Dottorato di Ricerca in Informatica
Università di Bologna, Padova
Settore scientifico disciplinare di afferenza: INF/01

Self-Organizing Mechanisms for Task Allocation in a Knowledge-Based Economy

Andrea Marcozzi

Marzo 2009

Coordinatore:	Relatore:	Tutore:
Prof. Simone Martini	Prof. Edoardo Mollona	Prof. Özalp Babaoğlu

Abstract

A prevalent claim is that we are in knowledge economy. When we talk about knowledge economy, we generally mean the concept of "Knowledge-based economy" indicating the use of knowledge and technologies to produce economic benefits. Hence knowledge is both tool and raw material (people's skill) for producing some kind of product or service. In this kind of environment economic organization is undergoing several changes. For example authority relations are less important, legal and ownership-based definitions of the boundaries of the firm are becoming irrelevant and there are only few constraints on the set of coordination mechanisms. Hence what characterises a knowledge economy is the growing importance of human capital in productive processes (Foss, 2005 [31]) and the increasing knowledge intensity of jobs (Hodgson, 1999 [44]). Economic processes are also highly intertwined with social processes: they are likely to be informal and reciprocal rather than formal and negotiated. Another important point is also the problem of the division of labor: as economic activity becomes mainly intellectual and requires the integration of specific and idiosyncratic skills, the task of dividing the job and assigning it to the most appropriate individuals becomes arduous, a "supervisory problem" (Hogdson, 1999 [44]) emerges and traditional hierarchical control may result increasingly ineffective. Not only specificity of know how makes it awkward to monitor the execution of tasks, more importantly, top-down integration of skills may be difficult because 'the nominal supervisors will not know the best way of doing the job – or even the precise purpose of the specialist job itself – and the worker will know better' (Hogdson,

1999 [44]). We, therefore, expect that the organization of the economic activity of specialists should be, at least partially, self-organized.

The aim of this thesis is to bridge studies from computer science and in particular from Peer-to-Peer Networks (P2P) to organization theories. We think that the P2P paradigm well fits with organization problems related to all those situation in which a central authority is not possible. We believe that P2P Networks show a number of characteristics similar to firms working in a knowledge-based economy and hence that the methodology used for studying P2P Networks can be applied to organization studies.

Three are the main characteristics we think P2P have in common with firms involved in knowledge economy:

- Decentralization: in a pure P2P system every peer is an equal participant, there is no central authority governing the actions of the single peers;
- Cost of ownership: P2P computing implies shared ownership reducing the cost of owing the systems and the content, and the cost of maintaining them;
- Self-Organization: it refers to the process in a system leading to the emergence of global order within the system without the presence of another system dictating this order.

These characteristics are present also in the kind of firm that we try to address and that' why we have shifted the techniques we adopted for studies in computer science (Marcozzi et al., 2005 [59] Hales et al., 2007 [39]) to management science.

Contents

List of Figures

Chapter 1

Introduction

In this chapter we give a brief definition of what is the context and the subject of our research and what are the questions that we think are interestingly related to them.

1.1 Research Context: the Knowledge Based Economy

Knowledge-based economy is a term that refers to the use of knowledge for the production of economic benefits. Firms give great reliance on intellectual capabilities than on physical material and the final product or service is often something which is a mix of different expertise held by different individuals. This kind of economy is also characterized by knowledge intensity, technology and economic processes intertwined with social processes, all things that make traditional forms of organization based on hierarchy or market increasingly ineffective.

Some scholars argue that today's global economy is in transition from the classical industrial economy to the knowledge-based economy and that this transition requires that the rules that determined success in the industrial economy need rewriting in an interconnected, globalized economy where knowledge resources such as know-how, expertise and creativity are as critical as other economic resources (Cascio and Aguinis, 2008 [16]).

1.2 The Task Allocation problem

Now that we have briefly introduced the context of our research, we will focus on the subject of this thesis which is the "Task Allocation" problem. The problem of task allocation in a firm refers to the necessity of assigning resources (human of natural) to particular activities which must be done in the firm, maximizing the number of task executed and minimizing the costs related to this allocation. The process of task allocation is determined by two components: task components and team components. Task components refers to the complexity of the task like the number of skills that are necessary for the completion of the task, the interdependence between the different parts of the task and the way in which task change over time (task variety) (Wood, 1986 [100]). Team components refers to the attribute of the team members such as expertise and motivation (Wilke and Meertens, 1994 [96]) and social components like coordination (Wilke and Meertens, 1994 [96]). Self-Organizing processes of task allocation have been studied within biology, for instances how ants allocate tasks on the basis on local interactions (Gordon, 2001 [36]). This problem has also been studied in the field on computer science, for example methods for resource allocation in optimizing microcode compilers (Mueller et al, 1984 [71]) or resource allocation in service oriented grids (Batista and Fonseca, 2007 [4]) or methods for the allocation of mobile code bringing ideas from managerial sciences (Bredin et al., 1998 [9]). In managerial sciences, the problem of task allocation studied through the adoption of self-managing teams has already been studied by Zoethout (2006 [103]). He developed a computer simulation model and one on the main aims of his work was to study the influence of task variety on the performances of two groups in the system: *specialists* and *generalists*. He found that although no variety leads to specialization and high variety leads to generalization, in general performance is better when variety is low. Moreover, in case of no variety, specialists outperforms generalists. In our work we aim at studying the problem of task allocation in a situation in which jobs are intellectual and vary rapidly (high dynamism). For this purpose we designed a simulation model with which we test different mechanisms for coordinating the diverse skills involved in the firm (we describe the model in details in chapter 4).

1.3 Research Questions

As firms' job becomes knowledge oriented and know-how and expertise play an increasingly critical role in the production of the final goods, the problem of coordinating employees bearing specific skills becomes arduous. A firm may increases its coordination flexibility either by recruiting employees which will be deployed and redeployed to different task over the time, or hiring temporary workers which will be involved temporarily on some specific task. The former method is certainly sustainable by a large firm using a central hierarchy as organization form, while it is not applicable in firms where hierarchy is not possible. It is arduous for a large firm with large groups of specialists having different interests to achieve a collective good: members may be tempted to benefit from the common good without contributing when the gain for collaborating is less than the cost. Glance and Huberman (1994 [35]) showed that fluid organizations display levels of cooperation that are higher than those found in groups that either are unorganized or have a fixed social structure and although their results are based on the assumption of hierarchical structures inside the groups, they can be generalized to less constrained forms of social networks. Moreover, Cascio and Aguinis (2008 [16]) argues that the twenty-first-century organization is undergoing radical transformation mainly due to the spread of new technologies, particularly the Internet and that this heavy dependence on technology, the reliance on “good” intellectual capital and the sourcing and retaining talent at various organization levels from global labor market, are some of the main aspects characterizing the new organization. Even though not all organizations display such characteristics, one of the fundamental question that they rise as a group is how should firms go about recruiting workers.

In this thesis we present a simulation model called FirmNet with which we try to better understand the dynamics occurring in a knowledge-based economy and which is the best form of organization that can help deploying skills. With our model (see chapter 4), basically we have tested two different forms of organization: “hierarchy” and “quasi-market”. The firm is structured as a network of agents. In the model representing hierarchy, tasks are organized and coordinated through the authority of the project managers which have the ability to select specialists to put in some team basing the choice on their skill. In the model representing a quasi-market, which we called self-organizing (SO), the firm have a number of project managers holding a task to be completed; both managers and specialists move in the network auto-forming teams on the basis of some criteria which we will see in the next chapters.

In both models we designed the task as a bundle composed of three jobs labeled by a number (see figure 4.1). Due to this simplification, the project manager agents actually know which are the skills needed. In a knowledge-based economy it is not always possible for managers to exactly know which are the needed skills for a certain task; furthermore, once a certain specialist has been recruited, managers cannot always verify if the skill is the appropriate one or if the specialist is doing a good work (Hodgson, 1999 [44]).

Based on this, the research questions that we pose are:

- When hierarchical means of coordination may result ineffective, how firms coordinate the dynamic deployment of skills?
- What influences individuals to join specific groups? Is it based on the wealth or prestige of the group, on the ability (skill) required by the group, on the dimension of the group or on a mix of some of these things?
- How can the Network’s structure influence the firm’s performance?

Some hypotheses concerning these questions have been produced with the Firm-Net model (chapter 4).

1.4 Thesis Plan

We have introduced the "knowledge economy" indicating its characteristics. We have seen that traditional form of organization based on hierarchy or market, are increasingly ineffective. Our aim is to crumble the hierarchy moving from a vertical form of organization to a horizontal one. We attempt to bring ideas from computer science and in particular from P2P networks to experiment new plausible forms of organization. The chapters of this thesis are organized as follows.

In chapter two we first propose the review of some literature related to organization theory and in particular we indicate the characteristics and problems of the organizations working in the knowledge-based economy; then we overview some literature related to P2P networks and P2P computing, plus an introduction on the concept of "Peer-production"; finally we make a synthesis of the first two sections indicating which are the concepts of the P2P which can be used for organization purposes and we propose some kind of hybrids that will be discussed over the whole thesis.

In chapter three we describe the methodology adopted for the research we conducted giving an overview of some literature related to social simulation and agent-based modeling.

In chapter four we start introducing the basics of our conceptual model on which our simulation experiments are based. In particular we describe the technical details of the basic model and we indicate the different versions we developed.

In chapter five we start with the first work we did with our model: here we compare a hierarchical form of organization with a self-organizing one. We first describe the model and the experiments we have done, then we give some results an comments.

In chapter six we study more in details the self-organizing model proposed in chapter five. Here we propose six different version of the rewiring mechanism that characterizes such model. We also propose some analysis on the structure of the network.

In chapter seven we introduce the concept of learning to our model. Here we give nodes the ability to learn skills and eventually to pass jobs. We also test two different reward policies: “individual reward” and “group reward”.

Finally in chapter eight we conclude our discussion highlighting the crucial points of our work and delineating some possible future work.

Chapter 2

Literature Review

In this chapter we first propose the review of some literature related to the characteristics and problems of the organizations working in the knowledge-based economy; then we overview some literature related to P2P networks and P2P computing, plus an introduction on the concept of "Peer-production"; finally we make a synthesis of the first two sections indicating which are the concepts of the P2P which can be used for organization purposes and we propose some kind of hybrids that will be discussed over the whole thesis.

2.1 The Knowledge-Based Economy

When we talk about knowledge economy, we generally mean the concept of "Knowledge-Based Economy" indicating the use of knowledge and technologies to produce economic benefits. Hence knowledge is both tool and raw material (people's know-how, skill) for producing some kind of product or service. In this kind of environment economic organization is undergoing several changes. For example authority relations are less important, legal and ownership-based definitions of the boundaries of the firm are becoming irrelevant and there are only few constraints on the set of coordination mechanisms. Hence what characterises a knowledge economy is the growing importance of human capital in productive processes (Foss, 2005 [31]) and the increasing knowledge intensity of jobs (Hodgson, 1999 [44]). Economic processes are also highly intertwined with social processes; they are likely to be informal and reciprocal rather than formal and negotiated. Another important point is also the problem of the division of labor: as economic activity becomes mainly intellectual and requires the integration of specific and idiosyncratic skills, the task of dividing the job and assigning it to the most appropriate individual becomes arduous, a "supervisory problem" (Hogdson, 1999 [44]) emerges and traditional hierarchical control may result increasingly ineffective. Not only specificity of know how makes it awkward to monitor the execution of tasks, more importantly, top-down integration of skills may be difficult because 'the nominal supervisors will not know the best way of doing the job – or even the precise purpose of the specialist job itself – and the worker will know better' (Hogdson, 1999 [44]). We, therefore, expect that the organization of the economic activity of specialists should be, at least partially, self-organized. Beginning with the studies of Leavitt (1951 [54]), the analysis of the distribution of information in problem solving has stimulated a large body of literature centered on the problem of comparing performances of hierarchical versus decentralized organizational structures. Skill integration could emerge bottom-up and organization theory ought to address how individual decision-making routines and incentives forge the emergence of self-organized adaptive structures.

