100, 200, 400. We let Growing the number of agents in the world.

2. Additional Network classes change the result of simulation?

   Experiments: we do three groups of five simulations, in each groups all the conditions are fixed except of the slider NetworkClasses that starts with value 1 and will be incremented by one in each simulation. We are endowing more and more the agents with social network classes.

3. How much a connected and developed network influence the decisions?

   Experiments: we do three groups of three simulations, in each group all the variables are stable except of the sliders PFamily, PFriends, PColleagues, PBarfriends, and PSoccerfriends that assume different values in each round.

4. What is the influence of the variable ReplacementRate on the behavior of the agents? and it changes if we add a social network structure?

   Experiments: we do three groups of three simulations, in each group all the variables are stable except of the slider ReplacementRate that assume different values in each round.

5. If we add heterogeneity to the agents in terms of weights in the decisions and turning points the influences the results change?

   Experiments: we do three simulations, the first with a population of identical agents and the other two with differences in their characteristics.

6. The social network structure in our model can be a good approximation?

   Here we do not made experiments but we do a reflection on how we have create our social network structure-

4.1 Experiment 1 - Population of the agents

So we start with the first question, "Large population instead of little population change the result of the simulation??". Using the interface we set all the variables except for NumberOfAgents constant in our first group, we assume the values:

- NetworkClasse : 3,

- WhichLink? : all,
- DifferentWeights : FindTheWeightsFix,
- ReplacementRate : 0.7,
- G.Family, G.Friends, G.Colleagues : 25, 30, 30,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire : 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues : 0.25, 0.25, 0.25,
- PFamily, PFriends,Pcolleagues : 35, 25, 25
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean : all equal,
- In the first round we set NumberOfAgents equal to 100, in the second equal to 200 and finally in the third equal to 400.

We compare the plots of the interface in the three cases and we want to analyze the differences between them, of course we analyze them in groups of three because we take every plot one time from all three versions.

Comparing the plot called "Retired Agents" we note that the behavior is the same: initially for the first 27-28 ticks the value grow, then decrease until a certain point and finally stay stable around this value, all three plots have the same "stable" value of 44-45 percent of the agents in the world, but is easy to note that higher is the number of agents smother is the line, in the case of 100 agents the line is around that value but with stronger changes up and own the value while if we increase NumberOfAgents the changes still appear but with lower intensity.

Figure 23: "Retired Agents" plot, 100 agents.

In the Plots "Average Age of Retirement in the Network" we note the same thing happens, the behavior of the line is less clear in the plots with 100 agents but we can appreciate that in all three case the points of the line are distributed around the value of 67.55 years and, as in the previous plots, the line in the case of 400 agents is smoother than in the case with 200 and 100 agents.

The same situation is perceived in the plot called "Standard Deviation in the Age of Retirement in the Network" in the case of 100 agents the standard

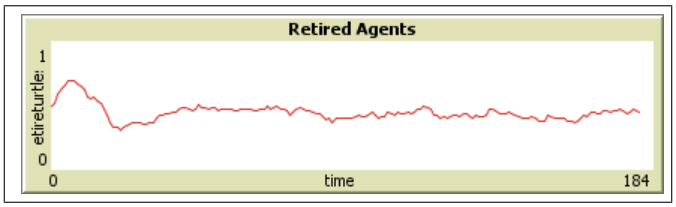

Figure 24: "Retired Agents" plot, 200 agents.

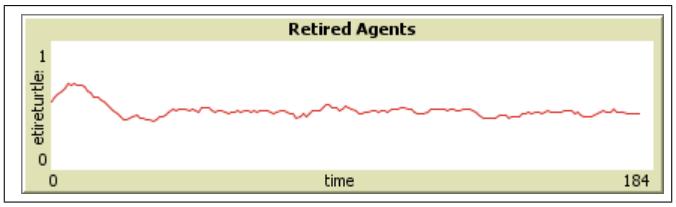

Figure 25: "Retired Agents" plot, 400 agents.

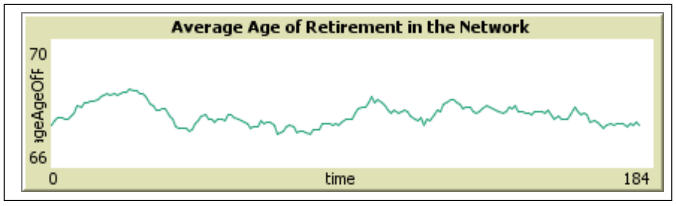

Figure 26: "Average Age of Retirement in the Network" plot, 100 agents.

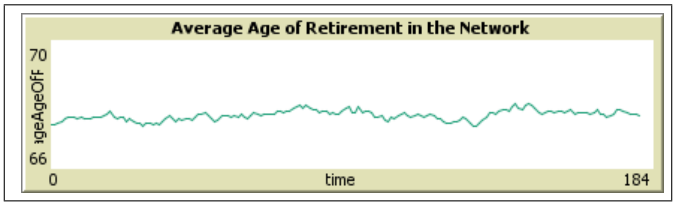

Figure 27: "Average Age of Retirement in the Network" plot, 200 agents.

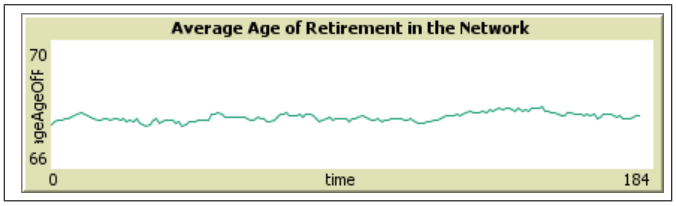

Figure 28: "Average Age of Retirement in the Network" plot, 400 agents.

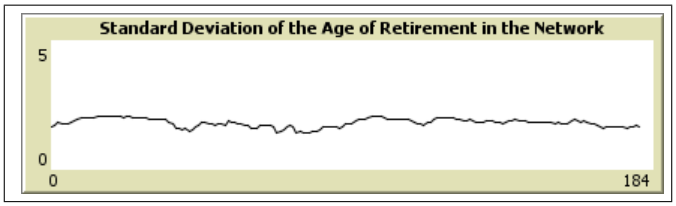

Figure 29: "Standard Deviation in the Age of Retirement in the Network" plot, 100 agents.

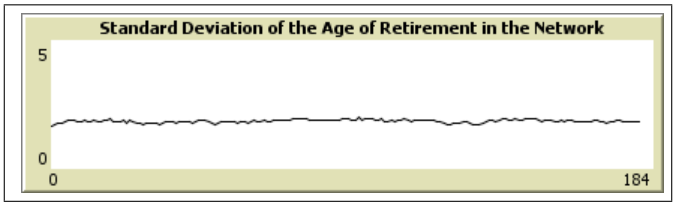

Figure 30: "Standard Deviation in the Age of Retirement in the Network" plot, 200 agents.

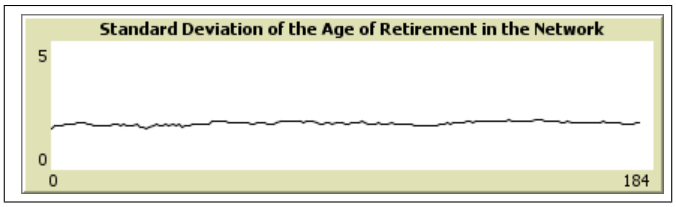

Figure 31: "Standard Deviation in the Age of Retirement in the Network" plot, 400 agents.

deviation varies a lot while already in the case with 200 agents the line is smooth around the value 1.82, and the same happens in the case with 400 agents.

So in these first three plots let growing the variable NumberOfAgents obtain the only effect to smooth the curves of the results. The situation changes with the plots "Average Clustering Coefficient", the "Density Coefficient", and the "Average Number of Links per Agents". The line of the clustering coefficient clearly step up every times we let the agents grow, in the first case the line is around the value 0.6, in the case of 200 agents is around 0.67, and in the case of 400 agents is around 0.76.

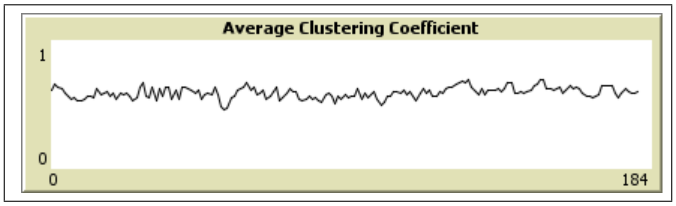

Figure 32: "Average Clustering Coefficient" plot, 100 agents.