In this scenario the classical types of organization like hierarchy and market may sometimes result ineffective and new ways of organizing and deploying skills are emerging. The so called "Peer-production" is very popular today and is giving many examples of how production based on cooperation among people on a given project, can give incredible results. Typical examples are the "Linux" [109] operating system and the free encyclopedia called "Wikipedia" [110]. Yochai Benkler (2006 [6]) argues that Peer-production refers to a phenomenon of large-scale cooperation among people on a given project or problem. What typifies "peer-production" is that it represents an alternative model of organizing people, alternative to firms and markets. Rather than responding to managerial commands, or to prices, peer producers use *social motivation* and *communication* to organize their efforts. Hence social motivations can be a powerful engine able to create good connections among people interested in certain problems, which can, in some scenarios, outperform the classical "vertical organization" approach.

2.1.1 Hierarchy and Market

For decades our common understanding of the organization of economic production has been that individuals order their productive activities either as employees in firms, following the directions of managers (hierarchy), or as individuals in markets, following price signals. Since we are in a knowledge economy, as long as the jobs become more knowledge-oriented, these classical form of organization may result ineffective.

Hierarchy uses authority to create and coordinate horizontal and vertical division of labor. Adler (2001 [1]) suggests that when specialized units are told to cooperate in task that typically encouter unanticipated problems requiring novel solutions (this is typical of the knowledge economy), the hierarchical form gives higher-level managers few levels with which to ensure that the participating unit will collaborate. By their nonroutine nature, such tasks cannot be preprogrammed, and the creative collaboration they require cannot be simply programmed. Hierarchy hence results weak and firms have to look for other models.

Knowledge is a "public good" and hence the market/price mode forces a trade-off between production and allocation. On the one hand, production of new knowledge would be optimized by establishing strong intellectual "property rights" that create incentives to generate knowledge. On the other hand, not only are such rights difficult to enforce, but more fundamentally, they block social optimal allocation. Adler (2001 [1]) argues that allocation of knowledge would be optimized by allowing free access because the marginal cost of supplying another consumer with the same knowledge is close to zero.

Knowledge-based jobs are also leading to increasingly incomplete employment contracts. It is not possible to state in a contract all the duties of an employer and an employee: this happens in particular when the job is an intellectual job and the "supervisory problem" emerges. When the job is very specific, the employer cannot check if the employee is actually working well or if he really has the required skill (perhaps he doesn't even know which are the required skills for a certain job); on the other hand the employee could be less protected due to the lacks in the employment contract. Benkler (2002 [7]) argues that where agents, effort or resources cannot be specified, they cannot be accurately priced or managed, he also states that human creativity is a difficult resource to specify for efficient contracting or management.

In such context "reciprocity" between employer and employee, becomes an alternative to authority: employer instead of using authority on employees, have to cooperate with them. This new form of organization is more similar to Peer-production.

Hodgson (1999 [44]) suggests that the lack of managerial control on knowledge-based jobs, especially when knowledge is tacit and cannot be codified impairs and bounds the appliance of traditional employment contracts. The nature of the contracts should evolve along with the evolution of the distribution of bargaining power.

2.2 Specialisation, Communication and Coordination

Grounding on Hayek's argument (1945 [41]) that information possessed by individuals can be used only with the active cooperation, Kim and Mauborgne (1998 [50]) highlight the challenge that firms face in knowledge-based activities to obtain active cooperation of different actors each holding a piece of information. Thus, human resource management and the nature of employment relationships are particularly sensitive topics when organizations need to coordinate a number of specialists bearing specific know how. The structure of incentives in which a professional is embedded forges his motivation to coordinate with other agents. If, on the one hand, firms provide a context in which, shared coding scheme, languages and norms make it easier to exchange knowledge; on the other hand, in their study in communities of practice, Brown and Duguid (2001 [8]) recommend that, to address inertia of knowledge, researchers should go beyond explanations that take the cultural unity of the firm for granted. Shared practices define differences in identities and demarcate the extent to which knowledge spreads. Knowledge leaks in the direction of shared practice but communities sharing practices may stand across different organizations. Specialists in different disciplines to the extent to which they embrace distinct practices, they define different communities with distinct identities and specific knowledge. Knowledge is sticky when it moves among different communities of practice. Thus, the more distinct are the disciplines of specialists within an organization and the more specific is their knowledge, the more difficult we expect to be for individuals to join different communities of practice, to assess each others abilities and to coordinate to complete tasks.

2.3 The Problem of Adaptation

The endeavour of coordinating skills within organizations becomes arduous when the environment is turbulent. As Wright and Snell (1998 [101]) suggest, firms may increase their coordination flexibility by either nurturing a wide repertoire of specialists with different skills, which are deployed and redeployed to different uses, or hiring temporary workers with specific skills to be used for a specific project. Brusoni, Prencipe and Pavitt (2001 [11]) suggest that firms in the knowledge economy may have an incentive to have an increasing number of specialists of technological disciplines in-house and Moch and Morse (1977 [65]), and Haveman (1993 [42]) propose that this is especially true, in turbulent environments, when in-house availability of diverse specialists, better helps to face change because specific skills provide access to cutting-edge knowledge and novel solutions to organizational problems. On the other hand, Matusik and Hill (1998 [63]) suggest that pressures to change tend to blur organizational boundaries and push firms to hire contingent workers to allow rapid inflow of public knowledge. Davis-Blake and Uzzi (1993 [22]) found both a negative correlation between jobs requiring firm-specific training and the use of temporary workers, and a negative correlation between size and the use of temporary workers. They explained their results with the hypothesis that large firms can reallocate employers within the organisation, as changes in the environment require the completion of novel tasks, rather than trying to hire a temporary worker that needs to be embedded in firm-specific work context. The problem, however, is to understand how firms should govern the reallocation of skills within the organization and the extent to which the problem of skill allocation ought to be decentralized.

2.4 Hierarchical and Decentralized Structures in Problem-Solving

An important body of literature in organizational studies has addressed the choice between hierarchical versus decentralized organizational structures in problem-solving. A first thread of contributions in this area sprung off from experiments conducted by Leavitt on the effects of different communication patterns on group performance (1951 [54]). Along similar lines, Cohen (1962 [18]), adopting the same experimental setting used by Leavitt, analyzed how continued practice in two different communication structures effected problem-solving activity. Cohen compared, the wheel, the more hierarchical structure, within which all agents exchange information with a central agent, with the circle, in which each agent communicates with neighbours, and found that wheels took shorter times and made fewer errors. In general, this tradition of studies was focused on the differences, not only in efficiency, but also in satisfaction and in organization produced in different communication networks. For example, in a later study, Cohen et al. (1969 [21]) found that once centralized groups are embedded within larger and complex organizations, they perform poorer than decentralized structures because members react against constraints of centralized structures and create links to members external to the subgroup thereby employing more messages and time for longer routings of answers. Cohen et al. (1962 [19]) had also demonstrated that circle group that had prior experience in less satisfying centralized structure, such as wheels, were more satisfied and performed better than those groups that had been circle throughout. Another perspective centred on the structure of information and communication flow is to associate appropriateness of a decision-making structure to a particular kind of problem or task environment. In this light, Sah and Stiglitz (1986 [83]) confronted performance of hierarchies and polyarchies in decision-making by considering probability with which a good project is likely to be approved in an organization. They build their argument upon the relative advantages of systems of successive filters provided by screening mechanisms in hierarchies and the higher number of chances given to projects in less hierarchical decision-making structure such as polyarchies. In their study, polyarchies are preferred if prevalence of good projects is expected whereas adoption of hierarchies should shelter organizations when quality of projects is uncertain. On the other hand, the filters imposed to information flow may inhibit innovation in hierarchies (Burns and Stalker, 1961 [13]). Malone (1987 [56]) compares production, coordination and vulnerability costs of organizational structures characterized by different degrees of decentralization. He suggests that decentralized structures, such as decentralized markets, in which all managers interact directly with all task processors, have large coordination costs but low vulnerability costs, which occur when by an organization in adapting to a new situation. Thus, it is suggested that decentralized market structures are preferable in turbulent task environments. However, given the variety of dimensions that may characterize task environments and decision-making contexts, relevant literature offers a repertoire of different suggestions concerning the selection of hierarchical and decentralized structures. More recently, for example, Carley (1992 [14]), considering hierarchies and teams, these latter representing decentralized organizational structures without chain of command, suggested that teams learn faster and outperform hierarchies in problem-solving but hierarchies' performances depend less on turnover of employees, on time pressures (Lin and Carley, 1997 [55]) and biases in task environment (Carley and Lin, 1997 [15]), and Masuch and LaPotin (1989 [62]), articulating the garbage can metaphor (Cohen, March and Olsen, 1972 [20]), propose that hierarchy improve organizational performances only when commitments of organizational members is low.

2.4.1 Teamwork and decentralized decision making

An area of literature in organizational theory has specifically addressed teams as the solution of decentralization problem in complex decision-making (Marschak and Radner, 1972 [61]). A number of studies have explored the relationship between team structure and performances (Hoegl and Gemuenden, 2001 [43]). These studies have investigated, for example, the relationship between intra-team incentives and teams productivity (Dickinson and Isaac, 1998 [24]; Natter et al. 2001 [72]), the problems of coordinating expertise of specialists within a team (Faraj and Sproull, 2000 [29]) and the extent to which a team ought to be self-managed (Kirkman and Rosen, 1999 [51]; Wageman, 2001 [93]). The view that we propose in this thesis, however, is less concerned with the internal structure of teams than with the nature of team formation processes and the mechanisms through which team structure adapt to changing needs of a dynamic environment. Organization may use authority to assign skills to tasks but, as the complexity of task and the specialization of expertise involved increases, along with the size of the organization, rationality of decision-makers may be bounded in efficiently assigning skills to tasks. Contributions that address dynamics of team formation are mainly in the area of game theoretic approach to coalition formation (Hart and Kurz, 1983 [40]; Rapoport and Kahan, 1984 [76]; Seidmann and Winter, 1998 [86]). These contributions, however, do not consider the role that behavioural decision-making rules, such as *imitation* of successful colleagues, and position within a network of professional relationships have in moulding the structure of emergent teams. Decision to join a team may depend on our expectations concerning the effect that joining a particular team has on own reward but, within organizations, patterns of interactions influence individual decision-making (Ibarra, 1992 [52]) and expectations are formed on information exchanged with colleagues that are part of my neighborhood in, formal and informal, organizational networks. In this respect, the effectiveness of an organization in executing a variety of tasks, requiring different skills, may depend on the structure and the dynamics of the inner network embedding professionals holding specific know how. Along these lines, Glance and Huberman (1994 [35]) address the dynamics of intraorganizational network evolution and describe advantages of teams characterized by fluid boundaries in organizing collective action towards cooperation.

2.5 Staffing the Twenty-first-century Organizations

The nowadays organization, is facing major changes. Cascio and Aguinis (2008 [16]) argues that the twenty-first-century organization is undergoing radical transformation mainly due to the spread of new technologies, particularly the Internet, and that this transformation is nothing less than a new industrial revolution. Everyone in the organization can access informations instantaneously and more and more people are using mobile technologies to work on the go or at home. This means that the new organization must adapt itself to management via Web and must be devoted to constant change: organized around networks, not rigid hierarchy. Intellectual capital will be critical to business success. For example the advantage of bringing breakthrough products to market first, will be shorter than ever because technology will let competitors match them almost instantly. Thus firms must endow with best talents. In their paper, Cascio and Aguinis (2008 [16]), argue that the current staffing methods are ineffective; they believe that improvement in the ability to forecast job performances lies in more careful specification of the domain of performances, together with increased effort to demonstrate correspondence between predictors and the wide range of elements of the performances domain. They call this performances domain "*in situ* performance": it is the specification of the broad range of effects – situational, contextual, strategic and environmental – that may affect individual, team, or organizational performances. The current staffing model is too simplistic, with it, context rarely matters, performances are considered isolated from context and it has not shown major improvements in its ability to predict performances. The staffing model they propose, involves an expanded view of the predictors of performaces. They propose two guiding principles for the development of the new staffing method.

The first is that predictors of *in situ* performances should consider the preminent role of time. If we consider group works, we know that most of of them function over an extended period of time. Unfortunately, much research on staffing in group works has focused on short-term group within a narrow time frame (McGrath et al, 2000 [64]; Williams and O'Reilly, 1998 [97]). McGrath et al. (2000 [64]) noted that groups are inherently dynamic systems operating via processes that unforld over time, with those processes dependent both on group's past history and in its anticipated future. Hence measurement that takes place over short periods of time, as in the current staffing method, is likely to result in thin slice of behavior that are not representative. Measurement should take place over extended periods of time.

A second guiding principle is related to the context. The selection environment must emulate the work environment. This principle involves creating a context with all of its complexity and situational, contextual, strategic and environmental constraints.

What Cascio and Aguinis (2008 [16]) propose with their paper, is a novel staffing model in which the measurement of present *in situ* performances should predict the measurement of future *in situ* performances. This is because the current staffing model uses individual knowledge, skills, ability and other individual characteristics assumed to underly certain predictors to make predictions about future jobs performances that is also assumed to be determined by the same characteristics.

2.6 Peer-to-Peer Networks

Open peer-to-peer (P2P) overlay networks have become very popular for file sharing applications and they have emerged as an important paradigm for distributed computing, due to their potential for the involvement of millions of peers in the process of sharing and collaboration. They organize a large set of participants (peers), in a logical network on top of a physical topology. Due to its good features, this paradigm is now applied to a wide spectrum of distributed applications. One of the most interesting features of P2P networks is their ability for direct resource sharing among dynamic, decentralized client peers. Scalability is also central in P2P applications and it relies both on an even distribution of the load between peers and the ability to react to system dynamics.

The P2P approach differs from the client-server in that no distinction is made between client and server machines: all the nodes in the network are both client and server so they must both serve some resource or service and can ask for them. Between the main feature of P2P networks we find that they are totally decentralized so there is no central and trusted authority which controls the system. For a given application to provide a service, a subset of nodes may be elected to act as servers, to which nodes can forward requests. When designing this kind of system it is possible to define local rules to be followed by each peer and the formation of global behavior on large scale should only appear as an *emergent behavior* obtained from local interactions between peers.

Three main classes of P2P applications have emerged: parallelizable, content and file management, and collaborative.

Parallelizable. Parallelizable P2P applications split large task into smaller sub-pieces that can execute in parallel over a number of independent peer nodes. Most implementations of this model have focused on compute-intensive applications. The general idea behind these applications is that idle cycles from any computer connected to the Internet can be leverage to solve difficult problems that require extreme amounts of computation. Most often, the same task is performed on each peer using different sets of parameters. Examples of implementations include searching for extraterrestrial life (SETI@Home, 2001 [106]), studying computational ways to design new anti-HIV drugs based on molecular structure (FightAIDS@Home [107]), developing more accurate climate models of specific regions in Africa (AfricanClimate@Home [108]).

Content and file management. Content and file management P2P applications focus on storing information on and retrieving information from various peers in the network. The model that popularized this class of application is the content exchange model. Applications like Napster and Gnutella (Ripeanu et al., 2002 [78]) allow peers to search for and download files, primarily music files, that other peers have made available. They focus on using otherwise unused storage space as a server for users.

Collaborative. Collaborative P2P applications allow users to collaborate, in real time, without relying on a central server to collect and relay information. Instant messaging is one subclass of this class of application. Services such as Yahoo! ([111]), Skype ([113]), and Jabber ([112]) instant messaging have become popular among a wide variety of computer users (Strom, 2001 [90]). Similarly, shared applications that allow people (e.g., business colleagues) to interact while viewing and editing the same information simultaneously, yet possibly thousands of miles apart, are also emerging. Examples include distributed Power Point (Rice and Mahon, 2000 [77]). Games are a final type of collaborative P2P application. P2P games are hosted on all peer computers and updates are distributed to all peers without requiring a central server.

2.6.1 P2P characteristics

Peer-to-peer systems have several interesting characteristics which we believe can be found also in new forms of organization like Peer-production:

- Decentralization: In a fully decentralized system, every peer is an equal participant. This makes the implementation of the P2P models difficult in practice because there is no centralized server with a global view of all the peers in the network or the files they provide. This is the reason why many P2P file systems are built as hybrid approaches as in the case of Napster [114], where there is a centralized directory of the files but the nodes download files directly from their peers. One way to categorize the autonomy of a P2P system is through the pure P2P versus hybrid P2P distinction. This categorization has a direct effect on the self-organization and scalability of a system, as the purest systems are loosely coupled to any infrastructure.

- Scalability: An immediate benefit of decentralization is improved scalability. Scalability is limited by factors such as the amount of centralized operations (e.g, synchronization and coordination) that needs to be performed. Recent P2P systems, represented by CAN, Chord, PAST and Pastry (Stoica et al, 2001 [89] and Rowstron, 2001 [82]), dictate a consistent mapping between an object key and hosting node. Therefore, an object can always be retrieved as long as the hosting nodes can be reached. Nodes in these systems compose an overlay network. Each node only maintains information about a small number of other nodes in the system. This limits the amount of state that needs to be maintained, and hence increases scalability.

- Cost of ownership: One of the premises of P2P computing is shared ownership. Shared ownership reduces the cost of owing the systems and the content, and the cost of maintaining them. This is applicable to all classes of P2P systems. Also the elimination of centralized computers for storing information also provides reduced ownership and maintenance costs.

- Performance: P2P systems aim to improve performance by aggregating distributed storage capacity and computing cycles of devices spread across a network. Because of the decentralized nature of these models, performance is influenced by three types of resources: processing, storage, and networking. In particular, networking delays can be significant in wide area networks.

- Fault-resiliance: One of the primary design goals of a P2P system is to avoid a central point of failure. Although most P2P systems (pure P2P) already do this, they nevertheless are faced with failures commonly associated with systems spanning multiple hosts and networks: disconnections /unreachability, partitions, and node failures. Pure P2P systems without centralized control are able to resist to these problems.

- Self-organization: In P2P systems, self-organization is needed because of scalability, fault resilience, intermittent connection of resources, and the cost of ownership. P2P systems can scale unpredictably in terms of the number of systems, number of users, and the load. It is very hard to predict any one of them, requiring frequent re-configuration of centralized system. There are a number of academic systems and products that address self-organization. In Pastry, self-organization is handled through protocols for node arrivals and departures based on a fault-tolerant overlay network (Druschel and Rowstron, 2001 [26]). Client requests are guaranteed to be routed in less than $\log_{16} N$ steps on average. Also, file replicas are distributed and storage is randomizes for load balancing.

2.6.2 Peer-production

Now that we have seen the main characteristics of P2P systems, we introduce a novel form of organization which takes inspiration from P2P. It is called "Peer-production" (Benkler 2002 [7]). By "Peer-production", we indicate a form of production of goods and services entirely based on auto-organized communities of voluntary individuals with the aim to pursue a common goal. It takes inspiration from the more famous concept of peer-to-peer networks (just described above) in which all the nodes in the network are peer and hence have the same power. For example the idea behind some P2P applications like SETI@Home (see previous section), is the same that is behind the Linux project: in both cases a "big task" is split into many "small subtasks" which are executed by many voluntaries using their own resources (CPU power in the first case and programming skills in the second case).

Actually Peer-production is often a mix of hierarchy and self-organization where expert members of the community give the guidelines for the production. Generally in these communities individuals work for free and they generally do this for social motivations like becoming popular or passion for a certain topic. Typical example of Peer-production are given by "Linux" [109] and "Wikipedia" [110]. Of course this kind of production cannot be applied on every kind of product of service. It's important that individuals working on a certain project can do this in a small time and with a limited effort of resources. This is why it is generally applied to knowledge-based services or goods for which common tools are enough. Anybody can contribute to Wikipedia spending their free time at home, they just need a PC and an internet connection.

It is also important that tasks can easily be divided into little pieces such that many people can work on them in a reasonably small time. Nowadays billions of people around the world can contribute and cooperate to realize any kind of product which requires *creativity* a *personal computer* and *an internet connection*. Now production costs, for certain kind of activities, are very low and anybody can produce or exchange several kind of goods or services (like newspapers) without the need of market-based or firm-based models (Tapscott and Williams, 2007 [91]).

This may sound like a threat for companies but in some cases it can be an opportunity for firms ready to take advantage of such creative power constituted by millions of enthusiastic people in their business areas.

The main characteristics of the products and services realized through peer production are (Benkler, 2002 [7]):

- no hierarchy, no market: the quantity of the goods realized with peer production cannot be decided neither by an authority (as in the hierarchy), nor by price (as in the market). It is decided by the spontaneous encounter between tasks and skills;
- individuals involved in a peer community must hold the production means (the same happens in the market);
- the final product is a "common good" with a public open license.

These characteristics are perfectly respected by the Open Source communities. In recent year we have seen how the main benefit of the peer production have been given to the Open source community. Von Krogh and von Hippel (2003 [92]) have shown how these communities have benefited greatly from the advent of the Internet, which has enabled members to interact and share resources extensively. The most famous example is the Linux operating system which has gained lot of popularity through the years and millions of buyers and users worldwide.

However for such kind of projects the main ingredients are the creativity and the passion of the thousand of people daily working on them. Many are the reasons why they decide to contribute for free, as make experience on important projects and establish contacts which other developers. On the other hand there are firms like IBM and Intel having employee working on open source projects, meaning that not everybody is really involved for free.

Three are the main conditions that must be respected to allow peer production (Tapscott and Williams, 2007 [91]):

- the object to be produced must be information-oriented;
- tasks must be divisible into small independent parts such that individuals can participate contributing with minimal time effort;
- costs relating the integrations of the produced parts and for their reviews must be minimal.

Beside of this, an other important aspect of the peer production is that communities must be endowed of a system for *peer reviews* for controlling the quality of the product and *leaders* able to govern and handle the interactions between members and merge heterogenous contents produced by them.

Peer production hence, provides a framework, within which individuals who have the best information available about their own fit for a task can self-identify for the task (Benkler 2002 [7]). This provides an information gain over firms and markets, but only if the system develops some mechanism to filter out mistaken judgments agents make about themselves. This is why practically all successful peer production systems have a robust mechanism for peer review or statistical weeding out of contributions from agents who misjudge themselves.

We believe that the mechanisms which are behind the peer production can be derived from the mechanisms behind P2P networks. Both firms and P2P networks are complex adaptive systems (CAS) and we think that common methods for organizing both exists.

2.7 Network Organization

Previously we have indicated how the increasingly knowledge intensity of jobs and high dynamism of task require a more fluid form of organization. Within a firm it is necessary that employees can encounter, cooperate and share their skills in order to achieve some particular task. In order to permit this, an infrastructure that allows individual to find each other is necessary. In large firms often happens that the knowledge is already in the firm but the head of the unit of the firm that needs such knowledge or skill doesn't know about that. That's why some kind of infrastructure is needed for making tasks match with skills; firms need to recreate a situation similar to what happen with peer-production (Tapscott and Williams, 2006 [91]). Since relationships between peoples can be considered as links in a network and since in the knowledge economy a central authority that leads the individuals is not always possible, the structure of a P2P network seems to fit appropriately.

Network Organization is one of the names that have been coined to indicate a kind of organization process that do not conform to traditional definition of markets or hierarchies. In the early 1980's a dichotomous view of economic organization saw market on one hand and hierarchy on the other; such dichotomous view was quiet explicit in the transaction cost economics. Oliver Williamson (1991 [99]) asserted that the alternative to pure markets and hierarchies can be interpreted as intermediate or hybrid forms combining elements of markets and hierarchies and also that pure types tend to prevail on mixed forms (Williamson, 1985 [98]). After this, lot of sociological research on these two point was done, trying to understand if network organization can be consider a mix of the two just cited approaches, or something different. Podolny and Page (1998, [80]) argues that from a purely structural perspective every form of organization is a network and markets and hierarchies are two manifestations of the broad type. When considered as a form of governance, instead, the network form can be characterized as a collection of actors that pursue repeated, enduring exchange relations with one another lacking a legitimate organizational authority that can arbitrate and resolve disputes that arise among actors (as in hierarchies). In a pure market, relations are episodic and ending after the transaction.