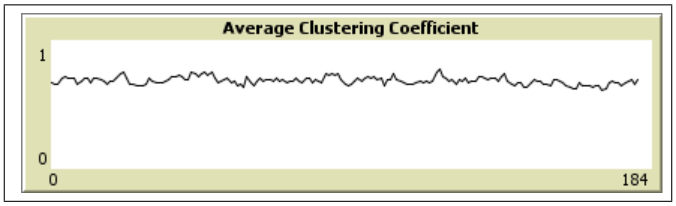

Figure 33: "Average Clustering Coefficient" plot, 200 agents.

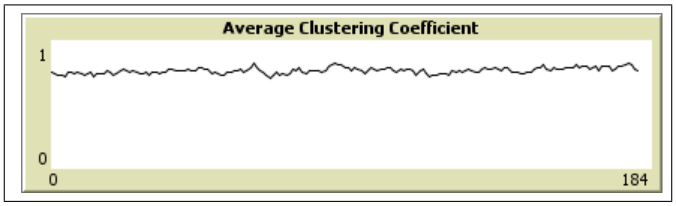

Figure 34: "Average Clustering Coefficient" plot, 400 agents.

The opposite occur in for the density coefficient, its line in the first case appear around the value 0.0045, in the second case around 0.0024, and in the last case around 0.0018. Both the coefficient line and the density one not only go up but also became smoother, so the same effect that we appreciate in the first three plots is presents also in this two.

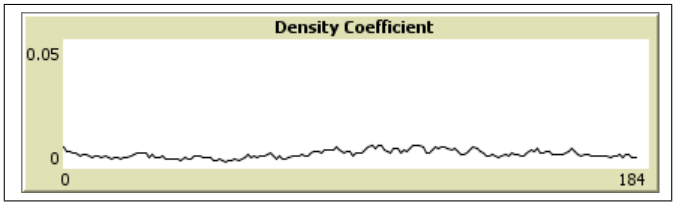

Figure 35: "Density Coefficient" plot, 100 agents.

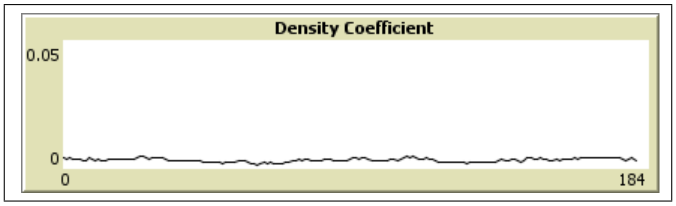

Figure 36: "Density Coefficient" plot, 200 agents.

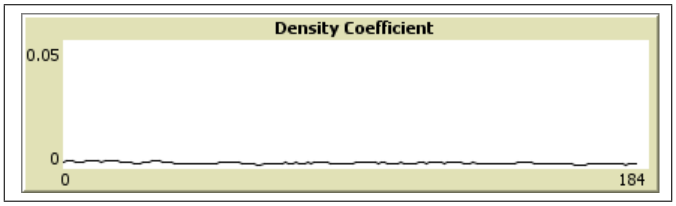

Figure 37: "Density Coefficient" plot, 400 agents.

The last plot that we describe here in this example is the "Average Number Of Links per Agents" here like in the last two plots described the situation change, during the three experiments the line goes up from around 2.5 links per agent to around 5.5 links per agent. A difference from all the others plots is that here the line does not became smoother, the changes from the âĂIJstableâĂİ value are always of the same intensity. Of course the maximum amount of links per agent find in the world, we know it from the monitor MaxLinks grow with the number of agents in the model, as well as the the maximum number of links per agent of each network classes, that we know from the monitors MaxLinksFamily, MaxLinksFriends, and MaxLinksColleagues.

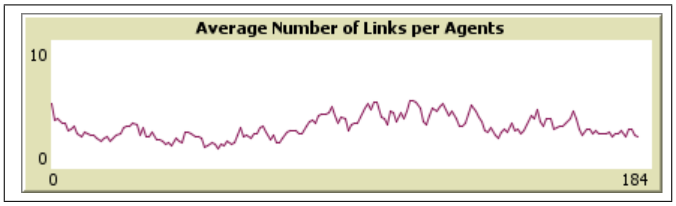

Figure 38: "Average Number Of Links per Agents" plot, 100 agents.

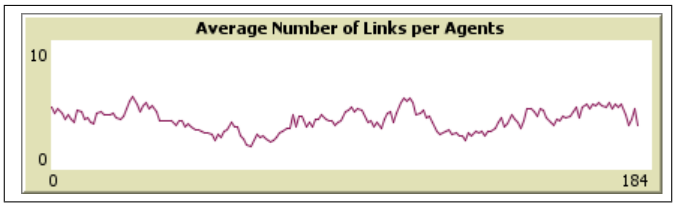

Figure 39: "Average Number Of Links per Agents" plot, 200 agents.

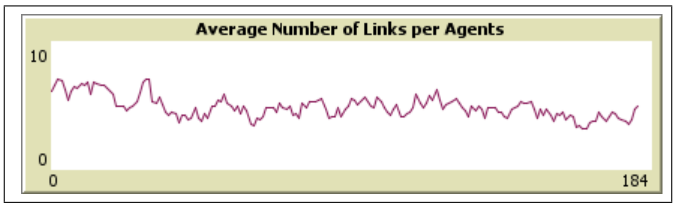

Figure 40: "Average Number Of Links per Agents" plot, 400 agents.

Now to be sure that this features does not appear only because of the specific combination that we have used we re-do the same simulations with another two setting of the variables, first

- NetworkClasse : 3,

- WhichLink? : all,
- DifferentWeights : FindTheWeightsRandom,
- ReplacementRate : 0.5,
- G.Family, G.Friends, G.Colleagues : 20, 35, 35,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire : 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues : 0.25, 0.25, 0.25,
- PFamily, PFriends,Pcolleagues : 35, 20, 20

and then,

- NetworkClasse : 3,
- WhichLink? : all,
- DifferentWeights : FindTheWeightsNormal,
- ReplacementRate : 0.4,
- G.Family, G.Friends, G.Colleagues : 25, 40, 30,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire : 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues : 0.25, 0.25, 0.25,
- PFamily, PFriends,Pcolleagues : 35, 20, 20
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean : 0.3, 0.2, 0.3, 0.2,
- P.W.st.dev., Fa.W.st.dev., Fr.W.st.mean., C.W.st.mean: all equal to 0.05
- as usual in the first round we set NumberOfAgents equal to 100, in the second equal to 200 and finally in the third equal to 400.

The second and the third group of simulations confirm the features of the first group, so its results do not depend on its the specific setting. So we can state that the number of agents does not influence too much the results of the model, is convenient anyway set the NumberOfAgents more that 100 in order to obtain a smoother behavior from the lines of information of the plots. In fact if we let NumberOfAgents grow and we do not modify the imputs G.Family, G.Friends, and G.Colleagues we let grow the average number of members of the groups of families, friends, and colleagues, in other words we have bigger groups that exchange informations between their members but we do not create new âĂIJbridgesâĂİ between a group and another. This seems to be not so efficient in changing the behavior. We can obtain the same result fixing the number of agents and let increase or decrease the monitors G.Family, G.Friends, and G.Colleagues.

4.2 Experiment 2 - Additional network classes

Our second question is "Additional Network classes change the result of simulation??", we do three groups of five simulations. We set the variables of the first one:

- WhichLink?: all,
- DifferentWeights: FindTheWeightsfix,
- ReplacementRate: 0.4,
- G.Family, G.Friends, G.Colleagues, G.Barfriends, G.Soccerfriends: 20, 35, 35, 25, 25,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire, BarfriendsToRetire, SoccerfriendsToRetire: 0.50, 0.50, 0.50,, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues, st.devBarfriends, st.devSoccerfriends: 0.25, 0.25, 0.25, 0.25, 0,25
- NumberOfAgents: 200,
- PFamily, PFriends, PColleagues, PBarfriends, PSoccerfriends: 35, 25, 25, 25, 25,
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean, B.W.mean., S.W.mean.: 0.20, 0.20, 0.20, 0.20, 0.20,
- in the first round we set NetworkClasses equal to 0 and then we increment it by one every round.

At the end we obtain five versions of each plot that we use to understand the effect of increasing NetworkClasses ceteris paribus. If we take the five "Retired Agents" plots we do not find differences in them, the same fraction of agents, around 44 percent, are retired in each period and increasing the network classes seem not have effect.