Although this definition of network organization (Podolny and Page, 1998 [80]) excludes employment relations, a number of scholars have argued that network form of organization can be characterized by a "distinct ethic". The buyer tries to work with the seller to address the bugs in the sellers performances instead of moving to another seller and Powell (1990 [81]) argues that reciprocity is a guiding principle underlying network organization. Something similar happens nowadays with Peer-production principle: what Peer-production does (Benkler, 2002 [7]) is to provide a framework within which individuals who have the best information available about their own fit for a task, can self-identify for the task. In a knowledge economy context, this gives advantages to Peer-production respect to hierarchies and markets but implies a mechanism for "peer-review" for individuals who misjudge themselves.

It's interesting to note how some of the P2P characteristics that we have seen in the previous section can be found also in the network organization. Self-organization for example is fundamental in those contexts where an authority (manager) able to allocate tasks to employees is not present or is not possible: in this case specialists (who better know what they are able to do) must be able to continusly adapt the structure of the firm in order to cope with dynamic tasks.

Previous work have shown that protocol previously designed for P2P networks can be applied also for social systems (Marcozzi and Hales, 2008 [60]), so in this work we try to go ahed modeling the firm as a P2P network in which specialists are the nodes and the relations between them are the links.

2.8 Complex Systems

Many different phenomena fall into the definition of *complex system*, all have in common some defining key aspects, even though it is not easy to give a formal definition of what a complex system is, complex systems are usually defined as a set of interconnected simple entities that are able to produce some kind of higher level properties.

Such properties, exhibiting global features even though they are obtained through simple local interactions, are usually referred to as *emergent properties.* Since complex system are usually formed by a large number of simple entities and since interactions among such entities are often guided, at least in part, by stochastic decisions, in general it is very hard to predict the behavior of a complex system, hence to explicitly guide it to the desired emergent properties.

An important characteristic of complex system is given by *adaptivity.* Typically, complex systems show high degree of adaptation to environmental changes. Intuitively, being formed by a large number of interconnected simple entities and showing very low hierarchical structures (even no hierarchy at all), complex systems are robust to the loss of some of their basic components . Moreover adaptivity is often achieved thanks to the system's nature itself and through the propagation on a global scale of small local reactions to environmental changes.

A big help in understanding how complex systems work, and a great source of inspiration to define new design techniques and models as well, is given by the large number of examples of complex systems present in nature. Typical examples are those of social insects (many "stupid" ants building large and structured nests), immune system (many immune cells adapting to various antigens and their mutations), or even human artifacts, as stock market or even human society as a whole!

P2P networks and firms clearly fall into the definition of complex systems, this explains why a trend to draw inspiration from complex system studied and analyzed in different disciplines such as biology and sociology has been growing stronger and stronger and has led to a plethora of *bio-inspired* and *socio-inspired* mechanism to solve various problems in P2P networks (Jelasity et al., 2005 [47, 48]; Babaoglu et al., 2005 [5]).

Chapter 3

Methodology: Computer Simulation in social and managerial sciences

Nowadays, computer simulation is becoming very popular and is gaining acceptance from the community of social and economic scientists. In this chapter we try to understand what value a computer simulation model can add to a research design: we first give an overview of computer simulation in social sciences, the we discuss on how economy can be considered Complex Adaptive System, finally we describe the *roadmap* we followed to develop our research.

3.1 Simulation in social science

Computer simulation introduces a new way of thinking about social and economic processes, related to the emergence of complex behavior from the interaction of actors performing relatively simple actions (Simon, 1996 [87]). This concept is generally indicated as "complexity theory" which is becoming very popular in several disciplines as physics, biology and computer science. It is an increasingly significant methodological approach to theory development in the literature focused on strategy and organizations, indeed several important research (Coehen et al., 1972 [20]; March, 1991 [57]; Burgelman and Mittman, 1994 [12]; Gavetti et al., 2005 [33]) have used simulation as their primary method. Simulation is a particular type of modeling: a model is built which is a simplification of a some structure or system.

This simplification is less detailed and less complex than the original structure. After the model is realized, experiments with it are executed: simulation involves that the researcher using it gives *inputs* and, while the simulation runs, observes *outputs*. In general, a simulation experiment entails the formalization of a theory, thus a computer simulation can be used as a theoretical laboratory to manipulate a deduction process and to explore emerging behavior of complex systems when analytical tools cannot cope with the complexity of a system. Davis et al. (2007 [23]) propose that simulation can be adopted for "theory development" when *simple theory* exists (Rudolph and Repenning, 2002 [79]). By simple theory is meant undeveloped theory that has only a few constructs and related propositions with modest empirical or analytical grounding such that the propositions are in all likelihood correct but are currently limited by weak conceptualization of constructs and/or rough underlying theoretical logic. Simple theory also includes basic processes that may be known like competition, imitation (this is our case), but have interactions that are only vaguely understood. Thus, simple theory contrasts with well-developed theory. From these perspectives, computer simulation can be a powerful method for sharply specifying and extending extant theory in useful ways.

In addition to theory development, computer simulation can have several uses; one can be that of observing a particular phenomenon. For example we can make a model representing laborers working in a steel mill to obtain a better understanding of their behaviors. Another interesting use of simulation is for prediction: we can develop a model that faithfully reproduces the dynamics of some behavior during the passing of time and thus use the model to predict what will happen (Gilbert and Troitzsch, 2005 [34]). A third use of simulation is to develop tools to substitute human capabilities as the building of "expert systems" that can be used by non-expert to carry out diagnoses which would otherwise require human experts. In addition, simulation can clearly reveal the outcomes of the interactions among multiple underlying organizational and strategic processes, especially as they unfold over time (Repenning, 2002 [75]).

There are several possible approaches for computer simulation in social sciences;

the most famous are: System Dynamics (SD), Agent-Based Modeling (ABM). System dynamics, which is connected to the work of Forrester (1961 [30]), is a powerful methodology and computer simulation modeling technique for framing, understanding, and discussing complex issues and problems. The basis of the method is the recognition that the structure of any system – the many circular, interlocking, sometimes time-delayed relationships among its components – is often just as important in determining its behavior as the individual components themselves. It deals with internal feedback loops and time delays that affect the behavior and the structure of the entire system. Agent-Based modeling, simulates actions and interactions of autonomous individual entities and build on the hypothesis that the behaviour of social systems can be modelled and understood as evolving out of interacting but autonomous learning agents (Epstein and Axtell, 1996 [28]; Axelrod, 1997 [2]; Axtell, 1999 [3]).

Independently of the approach adopted, research work employing computer simulation has frequently been regarded, in social sciences, as influenced by an autonomous logic in respect to mainstream research. Interestingly, computer simulation has not always gained lot of success from researchers: some of them suggest that simulations are only "toy models" of actual phenomena, in that they either replicate the obvious or strip away so much realism that they are simply too inaccurate to yeld valid theoretical insights (Chattoe, 1998 [17]). Two are the main consideration on which those researchers base their perplexities (Mollona, 2008 [70]). First, very often, the system of symbols, whose behavior is simulated by computers, is not entirely represented as a mathematical model but takes the appearance of a number of strings of programming code; these strings, which may be very numerous, embed the algorithms that both describe the social behaviours under study and a number of rules that direct the computer in executing the code. This program may be hard to communicate to readers who are not necessarily skilled in programming. Thus, replicating experiments can be a problem. Second consideration is that a computer simulation produces results by the means of numerical rather than analytical solution. Obviously, this is true because often we use computer simulation for the very reason that the phenomenon under study is so complex that cannot receive an analytical treatment. The problem with numerical solutions is that each simulation produces a result that depends on the specific calibration of the parameters that we used for that simulation. We should run an infinite number of simulations (as infinite are the values that we could use to calibrate parameters) to obtain the entire possible repertoire of behaviours that a model could produce. This fact may induce to think that any conclusion extracted from a simulation experiment is of limited value.

Davis et al. (2007 [23]) suggests that the controversy surrounding the value of computer simulation for theory development, partially arises from a lack of clarity about the methods and its related link to theory development. They argue that there is a limited understanding about when to use simulation (when it is a useful methodology), how to select the most appropriate simulation approach, how to perform simulation experiments and which are the relevant criteria for evaluating simulation research. Most scholars (Dubin, 1976 [27]; Priem and Butler, 2001 [74]) agree that theory has four elements: constructs, propositions that link those constructs together, logical arguments that explain the underlying theoretical rationale for the proposition, and assumptions that define the scope of boundary conditions of the theory. Consistent with these views, Davis et al. (2007 [23]) define theory as consisting of constructs linking together by propositions that have an underlying, coherent logic and related assumptions. In light of this, they developed a *roadmap* for how to use simulation to develop theory (see section 3.4).

We believe that computer simulation helps rigorously to deduce consequences from modelled assumptions when complexity of modeling makes difficult to obtain closed-form solutions. In addition, simulation allows looking at unfolding organisational and social processes, capturing the behavioural characteristics in transitory states. This approach has the advantage of creating an appropriate setting to conduct controlled experiments. Thus, simulation studies assist the discernment among groundless assertions and assertions that are true only within certain boundaries and given specific assumptions, and help researchers to identify missing variables and to elicit hidden assumptions thereby supporting testing of internal consistency (Langley 1999, [53]), robustness and generality of a theory. In addition, simulation, by illustrating non-obvious implications of a theory, sets the grounds for the theory development. Hence, our ultimate aim is to develop hypotheses and theories that can then be applied to real world phenomena and data. We use the computer model at this stage to help us generate and test, in a rigorous and deductive way, candidate ideas.

3.2 The economy as a Complex Adaptive System

The economy in general is a system where a large number of agents interact and the interaction of these agents leads to the formation of complexity. That's why the economy may be considered as a complex adaptive system (CAS). The same applies to organization science, which studies the interaction of individuals or groups of individuals involved in some particular task within a company. As we know, economics deals with a microlevel and a macrolevel: Microeconomics takes as its starting point the behavior of individual agents whereas macroeconomics theorizes about relations between aggregate magnitudes. Bruun (2006 [10]) suggest that while traditionally macroeconomics and microeconomics where apparently impossible to combine, complexity science now offers a way out to this situation. Rather than starting with either a single isolated agent or aggregate magnitudes, complexity science suggests focusing on the interaction between agents. Recognizing that what turns large composite systems into systems and not just collections is the interaction between the parts, it seems apparent to start with the interaction. Schelling (1978 [85]) was among the first to apply a complexity approach to social sciences. He argued that economic systems are particularly complex because, besides a large number of locally interacting agents, economic systems are characterized by a lot of relations that must hold in the aggregate, but does not necessarily hold for each individual.

Hence we can argue that the economic system can be considered as CAS and that complexity arise from three factors:

- the economy is a large composite system;
- economic agents adapt their behavior to the system;
- economics is characterized by a lot of relations that must hold in the aggregate, but need not hold for the individual agent.

The same factors arise in organization systems where the type of interaction of lot of different agents determines the kind of organization.

Another interesting aspect of CAS is also that they are at least partially self-organized: when in chapter 2 we talked about Peer-production, we said that in that kind of organization, individuals must be able to self-elige for certain task. When the task is to big and no central authority is possible, individuals involved should give themselves a kind of organization structure according to their skills and their preferences. This ability to self-organize in a changing environment is an interesting characteristic of CAS.

Hence economic and organization systems work in a complex way and it is very difficult to produce models that perfectly reproduce them without doing some simplifications. That's why, done these simplifications, it is possible to make computer models able to represent and study the complexity of economic and organization systems both in the *micro* and in the *macro*, simulating the actions of the agents involved.

3.3 Agent-Based Computational Economics

Since economics can be seen as a complex adaptive system, it is very difficult to study the system analytically. Sometimes, doing experiments can also be too expensive or not feasible. Modeling and simulation complement the traditional empirical and experimental approaches to research since they provide effective ways for organizing existing data, focus experiments through hypothesis generation, identify critical areas where data are missing, and allow virtual experimentation when real experiments are impractical or just too expensive. Thus, Agent-Based Modeling (ABM) gives a big help in understanding the behavior of the system which arise from the interaction of a large number of agents. Basically researchers try to understand economic and organizational processes by synthetically reproducing them on a computer model. The model is generally built bottom-up starting from simple components assembled into a working system. This model is composed of a number of agents having some specific behavior and interacting, generating some complex dynamics in the system. No general assumptions are needed on what will happen to the global system, this will "emerge" after the interactions of the agents.

Several are the reasons why computer simulation is very used for these studies (Gilbert and Troitzsch, 2005 [34]):

- programming languages are more expressive and less abstract;
- programs deal more easily with parallel processes and processes without a well-defined order of actions;
- programs are often modular, so that major changes can be made in one part without the need to change other parts of the program;
- it is easy to build simulation systems that include heterogeneous agents for example, to simulate people with different perspectives on their social worlds, different stocks of knowledge, different capabilities and so on.

The ABM approach is able to capture types of complex, dynamic, interactive processes so important in the social world. One important characteristic of ABM, is the potential asynchrony of the interactions among agents and between agents and their environments. In ABM agents typically do not simultaneously perform actions at constant time-steps, their actions follow discrete-event cues or a sequential schedule of interaction cohabitation of agents with different environmental experiences. Each agent is a software program comprising both data and behavioral rules (processes) that act on this data. Thus, ABM represents dynamic systems in a manner permitting the systems to evolve over time through agent interactions, with a minimum of a-priori assumptions. Macroscopic system behaviors are then observed as emergent properties (emergent behavior). Moreover, the richness of detail one can take into account in ABM makes this methodology very appealing for the simulation of biological and social systems, where the behavior and the heterogeneity of the interacting components are not safely reducible to some stylized or simple mechanism.

As we said, the ABM methodology follows a bottom-up approach and focuses on the interaction between many heterogenous interacting agents, which might produce a statistical equilibrium rather than a natural one as the mainstream approach assumes. The bottom-up approach models individual behavior according to simple behavioural rules; agents are allowed to have local interaction and to change the individual rule (through adaptation) as well as the interaction nodes (Gallegati and Richiardi, 2008 [32]). In ABM, aggregate outcomes (the whole) are computed as the sum of individual characteristics (its parts). However, aggregate behavior can often be recognized as distinct from the behaviour of the comprising agents, leading to the discovery of emergent properties. In this sense, the whole is more than - and different from - the sum of its parts. It might even be the case that the whole appears to act as if it followed a distinct logic, with its own goals and means, as in the example of a cartel of firms that act in order to influence the market price of a good. From the outside, the "whole" appears not different from a new agent type (e.g. a family, a firm). A new entity is born; the computational experiment has been successful in "growing" artificial societies from the bottom up (Epstein and Axtell, 1996 [28]).

3.4 Theory Development with Simulation

Davis et al. (2007 [23]) proposed a roadmap for developing theories with simulation. In this section we indicate the points of such roadmap and then map our work on it.

3.4.1 Finding an intriguing research question

As we know, studies that develop theory should start with a good research question that reflects deep understanding of the extant literature and relates to a substantial theoretical issue (Weick, 1989 [95]). Research questions can originate from many sources. An example is given by March (1991, [57]) which relied on complexity theory from biological and computer sciences to conceptualize a research question that examined the trade-off between the exploration and the possibility of exploitations of old certainties. The main question related to our work, relates the studying of the more appropriate form of organization to adopt when hierarchical means of coordination may result ineffective as in the case of the knowledge-based economy.

3.4.2 Choose a Simulation Approach

Assumed that simulation is the best way to proceed for our research, the next step is to select the more appropriate simulation approach. In our studies we decided to adopt the ABM methodology. This choice is based on the fact that in our work we try to bridge the P2P network world to the organization theory world adopting some mechanism related to the organization of P2P network. Our idea is to model a firm operating in a knowledge-based economy, as a P2P network in which each node represents an individual working in the firm. Since the testbed we decided to adopt (Peersim) has been developed with ABM techniques, we used ABM also for the models we developed.

3.4.3 Create the computational representation

The computational representation involves *building algorithms* with programming languages, that captures the step-by-step theoretical logic underlying the simple theory. In other word, the software code should embody the theoretical logic. The algorithms should consist of a series of steps for modifying construct values in accordance with the underlying theoretical logic of the simple theory. The agents we described in our work, follow a simple step by step logic which is described in chapter 4.

3.4.4 Experiment to Build Novel Theory

Experimentation is at the heart of the value of simulation methods for developing theory. Effective experimentation builds new theory by revealing fresh theoretical relationships and novel theoretical logic. There are several approaches to effective experimentation. A common one is *varying the value* of constructs that were held constant in the initial simple theory. A second one can be *varying assumptions*; this is particularly used when fundamentally different processes may reasonably exist. Experiments focus in revealing their possible distinct effects. A third one is *adding new features* to the computational representation. Additional complexity is particularly useful when researchers want to explore the interactions of multiple processes that are well-known alone but not in combination and when greater realism is desired.

Some of these approaches have been used in our experimental settings which we propose in chapters 5, 6, 7.

3.4.5 Validate with Empirical Data

A final step is validation. It involves comparison of the obtained results with empirical data. If the results of the simulation match the empirical evidence, the simulation is validated for that empirical context. There is however some debate over the value of validation. Some scholars argue that the central purpose of theory development through simulation is creating interesting theory and so they diminish the value of validation (Weick, 1989 [95]). Davis et al. (2007 [23]) suggest that the importance of validation depends on the source of the simple theory. For example if the theory is based primarily on empirical evidences, then validation is less important because the theory already has some external validity. In contrast, if theory is based primarily on non-empirical argument or on evidence from other disciplines (e.g. computer science), then validation is more important.

Hence, relating to our work, validation with empirical data should be done. However the main intent of this work, was to develop hypothesis. We think that empirical validation should be subject for future work.

Chapter 4

The FirmNet Model

Modeling and simulation constitute a fundamental element of the research design. We used an agent-based model to simulate interaction among professionals holding specific expertise and project managers, which receive from clients tasks to be performed. In this chapter we describes the core of our simulation model and we indicate all the variants that we will propose in the next chapters.

4.1 Introduction to FirmNet

In our model firms are represented indirectly as a network of agents that receive tasks from clients and offer to professional a reward in exchange of their collaboration. The FirmNet model (Mollona and Marcozzi, 2008 [68], [66], [67]) should be viewed as an "artificial society" type model (i.e. similar to the SugarScape model of Epstein & Axtell, 1996 [28]) that allows to express formally (computationally) a number of hypotheses about potential processes that may occur in organizations but in a stylised and executable manner such that experiments can be performed to deduce the consequences of those hypotheses when they are combined in complex, adaptive systems (CAS). We therefore purposefully present a simplified model in which we hope to capture the kinds of complex dynamics in which we are interested. In the next sections we will describe the basic elements of our model.

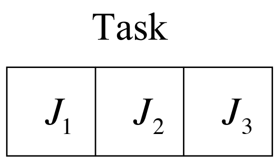


Figure 4.1: Structure of a Task. A task is bundle of three jobs. Each job requires a skill.

4.1.1 The Task

Lot of research has been done on the area of tasks and task allocation (Hunt, 1976 [45]; Steiner, 1972 [88]; Weick, 1979 [94]). In our work a "task" is an object which requires a set of skills to be performed. Theories indicate that a task can be split up into task actions. We modeled it as a bundle of three *jobs*, each requiring a particular skill (see figure 4.1). To complete a task of three jobs, three agents must provide their skill. Since we are in a knowledge economy, a skill refers to a particular knowledge or know-how of an agent. To each task is bounded a payoff which indicates the reward (R) that the agents involved in the completion of the task will share. If for example the reward for a certain task is 3 ($R = 3$), this means that R for each single job is $R_j = 1$ (we will see later how R_j is shared among the agents).

4.1.2 The Agents

In our simulation model, the firm is modeled as a P2P network. Each node of the network is an agent performing some specific action. We distinguish two kind of agents:

- **node-task agents** (NT): these agents play the role of *project managers*, have direct contacts with clients and receive a certain Task to be completed. They have the duty of aggregating employees able to work on the specific tasks. In addition they also have a personal skill which can be used for one of the jobs composing their task, or can be supplied to another NT as normal NS do. Each NT has a parameter α which indicates the percentage of R_j that the

NT is willing to share with the NS performing the single job. For example, if $R_j = 1$ and $\alpha = 0.3$, this means that NT is wiling to pay a maximum of 0.3 (this is calculated as $R_j \times \alpha = 0.3$) to the NS providing the skill, keeping the remaining 0.7 for itself. If a NT is linked to more than one NS with the same task, it will chose the one with lower β.

- **node-skill agents** (*NS*): these agents hold a certain skill (S) that they use to perform some job. They are considered *employees* of the firm. Each NS has a parameter β which indicates the minimum reward that the NS wants to accept a job (also called acceptance threshold). This parameter, can be considered as motivation: the lower it is, the more motivated the agent is. For example, $\beta = 0.4$ indicates that NS wants a minimum payoff of 0.4 in order to provide the skill; if NT offers 0.3, the skill will not be provided.

Both NT and NS have also a "busy" flag which is set to true when the node is using its skill for a certain job. When a node is busy, it cannot accept a call for a job from a NT and cannot move. After a node gets busy, it will be free again after the single job or after the entire task is completed (it depends by the model's version we are using); however, every nodes resets its busy flag to zero every time new tasks are assigned to NTs. The task flag T indicates if the node is a NT or a NS. Table 4.1 shows the node's state.

Hence in our basic model to complete a task, it is necessary to complete three jobs, each requiring a different skill and NT agents need to form a team attracting NS agents, which hold the required skill. Thus, the difference between NT and NS agents is that NT agents arbitrage, on behalf of the firm, the relationship between skills and clients. Hence, the model simulates an organizational network in which teams arise having certain skills.

Table 4.1: Node's state

Parameter	**Value**
Task flag	$T \in \{0, 1\}$
Skill type	$S \in \{1, 2, 3, 4, 5\}$
View size	$d \in \{3, 5, 10, 20\}$
Utility	$U \in \mathbb{R}$
Busy flag	$busy \in \{0, 1\}$
Commission (NT)	$\alpha \in \{0 \dots 1\}$
Accept Threshold (NS)	$\beta \in \{0 \dots 1\}$

4.1.3 The Network

As we said the firm is modeled as a P2P network in which each node has a maximum number of links (view size, d). Each link is bidirectional; a connection of a node a to another node b implies a connection of node b to node a. Links are undirected so the entire network can be considered as an undirected graph where each vertex is a node and each edge is a link. All the nodes in the network are NSs; some nodes are NTs, these latter are designated *a priori* with a random function. Figure 4.2 shows an example of how the firm is represented and of how tasks are assigned. Green nodes are NTs.

4.2 The Simulation model

We developed different versions of our simulation model with the aim to explore different individual behavior and then develop a number of hypotheses about potential process that may occur in an organization.