Figure 41: "Retired Agents" plot, NetworkClasses equal to 0.

However there is a change in the plot "Average Age of Retirement in the Network", adding network classes to the agents slowly decrease their average

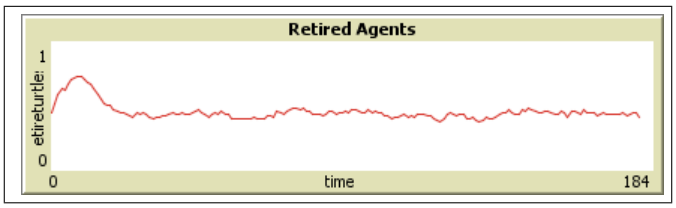

Figure 42: "Retired Agents" plot, NetworkClasses equal to 5.

age of retirement, in our example form an inital age around 68.21 to a value around 67.60. In other words we can say that more network classes push down the age of retirement of the agents. This result is also show by the second version of the plot "Average Age of Retirement in the Network", that we obtain using the chooser TypeOfAverages in the interface, that propose a comparison between the ages of retirement of the agents with links (so that are directly part of the network structure) and of agents without links (not involved directly in the network structure), the second line is higher for the most of the time meaning that the age of retirement of turtles without links is, on average, higher.

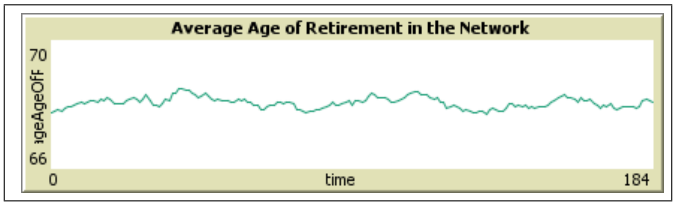

Figure 43: "Average Age of Retirement in the Network" plot, with NetworkClasses equal to 0.

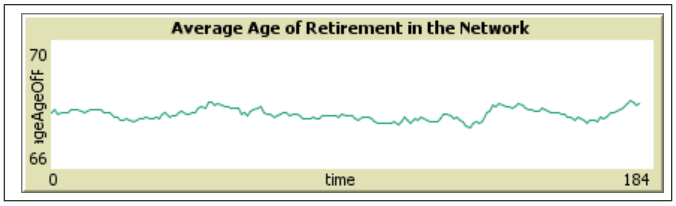

Figure 44: "Average Age of Retirement in the Network" plot, with NetworkClasses equal to 5 and TypeOfAverages equal to "oneline".

The output of the plot "Average Clustering coefficient" changes a lot if we have or not a network structure. If we set NetworkClasses equal to zero we, of course, obtain a clustering coefficient of zero because there is no network, if we

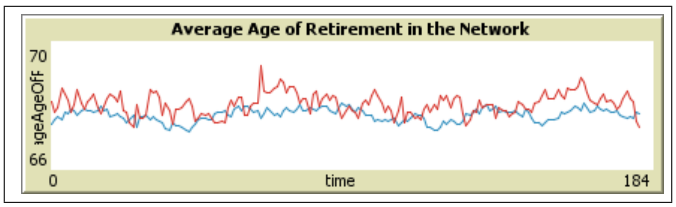

Figure 45: "Average Age of Retirement in the Network" plot, with NetworkClasses equal to 5 and TypeOfAverages equal to "twoline".

allow for only one network class, the Family relations, we obtain a clustering coefficient of 1, that is the maximum possible in a network, and this comes from the fact that each turtles can have only one family relation, only one number inside family list, and all the members of this family are connected to each others. Moreover if we add more network classes we see a gradually decrease of the clustering coefficient until a value around 0.56 in the case of 5 network classes.

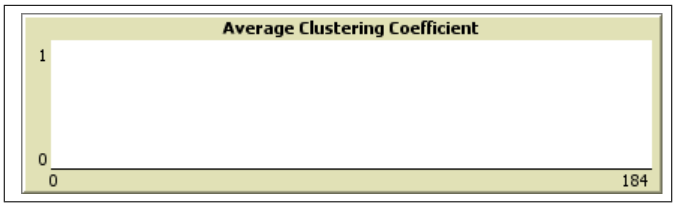

Figure 46: "Average Clustering coefficient" plot with NetworkClasses equal to 0.

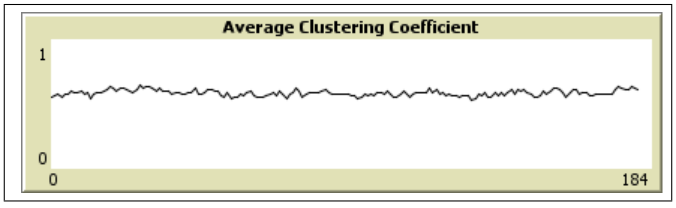

Figure 47: "Average Clustering coefficient" plot with NetworkClasses equal to 5.

Using the plot "Standard Deviation in the Age of Retirement in the Network" we understand how more network classes have no influence on the standard deviation of the age of retirement in the world, it remains always around a value of 1.82.

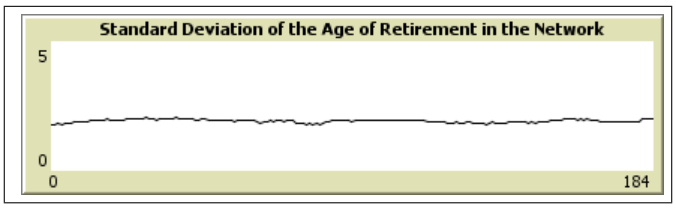

Figure 48: "Standard Deviation in the Age of Retirement in the Network" plot with NetworkClasses equal to 0.

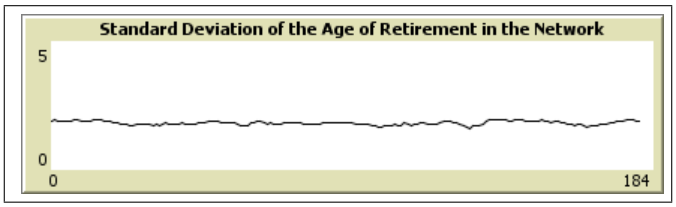

Figure 49: "Standard Deviation in the Age of Retirement in the Network" plot with NetworkClasses equal to 5.

With the plot "Average Number of Links per Agents" we can appreciate how if the slider NetworkClasses increases its value the number of links per agents increse, the line goes up from 0, in the case of no network structure, to around 8.1 links per agent in the case with 5 network classes.

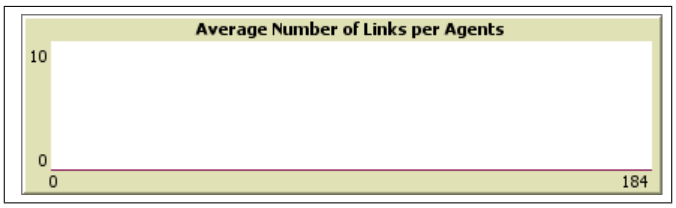

Figure 50: "Average Number of Links per Agents" plot with NetworkClasses equal to 0.

Finally the plot "Density Coefficient", if we have NetworkClasses equal to 0 the density coefficient is, of course, equal to 0 while if we add a network structure it assumes a value around 0.0029 indipendently of how much network classes there are.

We repeat the same experiments with others two different settings in order to prove that these features do not depend on the features of the previous simulation, so now we use first,

- WhichLink?: all,

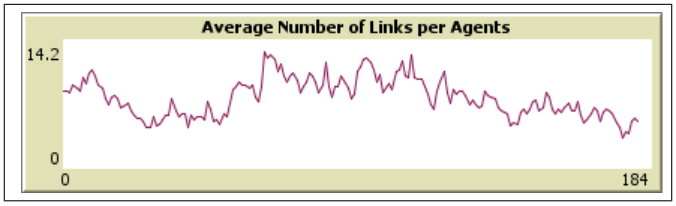

Figure 51: "Average Number of Links per Agents" plot with NetworkClasses equal to 5.

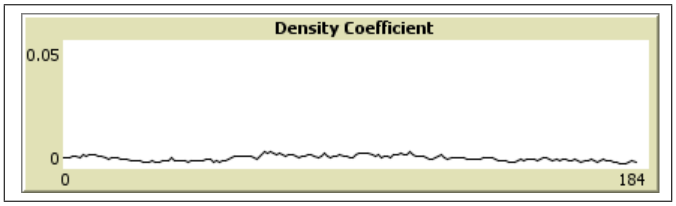

Figure 52: "Density Coefficient" plot with NetworkClasses equal to 5.

- DifferentWeights: FindTheWeightsNormal,
- ReplacementRate: 0.2,
- G.Family, G.Friends, G.Colleagues, G.Barfriends, G.Soccerfriends: 20, 35, 35, 25, 25,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire, BarfriendsToRetire, SoccerfriendsToRetire: 0.50, 0.50, 0.50,, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues, st.devBarfriends, st.devSoccerfriends: 0.25, 0.25, 0.25, 0.25, 0,25
- NumberOfAgents: 200,
- PFamily, PFriends, PColleagues, PBarfriends, PSoccerfriends: 40, 20, 20, 20, 20,
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean, B.W.mean., S.W.mean.: 0.20, 0.20, 0.20, 0.20, 0.20,
- P.W.st.dev., Fa.W.st.dev., Fr.W.st.dev., C.W.st.dev., B.W.st.dev., S.W.st.dev., : all equal to 0.05,
- in the first round we set NetworkClasses equal to 0 and then we increment it by one every round.

In this setting we can fully appreciate how social connections push down the average age of retirement, with a replacement rate of 0.2 there is for the agents the incentive to work in order to obtain benefits during the years between age 66 and 70 and this is confirmed by the behaviors of the agents, when the slider NetworkClasses is equal to 0 the average age of retirement is arount 70. But if we add the family dimension the line in the plot "Average Age of Retirement in the Network" goes down around the value of 69, and if we add another dimension, the friends one, the line goes again down to a value around 68.21. If we endowed the agents with five social dimension the line stay around a value behind the 68 years and this means that the behaviors of agents with and without social network sructure differ by more than two years. This indicate a huge influence that the social network structure can have on the decisions of the agents.

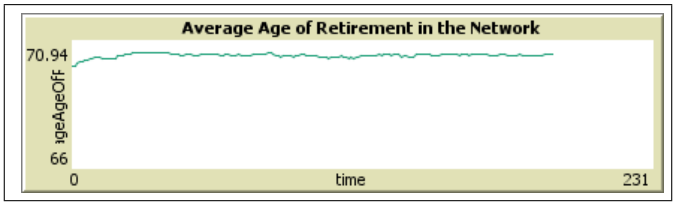

Figure 53: "Average Age of Retirement in the Network" plot with NetworkClasses equal to 0.

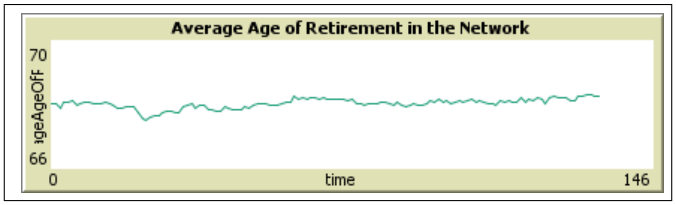

Figure 54: "Average Age of Retirement in the Network" plot with NetworkClasses equal to 5.

Then we use the second setting :

- WhichLink? : all,
- DifferentWeights : FindTheWeightsNormal,
- ReplacementRate : 0.9,
- G.Family, G.Friends, G.Colleagues : 20, 35, 35, 25, 25,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire, BarfriendsToRetire, SoccerfriendsToRetire: 0.50, 0.50, 0.50, 0.50, 0.50,

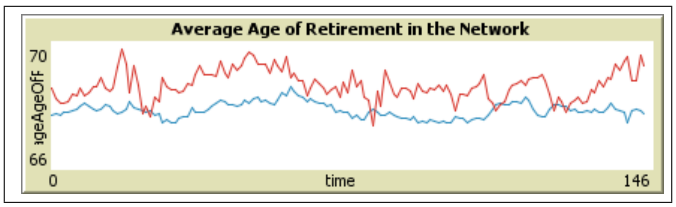

Figure 55: "Average Age of Retirement in the Network" plot with NetworkClasses equal to 5 and TypeOfAverages equal to "twoline".

- st.devFamily, st.devFriends, st.devColleagues, st.devBarfriends, st.devSoccerfriends: 0.25, 0.25, 0.25, 0.25, 0,25
- NumberOfAgents: 200,
- PFamily, PFriends, PColleagues, PBarfriends, PSoccerfriends: 40, 20, 20, 20, 20,
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean, B.W.mean., S.W.mean.: 0.20, 0.20, 0.20, 0.20, 0.20,
- P.W.st.dev., Fa.W.st.dev., Fr.W.st.dev., C.W.st.dev., B.W.st.dev., S.W.st.dev., : all equal to 0.05,
- in the first round we set NetworkClasses equal to 0 and then we increment it by one every round.

Here is very intresting to note one thing: before with no social influence we discover that if the replacement rate is very low the agents adopt the maximum age possible to retire, but now we see that if the replacement rate is high it seems that they do not retire early as soon as they can, at age 66. This is not true, in fact they retire as soon as they can but the presence of agents that are obliged to retire later than age 66 because of a low level of contribution years made the average retirement age be higher. For this reason with an high replacement rate the social influence cannot push down the line of average retirement age, and so with or without social influence the agents always retire around age 67.51.

4.3 Experiment 3 - More connected network

Our third question is "How much a connected and developed network influence the decisions?", we try to answer with three groups of three simulations. In the first group we set:

- WhichLink? : all,
- DifferentWeights : FindTheWeightsfix,
- ReplacementRate : 0.5,

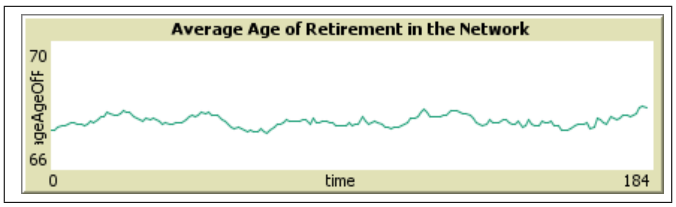

Figure 56: "Average Age of Retirement in the Network" plot with NetworkClasses equal to 5.

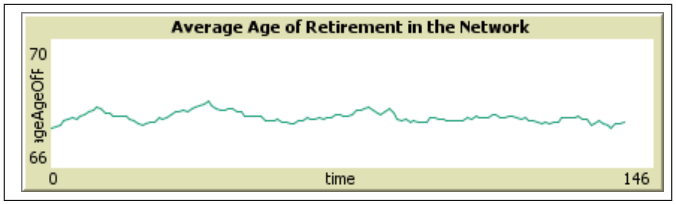

Figure 57: "Average Age of Retirement in the Network" plot with NetworkClasses equal to 5.

- G.Family, G.Friends, G.Colleagues : 20, 35, 35,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire : 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues : 0.25, 0.25, 0.25,
- NumberOfAgents: 200,
- NetworkClasses: 3
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean : 0.20, 0.20, 0.20,
- in the first round we set PFamily equal to 25, PFriends and PColleagues equal to 15. In the second round PFamily equal to 40, PFriends and PColleagues equal to 30. In the third round PFamily equal to 50, PFriends and PColleagues equal to 40.

As before we compare the result obtained from the three rounds, this time we are studing the influence of modify the probability of creating a new link (in all the network classes contemporaneously). If we compare the plots "Retired Agents" we do not find differences, the proportion on retired agents is always around 45 percent. We do not find changes also in the plots "Average Age of Retirement in the Network","Average Clustering coefficient", and "Standard Deviation in the Age of Retirement in the Network". Instead we observe changes in the average number of links per agent and in the density coefficient, the line of the first in the plot "Average Number of Links per Agents" goes up froma

around 2.8 to around 5.5 and the line of the plot "Density Coefficient" goes up from 0.024 to 0.041.

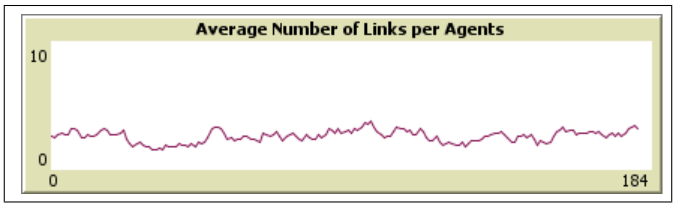

Figure 58: "Average Number of Links per Agents" plot with PFamily equal to 25, PFriends and PColleagues equal to 15.