We performed simulations using PeerSim [105], an agent-based platform for testing P2P protocols developed at the Department of Computer Science of the University of Bologna (see appendix A). Peersim is a P2P protocol testbed developed with Java and the simulation scheme is cycle driven. Peersim has been previously adopted

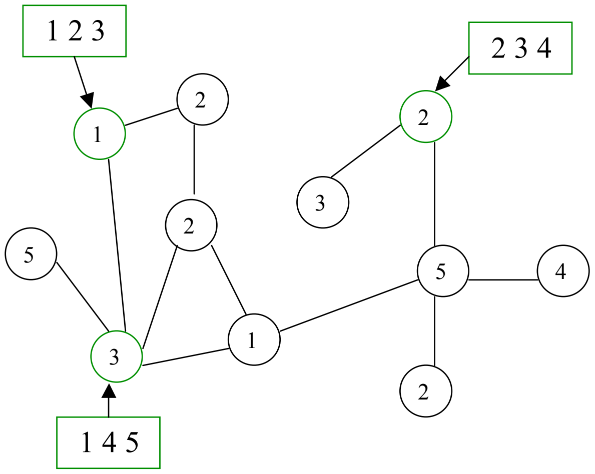


Figure 4.2: Example of how the firm is modeled in our system. Green nodes are NTs: they are designated *a priori* and never change over the time.

for testing protocols for P2P networks, but we found that it can be a good testbed also for organizational strategies. In our model the firm is a network composed of 1000 nodes; in this network we distinguish 250 NT and 750 NS. In all the simulation experiments we run 500 cycles. At cycle 0, with a random function NTs are designated. Every node has an uniform probability to become NT; once a node becomes NT, it will be NT for the rest of the simulation. Every 20 cycles the managers (NTs) receive a task to be completed. In the simulation environment we distinguish two main phases: an **Interaction** phase and a **Network Evolution** phase. The first is the one in which NTs try to complete the task asking their neighbors to provide their skills; the second (also named rewiring phase) is the one in which the network evolves according to some particular mechanisms[1] forming teams of specialists. In the Interaction phase managers receive the tasks according to some particular task environment[2]. Their aim is to complete as many tasks as possible. To achieve this they start looking for employees among their immediate neighbors. If they find a node with the right skill which is not busy, they make an offer to him for a certain

[1]The different mechanism that we implemented will be described in the next chapters and constitute the main differences between the different steps toward which we moved our model.

[2]In each chapter we describe the task environment that we use.

margin α (commission); if this offer is greater than the node's acceptance threshold (β), the job will be assigned and the node will get the appropriate payoff (figure 4.3 shows a typical example). The payoff that NS will get, is always equal to its β: of course in order to accept a job, α must be greater than or equal to β but at the end the NT will pay the minimum amount. Following this line, if a NT is connected to two or more NS bearing the same skill, it will assign the job to the one with lowest β.

In the Network Evolution phase (rewiring) the network evolves according to some rules. When the interaction phase takes place, NTs not always succeed in completing all their tasks and this is due to the fact that their neighbors are not always able (for the reasons seen above) or are not enough to complete the task currently held. For this reason the network and hence the teams must be reorganized in order to be more efficient in the next interaction round. With our work we developed several different ways for organizing the network. First we tested and compared some Hierarchical mechanism and some Self-Organizing mechanisms. Then we investigated more on different Self-Organizing mechanisms. In general what happens with all the rewiring mechanisms we developed, is that periodically a certain node i selects a random node j to which eventually establish a bidirectional connection. The random node is provided by a peer sampling service implemented by the Newscast protocol (Jelasity et. al., 2002 [46]). To better understand how this process of node selection takes place, In the next subsection we give a description of the architecture of our model within Peersim.

Another phase which is in general present in our model is the **Imitation** phase. With this phase, periodically nodes imitate strategies of better performing nodes. In general by better performing nodes, we mean nodes having an higher accumulated payoff (or prestige). More details on this phase will be given in the next chapters, when it will be used.

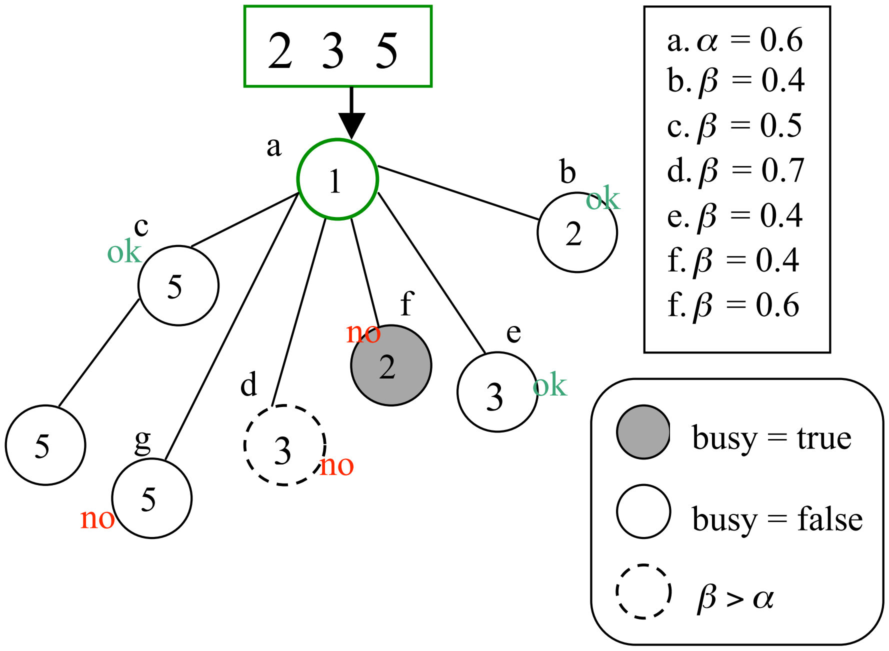


Figure 4.3: Example of the way NT bargains with its neighbors for the completion of the task (Interaction phase). Node f is busy and cannot accept the job; node d refuses the job because its β is greater than NT's α. Node g has the same skill as node c but it will not get the job: node c is selected because it has a lower β.

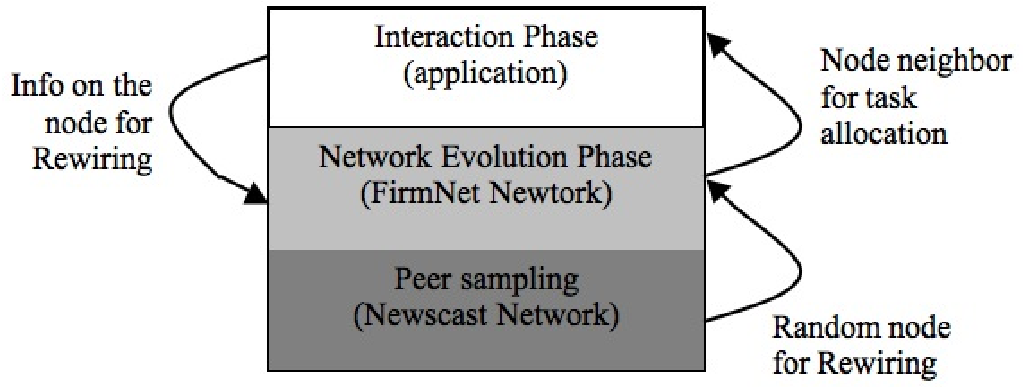


Figure 4.4: Layered architectural model of FirmNet in Peersim.

4.2.1 FirmNet in Peersim

Given the cycle driven nature of Peersim and its high modularity with respect to the nodes' protocols, we decided to implement FirmNet as a three-layer architecture, looking at the random peer sampling (Newscast), the Network Evolution phase (FirmNet network), and the Interaction phase (application) as three different and distinct layers, as shown in figure 4.4. The lower layer, providing randomly sampled peers to the rewiring mechanism, is implemented using the Newscast protocol (a deeper description of Newscast is given in appendix B). The Network Evolution layer (rewiring) is defined on top of the Newscast layer. It manages the topology and performs the rewiring mechanism (described in the next chapters), accessing to the Newscast layer when a random peer is needed and accessing to the application layer for retrieving informations on such node. The Interaction layer (application) is the one which carries the Interaction phase where NTs try to complete the task asking their neighbors obtained accessing the FirmNet network on the underlying layer.

To exploit the modularity in the 3 level architecture of FirmNet, the only thing to take care of is the implementation of the methods used to interface different layers, namely Network Evolution would need a `getRandomNeighbor()` method to access the random sampling layer view and a `getINFO()` method to get the needed informations on the selected node for rewiring purposes. Finally since the applica-

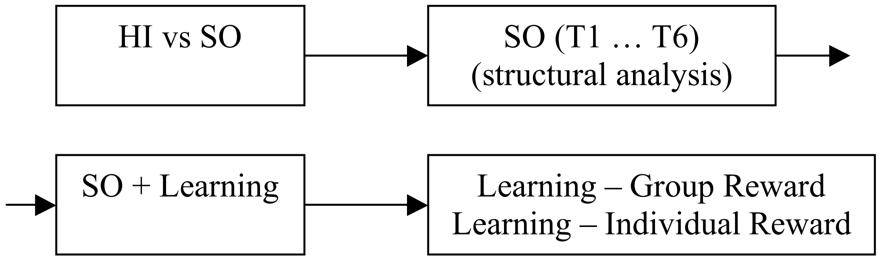


Figure 4.5: Evolution of the FirmNet model

tion layer does not manage any topology but relies on the one defined by Network Evolution it needs a `getNeighbor()` method to access the Network Evolution layer (FirmNet network) node view.

4.3 Model Evolution

So far in this chapter we have described the core of our model. We developed several hypothesis and for doing this our model has evolved over the time.

In all the version of our model that we developed, the task is composed of three jobs as in figure 4.1, while the number of possible skills changed from 5 to 9. The main evolutions related to our model are four and are indicated in figure 4.5. The first thing that we did (Mollona and Marcozzi 2008, [68]) was to compare a Hierarchical model with a quasi-market based model (we called it Self Organizing) – see chapter 5; then we focused more on the SO model developing six different versions of the rewiring phase and performing some structural network analysis (Mollona and Marcozzi, 2008 [66, 67]) – see chapter 6; after this we decided to introduce some learning mechanisms to our SO model and finally we compared two different SO models in which we compared a "group reward" policy with an "individual reward" policy (Mollona and Marcozzi, *submitted* [69]) – see chapter 7. All these works will be described in the next chapters.

Chapter 5

Hierarchy vs. Market

In this chapter we highlight the early work we did with the FirmNet model (Mollona and Marcozzi, 2008 [68]). We begin describing the general model and the motivations, then we describe the "hierarchical" and "self-organizing" models and finally we indicate the experiments we conducted and analyze the main results. The last section points on the conclusions and future steps.

5.1 Introduction

Grounding on the analysis of decentralized decision-making, an area of literature in organizational theory has emerged which addresses teams as the solution of decentralization problem in complex decision-making (Marschak and Radner, 1972 [61]). Zoethout (2006 [103]) proposed a multi-agent simulation model to explore how different types of task variety cause workgroups to change their task allocation accordingly. Within this area, we address two problems. First, we analyse how firms, which operates by the means of teams of specialists, adapt to dynamic task environments. Second, we compare hierarchical control with a market-based mechanism and investigate dynamics of self-organization of a firm's skills partition into teams. The firm we represent is a consultancy firm providing professional service. In this kind of firm we can distinguish at least a number of team leaders, or project managers, that have contact with clients and are responsible for the completion of tasks. In addition, a number of other professionals, holding specific skills, are linked to the organization with employment relationships which may include market-like incentives (the basic technical details of our model are described in chapter 4). Our focal firm operates using a team-based organization. We compare the impact of two idealtypes of organisations on the focal firm's performances, these latter calculated as profits accumulated and percentage of completed tasks. First type of organization adopts traditional hierarchy and employment relationship which gives the firm the authority to direct professionals where they are needed in different teams. However, such a solution implies that employed professionals are paid independently of the demand for the skill which they have. In the second idealtype of organization, professionals are not employed but work on a margin-per-consultancy basis. Boundaries of teams are permeable and informal interaction is likely to take place among members within the same team and across teams. Such a solution is probably more flexible but entails opposite incentives. For example, the firm favours skills travelling among teams in order to match skills to tasks. On the other hand, such a free movements of skills within the organization may also creates competition among project managers for skills hold by professionals. To explore different performances of the two idealtypes, we designed an artificial organization, the FirmNet proposing two models, a Hierarchical model (HI) and a Self-Orfanizing model (SO). We expect that in a knowledge-economy the organization of the economic activity of specialists should be, at least partially, self-organized; we used the performances of the HI model as a benchmark for the SO one.