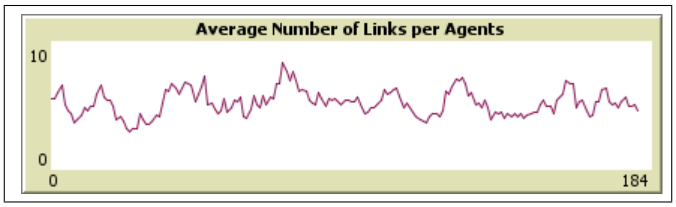

Figure 59: "Average Number of Links per Agents" plot with PFamily equal to 50, PFriends and PColleagues equal to 40.

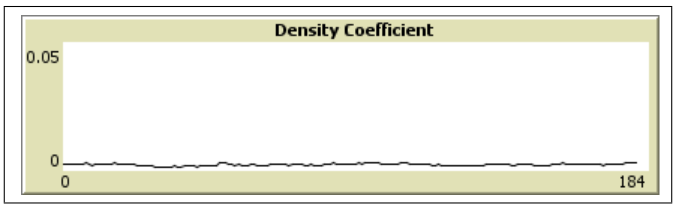

Figure 60: "Density Coefficient" plot with PFamily equal to 25, PFriends and PColleagues equal to 15.

In order to verify the conclusions we try other two different groups of simulations with different settings, the first is,

- WhichLink? : all,
- DifferentWeights : FindTheWeightsRandom,
- ReplacementRate : 0.6,
- G.Family, G.Friends, G.Colleagues, G.Barfriends : 40, 70, 70, 50,

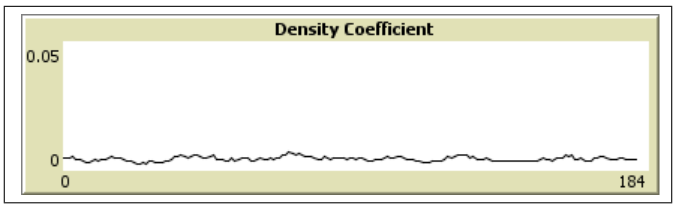

Figure 61: "Density Coefficient" plot with PFamily equal to 50, PFriends and PColleagues equal to 40.

- FamilyToRetire, FriendsToRetire, ColleaguesToRetire, BarfriendsToRetire: 0.50, 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues, st.devBarfriends : 0.25, 0.25, 0.25, 0.25
- NumberOfAgents: 200,
- NetworkClasses: 4
- in the first round we set PFamily equal to 25, PFriends and PColleagues equal to 15. In the second round PFamily equal to 40, PFriends and PColleagues equal to 30. In the third round PFamily equal to 50, PFriends and PColleagues equal to 40.

The second setting instead is:

- WhichLink? : all,
- DifferentWeights : FindTheWeightsNormal,
- ReplacementRate : 0.9,
- G.Family, G.Friends, G.Colleagues : 20, 35, 35,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire : 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues : 0.25, 0.25, 0.25,
- NumberOfAgents: 200,
- NetworkClass: 3,
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean : all equal,
- P.W.st.dev., Fa.W.st.dev., Fr.W.st.mean., C.W.st.mean: all equal to 0.05,
- in the first round we set PFamily equal to 25, PFriends and PColleagues equal to 15. In the second round PFamily equal to 40, PFriends and PColleagues equal to 30. In the third round PFamily equal to 50, PFriends and PColleagues equal to 40.

This two groups of simulations confirm the observations made on the first group.

4.4 Experiment 4 - Replacement rate

In our question four we want to study the Variable ReplacementRate and its influence on the agents. We use three groups of simulations in order to test the different situations: every group is composed by three simulations, the first test high, low, and medium replacement rate in a environment without network structure, the second test the same three levels in an environment with 3 network classes, and the third in an environment with five network classes.

The first group of simulation is made with this variables:

- WhichLink? : all,
- DifferentWeights : FindTheWeightsNormal,
- G.Family, G.Friends, G.Colleagues : 20, 35, 35, 25,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire : 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues : 0.25, 0.25, 0.25,
- NumberOfAgents: 200,
- NetworkClass: 0,
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean : all equal,
- P.W.st.dev., Fa.W.st.dev., Fr.W.st.mean., C.W.st.mean: all equal to 0.05,
- PFamily, PFriends,Pcolleagues : 35, 25, 25,
- we fix the value of the slider ReplacementRate equal to 0.2 in the first turn, then 0.5 and then 0.9.

With this group of simulations we study the behavior of the agents without the social influence, the variable replacement rate has an huge influence on the result of the plot "Average Age of Retirement in the Network", in the case of low replacement rate the agents retire around an age on 70 years, while in the case of medium and high level of the replacement rate they retire around 67.5 years. We have already explain because the change from a medium level of replacement rate to an high level do not lead to a further decrease in the age of retirement, it is simply because that some agents that have not reach the requirement of the years of contribution cannot retire before. So the behavior of the agents without social influences is clear they retire as soon as possible if there is a adequate replacement rate and procrastinate retirement in orther to obtain benefits if the replacement rate is low.

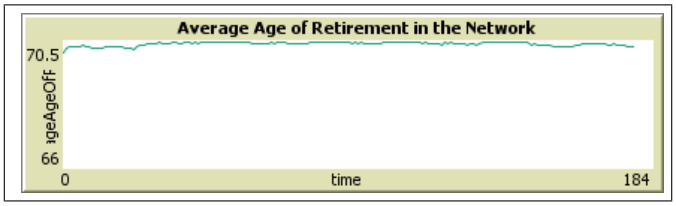

Figure 62: "Average Age of Retirement in the Network" plot with ReplacementRate equal to 0.2, with no social Influence.

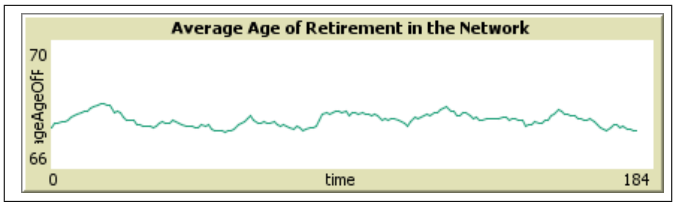

Figure 63: "Average Age of Retirement in the Network" plot with ReplacementRate equal to 0.5, with no social Influence.

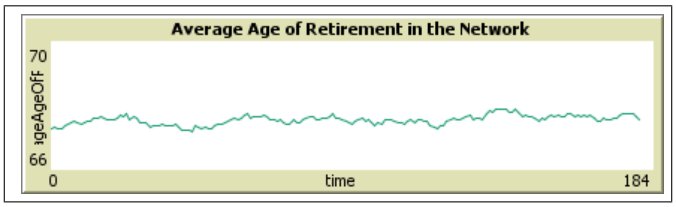

Figure 64: "Average Age of Retirement in the Network" plot with ReplacementRate equal to 0.9, with no social Influence.

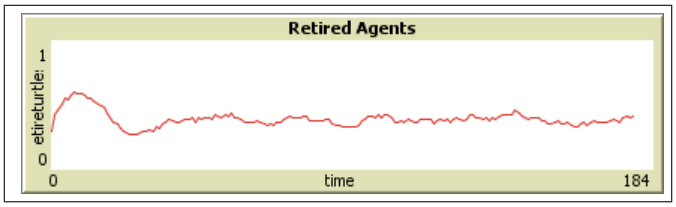

Figure 65: "Retired Agents" plot with ReplacementRate equal to 0.2.

The plot "Retired Agents" in the case of low replacement rate show a decrease in the percent of people that have retired, due to the fact that the agents that reach the pension time are less because of the increase of the age of retirement-

Moreover we see an increase in the expenditure of the pension system as the replacement rate goes up.

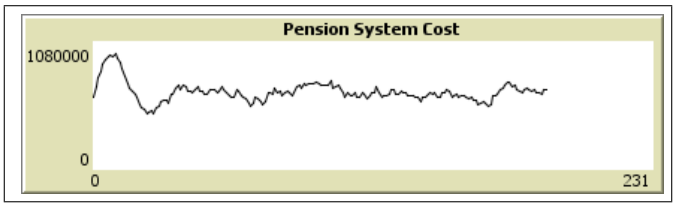

Figure 66: "Pension System Cost" plot with ReplacementRate equal to 0.2.

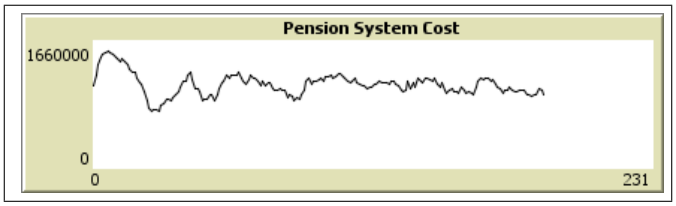

Figure 67: "Pension System Cost" plot with ReplacementRate equal to 0.5.

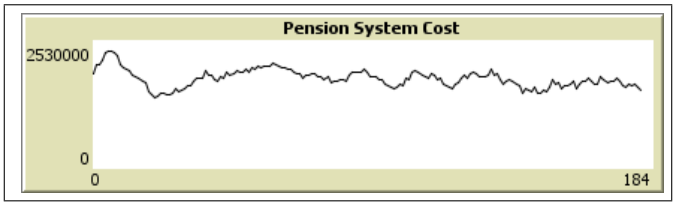

Figure 68: "Pension System Cost" plot with ReplacementRate equal to 0.9.

The second group of simulations follow this setting:

- WhichLink? : all,
- DifferentWeights : FindTheWeightsNormal,
- G.Family, G.Friends, G.Colleagues : 20, 35, 35,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire : 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues : 0.25, 0.25, 0.25,

- NumberOfAgents: 200,
- NetworkClass: 3,
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean : all equal,
- P.W.st.dev., Fa.W.st.dev., Fr.W.st.mean., C.W.st.mean: all equal to 0.05,
- PFamily, PFriends,Pcolleagues : 35, 25, 25,
- we fix the value of the slider ReplacementRate equal to 0.2 in the first turn, then 0.5 and then 0.9.

These Simulations compared with the ones of the first group introduce a social network structure made of three network classes. In this setting if we have a low replacement rate agents retire around 68.26 years while if we have a medium or high replacement rate agents retire around 67.46 years, this two values are both smaller than the corresponding ones found in the first group of simulations. Within this two groups of simulations we can appreciate how the social dimension decrease the average age of retirement.

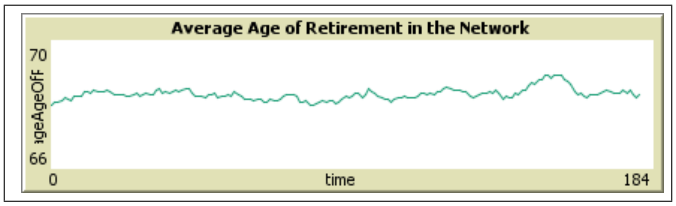

Figure 69: "Average Age of Retirement in the Network" plot with ReplacementRate equal to 0.2, with social Influence.

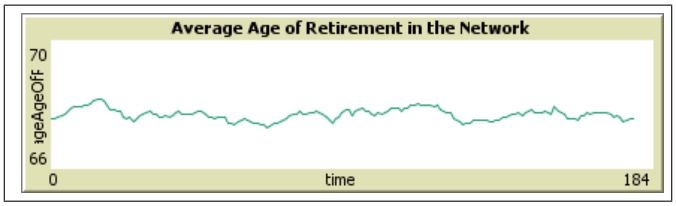

Figure 70: "Average Age of Retirement in the Network" plot with ReplacementRate equal to 0.5, with social Influence.

The plot "Average Age of Retirement in the Network" shows this fact too, if we use the chooser TypeOfAverages to obtain the difference in the retirement age between agents involved in the network structure and agents not involved.

The third group of simulations runs with this setting:

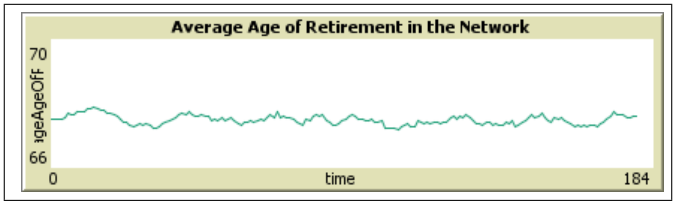

Figure 71: "Average Age of Retirement in the Network" plot with ReplacementRate equal to 0.9, with social Influence.

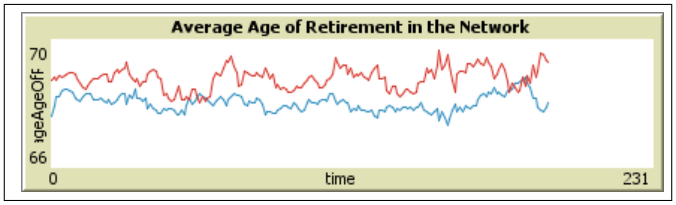

Figure 72: "Average Age of Retirement in the Network" plot with the slider TypeOfAverages equal to "twoline".

- WhichLink? : all,
- DifferentWeights : FindTheWeightsNormal,
- G.Family, G.Friends, G.Colleagues, G.Barfriends, G.Soccerfriends: 20, 35, 35, 25, 25,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire, BarfriendsToRetire, SoccerfriendsToRetire: 0.50, 0.50, 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues, st.devBarfriends, st.devSoccerfriends: 0.25, 0.25, 0.25, 0.25, 0.25,
- NumberOfAgents: 200,
- NetworkClass: 5,
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean, B.W.mean., S.W.mean.: all equal,
- P.W.st.dev., Fa.W.st.dev., Fr.W.st.dev., C.W.st.dev., B.W.st.dev., S.W.st.dev., : all equal to 0.05,
- PFamily, PFriends, PColleagues, PBarfriends, PSoccerfriends: 35, 25, 25, 25, 25,
- we fix the value of the slider ReplacementRate equal to 0.2 in the first turn, then 0.5 and then 0.9.

With this group of simulations we study the behavior of the agents within an environment with a social network structure of five network classes. These simulations support our previous conclusions on the influnce of the social network structure on the behavior of the agents.

4.5 Experiment 5 - Hetereogeneity

Our fifth question states "If we add heterogeneity to the agents in terms of weights in the decisions and turning points the influences the results change?", in other words we start our line of reasoning from the fact that in the real world every agent has a specific setting of the variables that influence his choices, the weights in the decision and the turning points of the classes are very specific for each agent and related to his/her experience as human being, we can refer to this fact saying that the heterogeneity of the agents in the real world is very high. In order to undestand how much the heterogeneity of the agents influence the whole system we run the model first with no differences in the starting conditions between the agents, then we run the model with hetereogenity in the starting conditions and finally we increment this hetereogenity. We try two different ways to introduce hetereogeneity in the model, one is to set the weights random and the turning points of the classes from a normal distribution with an high standard deviation, while the second is to set the weights and turning points of the classes both from a normal distribution with an high standard deviation.

the settings that we use in this experiment are three: the first is the case with identical agents,

- WhichLink? : all,
- DifferentWeights : FindTheWeightsFix,
- G.Family, G.Friends, G.Colleagues : 20, 35, 35,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire : 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues : 0, 0, 0,
- NumberOfAgents: 200,
- NetworkClass: 3,
- P.W.mean, Fa.W.mean, Fr.Wmean: all equal,
- P.W.st.dev., Fa.W.st.dev., Fr.W.st.dev: all equal to 0,
- PFamily, PFriends, Pcolleagues : 35, 25, 25,
- ReplacementRate: 0.2,

while in the second we insert heterogeneity of the agents,

- WhichLink? : all,
- DifferentWeights : FindTheWeightsRandom,
- G.Family, G.Friends, G.Colleagues, G.Barfriends: 20, 35, 35, 25,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire, BarfriendsToRetire : 0.50, 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues, st.devBarfriends : 0.50, 0.50, 0.50, 0.50,
- NumberOfAgents: 200,
- NetworkClass: 4,
- PFamily, PFriends, Pcolleagues, PBarfriends: 35, 25, 25, 25,
- ReplacementRate: 0.6,

and in the third we do the same but using another way,

- WhichLink?: all,
- DifferentWeights: FindTheWeightsNormal,
- ReplacementRate: 0.7,
- G.Family, G.Friends, G.Colleagues, G.Barfriends, G.Soccerfriends: 20, 35, 35, 25, 25,
- FamilyToRetire, FriendsToRetire, ColleaguesToRetire, BarfriendsToRetire, SoccerfriendsToRetire: 0.50, 0.50, 0.50, 0.50, 0.50,
- st.devFamily, st.devFriends, st.devColleagues, st.devBarfriends, st.devSoccerfriends: 0.25, 0.25, 0.25, 0.25, 0,25
- NumberOfAgents: 200,
- NetworkClass: 5,
- PFamily, PFriends, PColleagues, PBarfriends, PSoccerfriends: 35, 25, 25, 25, 25,
- P.W.mean, Fa.W.mean, Fr.Wmean, C.W.mean, B.W.mean., S.W.mean.: all equal,
- P.W.st.dev., Fa.W.st.dev., Fr.W.st.dev., C.W.st.dev., B.W.st.dev., S.W.st.dev., : all equal to 0.20,

Our expectation is to find a correlation between the presence of hetereogeneity in the agents and the standard deviation of the average age of retirement in the model. Unfortunately we do not find such a strong correlation, from the plot we can appreciate that in all three round the level of the standard deviations stay constant around a value of 1.94. However the lines in the thirda and second plots are more variable in their behaviors than the line in the first plot, this change is reasonably due to the introduction of hetereogenity of the agents.