5.2 Model Overview

In this work we used an agent-based model to simulate interaction among professionals holding specific expertise and project managers, which receive from clients tasks to be performed. In the previous chapter we have seen the core of our model describing the state of the agents, the task and how managers allocate tasks.

As we said, in our model firms are represented indirectly as a network of agents that receive tasks from clients and offer to professional a reward in exchange of their collaboration. The first thing that we did, was to compare a hierarchical form of organization with a quasi-market form of organization, trying to understand which are the conditions that make one perform better than the other one. For this purpose we implemented two different models: a *hierarchical* model (HI) and a *marlek-based model* (we called it self-organizing, SO) in order to understand which one of the two decision making processes performs better in some specific environment.

As already seen in chapter 4, we have two kind of agents: the *node-skill agents* (*NS*) and the *node-task agents* (*NT*). These agents are nodes in a Peer-to-Peer Network; they all hold a certain skill (S) that they use to perform some task. Agents NT, in addition, play the role of project managers, have direct contacts with clients and receive a certain Task to be completed. To complete a task, it is necessary to complete three jobs (see figure 4.1), each requiring a different skill and NT agents need to form a team attracting NS agents, which hold the required skill. Thus, the difference between NT and NS agents is that NT agents arbitrage, on behalf of the firm, the relationship between skills and clients. Hence, the model simulates an organizational network in which teams arise having certain skills. In both models the designed organization network is a Peer-to-Peer Network in which each node has a maximum number of links (network degree). Each link is bidirectional; a connection of a node a to another node b implies a connection of node b to node a. Links are undirected so the entire network can be considered as an undirected graph where each vertex is a node and each edge is a link.

In both models we can distinguish two phases: an *Interaction* phase and a *Network Evolution* phase. The first is the one in which NT try to complete the task, also asking its neighbors; the second is the one in which the network evolves according to a self-organizing or a hierarchical algorithm, forming teams of specialists. However the main differences between the HI and the SO model are the following: in HI all the NS agents periodically receive a fixed wage even though their *skill* is not used for any job and once they are asked for working on a certain job, they cannot refuse unless they are busy (already working on another task); in SO instead, a NS will accept a call for a job only if not busy and if the margin α (commission) it will receive from NT is greater than its own satisfaction threshold β; moreover, NS will not receive a fixed salary at a fixed period, but only the commission paid by the NT after the *entire task* is completed. NT will receive its payoff only when the task is completed.

So with these two models we try to develop a number of hypotheses about potential process that may occur in an organization. We do this computationally using computer simulation. As we said in our model firms are represented as Peer-to-Peer networks and we performed simulation using PeerSim [105]. The simulation time is divided into cycles. The network is composed of 1000 nodes (agents) and at cycle 0 the 25% of the them receive a task to be completed; so we have 25% of NT and 75% of NS. The tasks are produced selecting at random three values from a set of five elements ($J \in \{1, 2, 3, 4, 5\}$); the receiving nodes will then act as a *project manager* (NT) and will start looking for employees among their immediate neighbors (this is the interaction phase). After this, with a certain probability the Network Evolution phase takes place. Basically the nodes invoking this phase select a random node in the network and then, according to some rules, some rewiring action takes place (changing of the links). In the HI model, only NTs perform this phase and the selection is made on the entire network. In the SO model this phase can be performed by anybody and the selection is made only on a portion of the network (via a peer sampling service). As we said in the HI model, only NTs invoke this phase searching for NS with the right skill; in SO this phase can be invoked by anybody: we implemented two different mechanisms of selection and rewiring called CSLAC and CSLAC2 which will be described in sections 5.4.2 and 5.4.3.

5.3 The Hierarchical model

As said in the previous section, both the hierarchical and the self-organiziging model are made of two phase. In this section we describe the *interaction* phase and the *network evolution* phase of the HI model.

```
At each cycle if Ni == NT :
        if (Ni != busy) && (Ni.task.contains(Ni.skill)) then
                Ni=busy; Ni.task.remove(Ni.skill);
        endif
        if (Ni.task != empty) then for all Ni's neighbors j :
                if(Ni.task.contains(Nj.skill) && (Nj != busy) && (Ni.$\alpha$ > Nj.$\beta$))
                then
                        Nj = busy;
                        Ni.task.remove(Nj.skill);
                endif
End
```

```
At each cycle if Ni == NT :
        if (Ni != busy) && (Ni.task.contains(Ni.skill)) then
                Ni=busy; Ni.task.remove(Ni.skill);
        endif
        if (Ni.task != empty) then for all Ni's neighbors j :
                if(Ni.task.contains(Nj.skill) && (Nj != busy))
                then
                        Nj = busy;
                        Ni.task.remove(Nj.skill);
                endif
End
```

Figure 5.1: Interaction phase pseudo-code: the code on the top is executed in the Self-Organizing model; the one on the bottom in the Hierarchical model. We can note how the only difference is that in the HI model, when a NT asks to a neighbor to provide its skill, no bargaining takes place.

5.3.1 Interaction phase

Periodically NTs receive a task to be completed: they first look among their immediate neighbors for NSs able to work on such task; if the task is completed the NT will get the appropriate payoff; if the task is not completed, with a certain probability the network evolution phase takes place. In this model payoffs are fixed, hence the nominal reward of each job is not shared among NT and NSs. Every 20 cycles all NSs get a fixed wage even though they did not participate in any task completion. On the other hand, NTs will get a fixed payoff only if they complete the assigned task; if not after 20 cycle they receive a new task. Figure 5.1 shows the pseudo-code of this algorithm.

5.3.2 Network Evolution phase

The evolution of the network is handled by the NT agents which can select the appropriate NS. This happens periodically with probability 0.9. Since a NS can refuse a link only if it is busy, NT agents decide who must join their team. When a NT performs this phase, it select a random NS from the entire network[1]: if this node is not busy and its skill is needed by NT, it will drop all its links and will link to NT. Since NSs every 20 cycles get a fixed wage, they cannot refuse to join a certain team, unless they are 'busy'.

5.4 The Self-Organizing model

Also in the SO model we have an interaction phase and a network evolution phase but for this model we developed two different evolution phase (sections 5.4.2, 5.4.3) in order to test different possible behaviors of the agents. Basically the characteristics of the two algorithms are the following:

[1]The random node is provided by a peer sampling service. In the HI model, the random node is selected from the whole network (global view).

- CSLAC[2]: all the nodes behave in the same way. When node a selects node b, it will link to him and to its neighborhood only if it is richer ($U_b > U_a$). Node b cannot refuse. Some rules for imitation are applied (more details will be given later);

- CSLAC2: NTs and NSs behave in different ways. NTs have a preferential for NSs holding a skill that can be used in its current task; NSs have a preferential for NTs. A node receiving a request for a link can refuse. Also here some rules for imitation are applied (more details will be given later).

5.4.1 Interaction phase

Periodically NT receive a task to be completed. To achieve this they start looking for employees among their immediate neighbors. If they find a node with the right skill which is not busy, they make an offer to him for a certain margin α (commission); if this offer is greater than the node's acceptance threshold (β), the job will be assigned and the node will get the margin payoff only when the entire task gets completed. Figure 5.1 shows the pseudo-code of this algorithm.

In force of this, here NTs have less chances of completing tasks than in the HI model, but here the firm has less costs due to the fact that wage is not fixed but is paid only if tasks are completed.

Execution and Bargaining

The NT nodes define which task they need to complete; assesses which skills are needed and assesses which skills are available. If more than one NS with same skill is available, select the one with lowest β.

[2]The name CSLAC stands for Competitive-SLAC. This algorithm is based on the SLAC algorithm (Selfish Link and behaviour Adaptation to produce Cooperation) proposed by Hales (2002 [37])

5.4.2 Network Evolution phase: CSLAC

The CSLAC algorithm is an evolution of the SLAC algorithm proposed by Hales (2002 [37]) and than further developed and adapted to networks (Marcozzi et al, 2005 [59]; Hales and Arteconi, 2006 [38]). SLAC (Selfish Link-based Adaptation for Cooperation) has the ability to produce high levels of cooperation in P2P networks while performing some tasks. It specifies how nodes update their strategies (in our case the α and β values) and links under the assumption that they are involved in some on-going interaction with neighbors from which they can derive utility measures. At each cycle of the simulation task, with a certain probability, the SLAC algorithm is invoked. It is executed by each node: it periodically compares its own utility (say U_i) with the utility of another node (say U_j) randomly chosen from the network[3]. Suppose node i has an utility greater than j ($U_i > U_j$): in this case j copies node i's α and β values; j drops all its links and moves to i's neighborhood (copies all i's links and adds a link to i itself). As already seen, each node has a *maximum view size* (d): if a new node has to be added in an already full view, a randomly selected node is discarded to make place for the new node. The rewiring operations are symmetric: if node i makes a link to node j then node j makes a link to i; if node i drops a link to j, the link from j to i is dropped as well.

The CSLAC (Competitive-SLAC) algorithm is very similar to the original SLAC. The rewiring step is carried in the same way as in SLAC. The wiring action takes place only if selected nodes are not busy. The difference is in the coping of the strategies phase. While in SLAC the losing node copies the winner's one without taking into account its values, here the losing node before coping the winners β checks if some of the winner's neighbors has its same skill: if this happens and this node also has a smaller β than its own, it will copy this value minus a 0.1 constant. The rationale behind this, is to create competition between NS agents. If a certain

[3]The random node is provided by a peer sampling service (Newscast). In the SO model the node is selected from a portion of the network. This limited view depends on the network degree: the higher the degree, the greater the view. Degree 20 gives a global view of the network.

NS is going to join a new community, it will go in competition with the other nodes having its same skill by lowering its own accept threshold (β). Next cycle it will have more chances to get the job.

5.4.3 Network Evolution phase: CSLAC2

Now we propose a new mechanism based on the own needs of each agent. Every node can perform this phase. The key point is that there is always a preference for NSs to wire to NTs and for NTs to wire to NSs. Periodically a node i selects a random node j through the peer sampling service implemented by Newscast (same as in CSLAC). Assessment to decide how to wire is based on the following criteria:

- *for NT*: 1) wire to node holding the skill desired; 2) node with high degree;
- *for NS*: 1) wire to NT; 2) node with high degree;

The wiring action takes place only if selected nodes are not busy. Thus we have four cases of matching (the expression X–>Y indicates that X selects Y). In this model this phase is composed of two steps: **rewiring** in which nodes make new links; **imitation** in which nodes copy the strategies (α and β) of the counterpart. We must point out that the rewiring step and the imitation step are executed consecutively, meaning that if rewiring takes place, also imitation is done.

Rewiring step

Table 5.1 shows the rules used in the rewiring step. We have four cases, one for each type of possible connections. In addition to the selected node, the rewiring may involve also its neighbors, as in SLAC. As we said each node can have a maximum of d links (network degree) and the idea behind this step is to maximize the number of links for each node. When a node x selects a node y, it first asses whether to wire to it according to the criteria described above, then it can chose its new links among a list of N nodes, where N is given by the sum of x's link plus y's links. The basic idea is that x will first link to all these nodes, and suddenly then will drop D links in order to respect the limit of d links. The D nodes to be dropped are chosen with the *degree criteria*, which selects nodes with low degree.

Imitation step

Also in the imitation step we have four cases: they are indicated in table 5.2. As in CSLAC when a node joins a new new neighborhood, it looks for other nodes with its same skill; if someone is found, with a lower β too, it will copy its β minus a 0.1 constant.

5.5 Experimental configuration

We performed several experiments with both the models we have mentioned above. Simulations were carried on *Peersim* ([105]), an open source P2P systems simulator platform, using the *Newscast* protocol (Jelasity et al., 2002 [46]) for the management of the overlay topology. We use this protocol to implement a service (peer sampling service) to pick a random node which is used in the Network Evolution phase. The time is divided in cycles and in each cycle each node performs the specific actions described in the previous sections. In each experiment we checked how the algorithms adopted influence the Percentage of completed tasks (Pct) and the Wealth of the Firm.