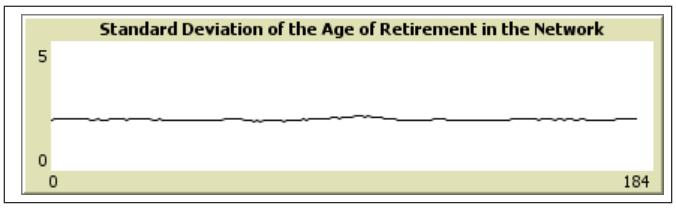

Figure 73: "Standard Deviation in the Age of Retirement in the Network" plot, first setting of the experiment.

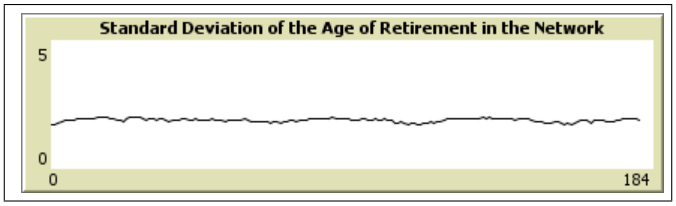

Figure 74: "Standard Deviation in the Age of Retirement in the Network" plot, second setting of the experiment.

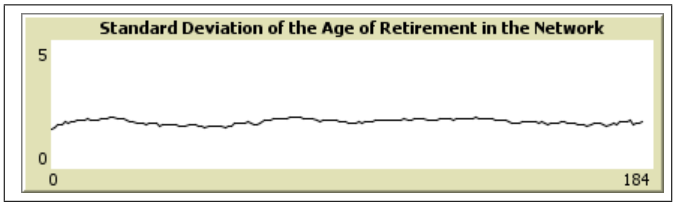

Figure 75: "Standard Deviation in the Age of Retirement in the Network" plot, third setting of the experiment.

4.6 Experiment 6 - Network structure

In this part we want to make a reflection on the way that we use to simulate social network structure, a lot of excelent studies have been made on the subject and have obtained very good results but we are confidend that our choice is a good way to model the reality, we try to stay in line with the idea, common in ABMs, to hold a simply approach and see how agents responds using simply rules in complex environment.

We have already explained how social relations are formed in our code, every turtle has a list containing other 5 lists and two turtles have a social relation if they have the same number, does not matter the position of the number in the list, in the same list, here the position of the list is important because indicates the type of network relation. What we obtain is a network structure made of groups of fully connected agents, better say if we take a single group of relations, for example all the turtles that have in the second list of the list ClassOfNetworkList the number "2", the group of agents involved is fully connected. This is a very simple rule of connecting agents but we develop the range of this rule in a very intresting fashion.

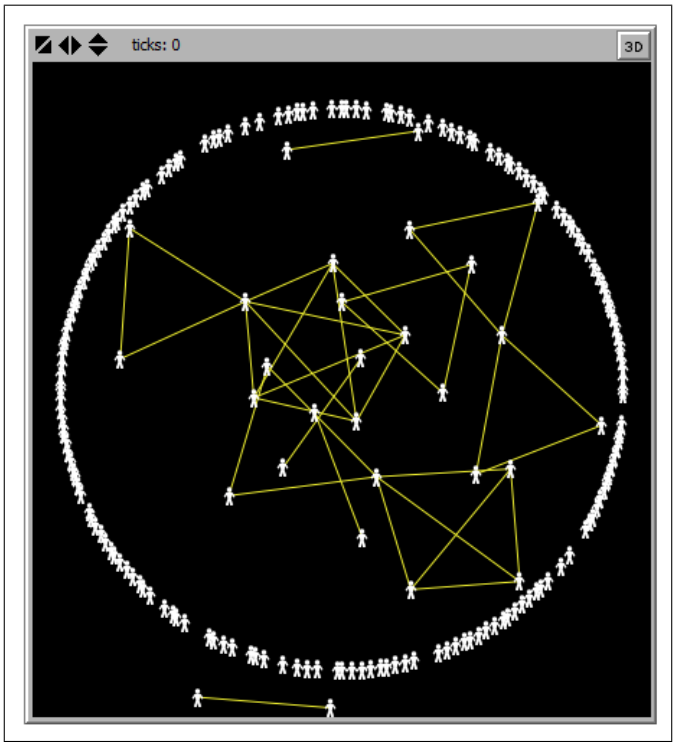

Figure 76: Example of agents connections in groups.

- First, inside each network classes, every agents can have more than one number for each list, this means that each agents can be part of more than one network groups, in other words a turtle can became the bridge through which one group influence the other.
- Second, taking in consideration the whole network structure, every agents can be part of two or more social groups of two or more different network classes, for example be member of one family group and one friends group, and becames the bridge through which one group influence the other across network classes.

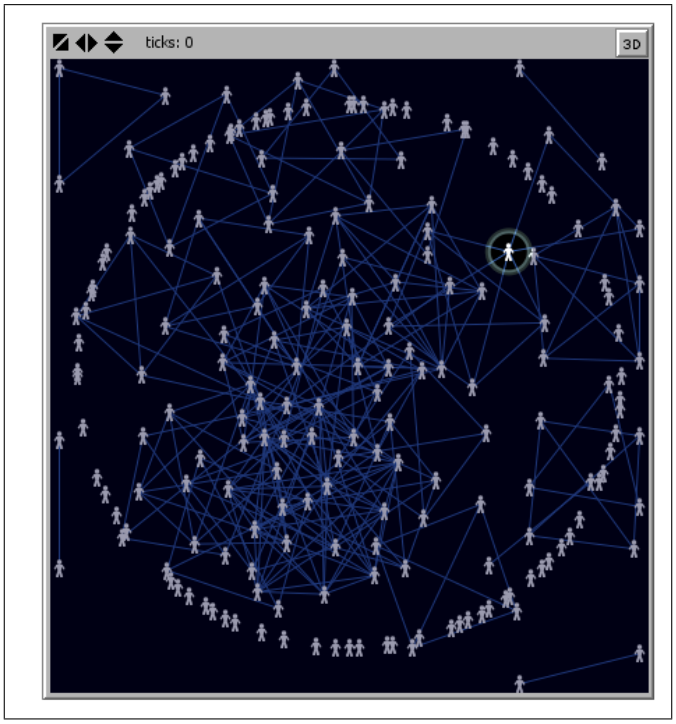

Figure 77: Example of an agent connecting two social groups from the same network class.

The final result of the network structure has some good properties that indicates that represent a good approximation of the reality:

- The system has a low density coefficient, this feature fully reflects the reality of a social network structure between human being where only few connections over the theorically possible are actually create.
- The system has some agents more connected than others, this result is

obtained from the using of the social network structure and the variable SocialStrenght and reflects the hetereogeneity of the real world.

- The system has a good clustering coefficient from the agents directly involved in the social network.

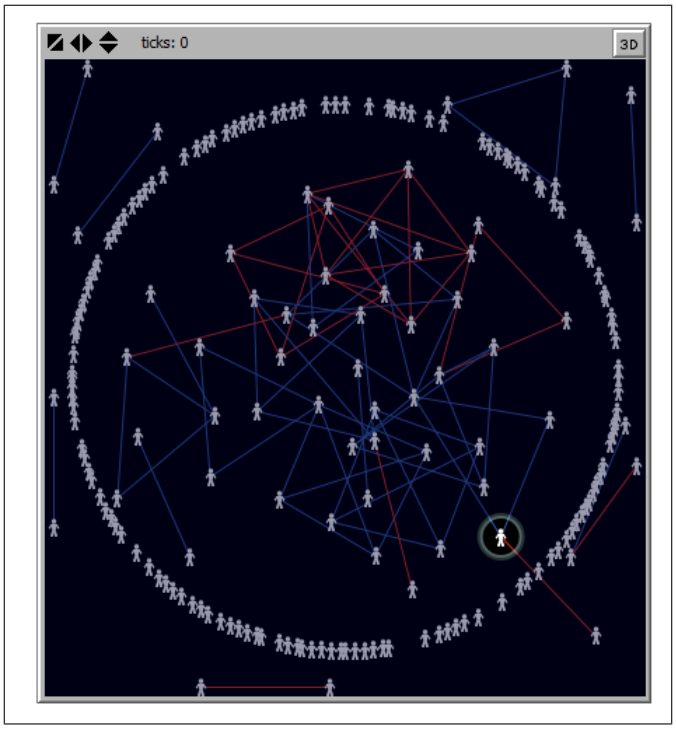

Figure 78: Example of an agent connecting two social groups from different network classes.

5 Conclusions

The aim of this work is try to answer to the question "do social connections modify the behavior of the agents in the decision of retirement?", we think that using this model we have improved our comprehension over the subject.

The first and most important result of our study is that agents included in a social network retire early (or at least at the same time) compared to agents without social relations that facing the same conditions, as we can see in the Experiment 2 when we add social network classes to the model, that means we increase the social dimension, the average age of retirement decrease, we can clearly appreciate this from Figure53 and Figure54. This is a strong result that gives new indications to policymakers, it is a common and broadly accepted idea that the increase of the life expectancy and the change in the composition of the society ask for a parallel increase in the age of retirement but the incentives created in order to obtain this result are always focused on the personal dimension and do not take in consideration the social dimension of the decision, the effectiveness of personal incentives directed to agents living in a social framework is strongly weakened. In our experiments we note that the decrease in the retirement age that we observe when we add one more network class decrease in absolute value every time we add a network class, in other words when we add the first network class the change inaverage retirement age in very big, with the second network class the change is slighter and the same happens for all the additional network classes. This result tell us that to obtain such a negative influence over the retirement age is not necessary needed a highly connected and developed network structure but also with a simple one there ia a big influence on the decisions of the agents.

We find our second result in the experiment number 1 and in the experiment number 3 together, in both experiments we increase the number of links in the world as we can appreciate from Figure40 and Figure59 but even thought the result is similar, the links in the network increase, there is a difference between the two ways. The buildup of the first experiment is due to the increase of the number of agents and we can say that this is a rude way to obtain more links, instead the increment in the second experiment is due to the increase in the probability of create a social connection between the agents and this is a more qualitative way to obtain more links. The network structure obtained from the first experiment is composed by bigger groups of agents but the structure as a whole do not bacame more connected, in other words there are the same number of groups in the society but the size of this groups increase. In the second experiment instead the size of the groups increase but less than in the previous case while the structure as a whole became more connected with more agents that are member of more groups. It is clear that the Network structure of the second experiments is better and our expectation was that it obtain better performances in transmitting the incentives to the agents, and ultimately this is true. The two network structures obtain:

- the same performances in terms of average age of retirement, both plots

show a value around 67.55, but if we take in consideration only the agents actually involved in the network structure we see that the "qualitative", more connected structure transmits better the incentives to the agents, the agents retire later with a low replacement rate and with an high replacement rate.

- the second network structure present a smaller value of standard deviation of the retirement age, it is 1.72 compared to 1.82 of the first network structure.

We can appreciate this results if we compare Figure28 and Figure31 from experiments 1 to Figure79 of experimets 2, in other words the more "qualitative" network structure helps in decreasing the standard deviation of the age of retirement, and also in this sense is more efficient in transmitting the "right" message to the agents, and also comunicates better the incentives to the agents actally involved in the network structure.

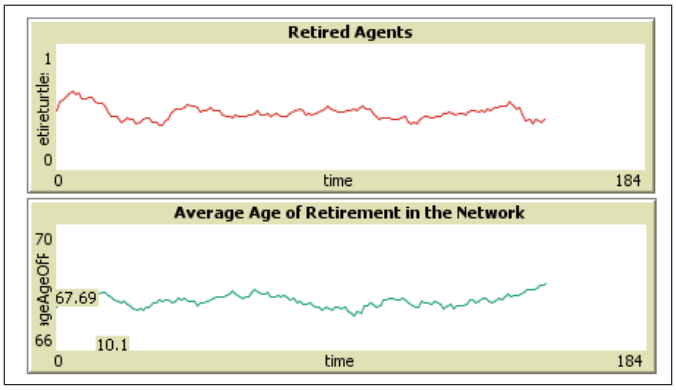

Figure 79: "Retired Agents" and "Average Age of Retirement in the Network" plots with "qualitative" network structure.

The third result that we find using our model is centered on how the replacement rate influence the agents, this is the most predictable of the three results that we have achived. We improve our knowledge of how low replacement rate induce to retire later while high replacement rate induce to retire early. What we discover is that although with the presence of connections between the agents the replacement rate is the main variable that guide the behavior of the society as a whole, and in our model the replacement rate hold the more general concept of incentives.

What we can finally say to answer our initial question is that as expected the personal economic incentives are the main variable that influence the agents and their choices but in a connected society the influence of this variable do not have the strenght that we usual suppose, in the implementation of a policy over

the subject like pension system is important to have this fact clear in order to calibrate the measures to really achive the desire situation.

6 Future Developments

Despite the fact that we consider our model a good approximation of the reality of course many things can be done in order to improve its features. Here we consider two possible improvements that can be done:

- define a utility function that the agents use instead the PProbRetire,
- define different rules to create the network structure, based on different principles such for example the proximity principle or social distance,

We consider the first point very intresting, the idea is find those variables that influence the personal probability to retire of the agents and define a society composed by members endowed with these variables. Using this approach means mix, in some sense, the ABMs theory of our model with the classical concept of utility function, to do so we have to assume that the propability depends on some variables that we study but using econometrics is surely possible to obtain a better approximation than a random selection (that is what we used in our model). Moreover this utility function does not necessarily need to be fix but we can imagine a kind of dynamics related to what happens in the world, in other words that the weights inside the utility function changes according to some conditions.

The second point that we indicate as a possible development is the rules used to create the social structure, in these framework there are many open possibilities in the litterature, here we present two of them.

- A solution can be used a proximity principle: we can consider the simple physical distance between agents in the world as related to the robability of creating a link, in order to capture the fact that if two agents live close to each others they will probably establish a connections. If we add also a procedure that makes the turtles move during their life we can obtain a dynamics network structure model of the society.
- Another solution is proposed by Hamill and Gilbert (2009), they define what they call social distance that measures the similarity between agents, in this view

  > ... those are short distances apart are likely to interact, Hamill and Gilbert (2009).

  They propose a world based on a social map where closer are two agents, stronger is the tie between them and where each agent has a circle around himself that represent his social possibility, he can establish a connections only with other agents that are inside the radious of his circle.

References

(????).

Aaron, H. J. (2010). *Behavioral dimensions of retirement economics*. Brookings Institution Press.

Axelrod, R. and Tesfatsion, L. (2006). *Appendix AA Guide for Newcomers to Agent-Based Modeling in the Social Sciences*. In «Handbook of computational economics», vol. 2, pp. 1647–1659.

Axtell, R. (2006). *Coordination in Transient Social Networks: an Agent-Based Computational Model of the Timing of Retirement*. In «Generative social science: Studies in agent-based computational modeling», p. 146.

Borella, M. and Moscarola, F. C. (2010). *Microsimulation of pension reforms: behavioural versus nonbehavioural approach*. In «Journal of Pension Economics and Finance», vol. 9(04), pp. 583–607.

Conte, R., Gilbert, N., Bonelli, G., Cioffi-Revilla, C., Deffuant, G., Kertesz, J., Loreto, V., Moat, S., Nadal, J.-P., Sanchez, A. *et al.* (2012). *Manifesto of computational social science*. In «The European Physical Journal Special Topics», vol. 214(1), pp. 325–346.

Goyal, S. (2012). *Connections: an introduction to the economics of networks*. Princeton University Press.

Hamill, L. and Gilbert, N. (2009). *Social circles: A simple structure for agent-based social network models*. In «Journal of Artificial Societies and Social Simulation», vol. 12(2), p. 3.

Queisser, M. and Whitehouse, E. (2005). *Pensions at a glance: public policies across OECD countries*. In .

Reilly, D. Q., Cavalleri (2013). *Pensions at a Glance: OECD and G20 Indicators*. In .

Richiardi, M. (2014). *The Missing Link: AB Models and Dynamic Microsimulation*. In *Artificial Economics and Self Organization*. Springer, pp. 3–15.

Waldherr, A. and Wijermans, N. (2013). *Communicating Social Simulation Models to Sceptical Minds*. In «Journal of Artificial Societies and Social Simulation», vol. 16(4), p. 13.