The configuration we adopted in the experiments is the following:

- Network size (N): 1000;
- Network degree: $d \in \{3, 5, 10, 20\}$;
- Rewiring period: $c \in \{1, 15, 30, 60\}$;
- Initial network topology: *random*;
- Skill initialization: all the nodes were initialized with a random *skill* taken from a set of 5 elements ($S_i \in \{1, 2, 3, 4, 5, \}$);

- α and β initialization: a random value between 0 and 1 (these parameter are present only in the SO model);
- Payoffs: in the SO model the reward given by each job is 1 (3 for the total task); in the HI model the reward for NT is 1.80 and the wage for NS is 0.40 (also here 3 for the total task but fixed margins for the NS) .

For each configuration we performed 10 different run and we took the average of the results. In the simulation experiments we tested effectiveness of the two individual decision-making. We tested the two models in two different task environments, a *static* one in which the same task arrives every 20 cycles and a *dynamic* one in which tasks change every 20 cycles.. We also varied the rewiring period, performing rewiring actions with probability of 0.9 every 1 cycle, 15 cycles, 30 cycles and 60 cycles. In addition, we varied network degree.

The parameters indicated in this section are those adopted for each model. We performed different set of experiment for each different evolutionary algorithm. Table 5.3 resumes the characteristics of each algorithm.

5.6 Experimental Results

We performed a large number of experiments with our models trying to work out which one performs better in some situation, investigating both in a dynamic environment (in which NTs periodically substitutes their tasks with a new different one) and a static environment (in which each NT has a different task; periodically each task is replaced with a new one identical to the previous one). We adopted two measures for evaluating them: the percentage of completed tasks (Pct) calculated as the ratio between the number of tasks completed and the number of tasks injected and the firm wealth (wealth) calculated as the total income of the firm minus the total costs (payoffs given to NSs). We performed experiments with different network degree values (maximum number of links of each node: d3, d5, d10, d20). We found that hierarchy outperforms self-organizing in terms of percentage of task completed. As the size of intra-organizational networks increases (this latter measured as the "degree"), SO algorithm improves. The wealth created under hierarchical mechanism is higher than SO only for low network degree. When degree of intraorganizational network increases, SO algorithm produces significantly more profits. The clear argument that emerges from the simulation experiments is that the selection of a hierarchal or a market-based mechanism of control entails the trade off between reliability and costs. At a cheaper price SO gives the same good performaces as HI but only when the organisational environment keeps a high frequence of interactions together with a large number of links at the same time. Hence, it is clear that the efficiency of the market-based model depends on the structure of the network that can be formed, for example, when the network degree is high.

5.6.1 Static Environment

In this section we give an illustration of the results obtained with the static environment: here nodes task (NT) receive a particular task at the beginning of the simulation and such task never changes over the time (every 20 cycles they receive a new task which is identical to the original one). Here different NTs have different task.

Figures 5.2 and 5.3 indicate respectively the Pct and the Wealth of the firm with different rewiring periods with the **Hierarchical** model. We can note how better results are obtained when $c = 1$ (actually, this happens in all the experiments we have done).

Figures 5.4 and 5.5 indicate respectively the Pct and the Wealth of the firm with different rewiring periods with the **CSLAC** algorithm. Also here we can note how better results are obtained when $c = 1$.

Figures 5.6 and 5.7 indicate respectively the Pct and the Wealth of the firm with different rewiring periods with the **CSLAC2** algorithm. Also here we can note how better results are obtained when $c = 1$.

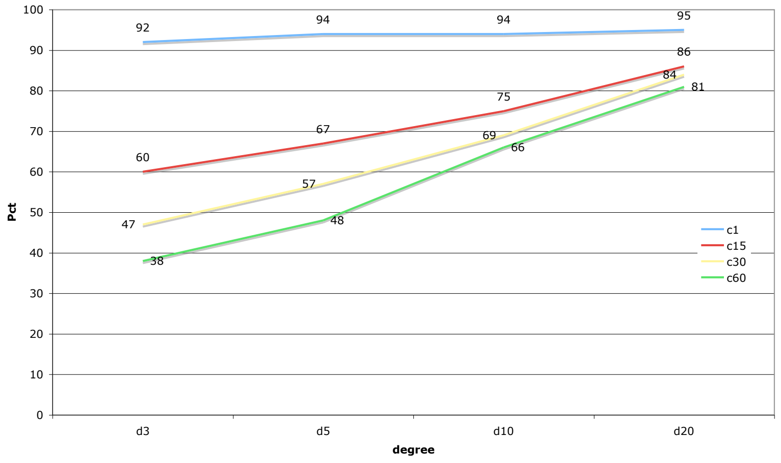


Figure 5.2: Hierarchical algorithm: Pct (percentage of completed tasks) in different periods. Static environment. Networks with undirected links and uniform degree over all nodes. Each curve indicates results with different periods for the execution of the "network evolution phase".

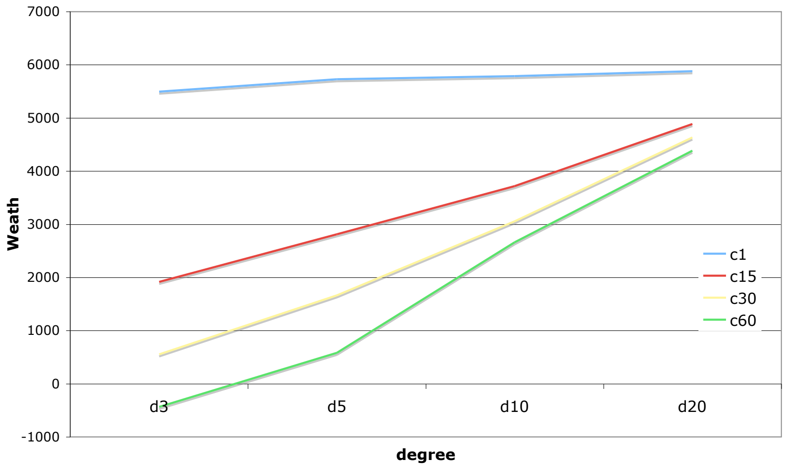


Figure 5.3: Hierarchical algorithm: Wealth in different periods. Static environment. Networks with undirected links and uniform degree over all nodes. Each curve indicates results with different periods for the execution of the "network evolution phase".

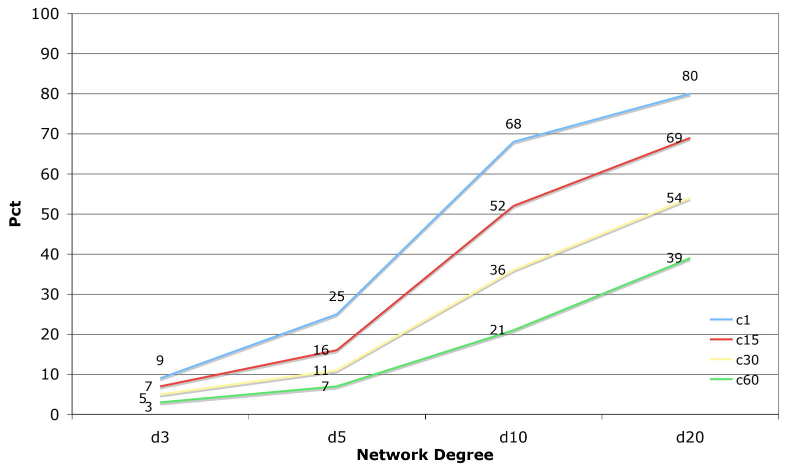


Figure 5.4: CSLAC algorithm: Pct (percentage of completed tasks) in different periods. Static environment. Networks with undirected links and uniform degree over all nodes. Each curve indicates results with different periods for the execution of the "network evolution phase".

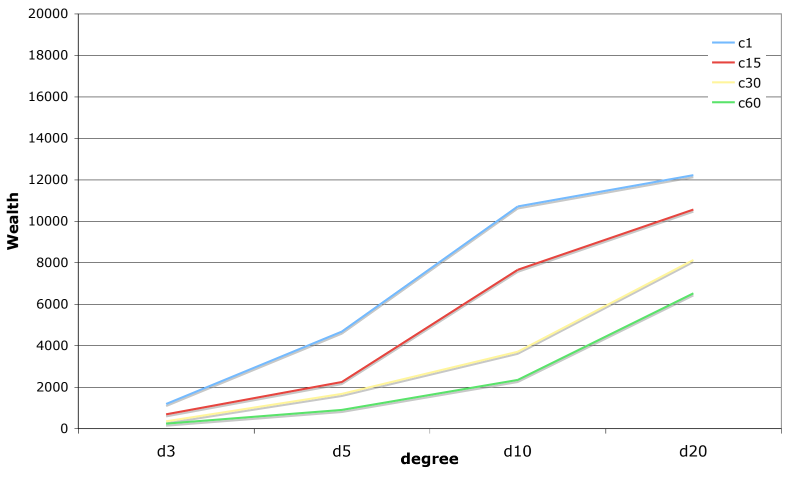


Figure 5.5: CSLAC algorithm: Wealth in different periods. Static environment. Networks with undirected links and uniform degree over all nodes. Each curve indicates results with different periods for the execution of the "network evolution phase".

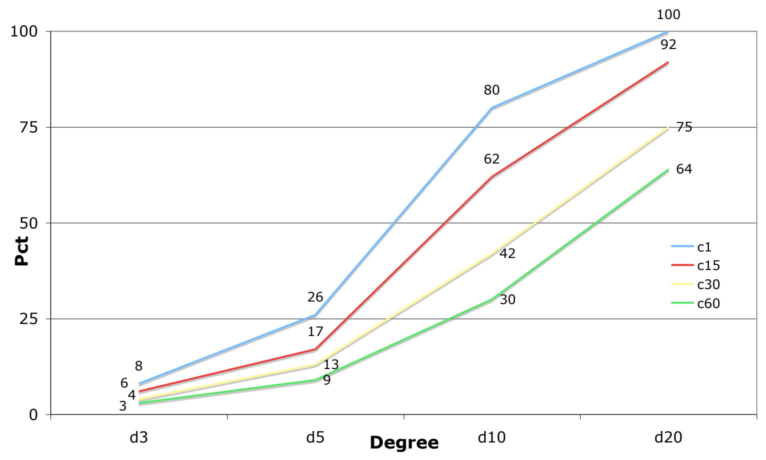


Figure 5.6: CSLAC2: Percentage of completed task in different periods. Static environment. Networks with undirected links and uniform degree over all nodes. Each curve indicates results with different periods for the execution of the "network evolution phase".

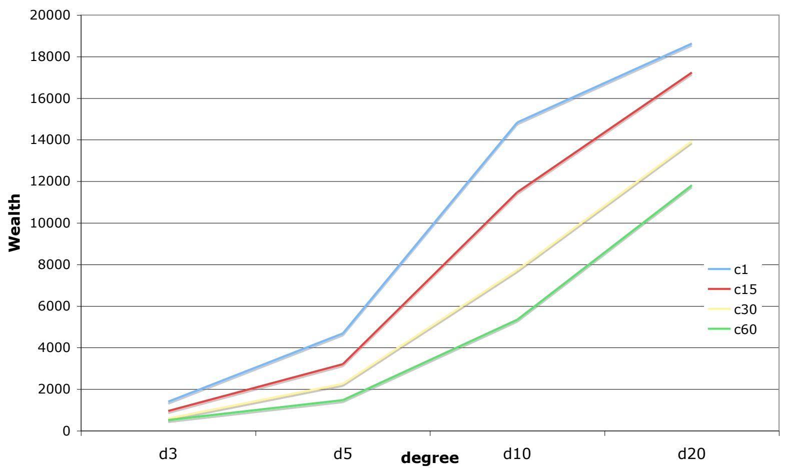


Figure 5.7: CSLAC2: Wealth in different periods. Static environment. Networks with undirected links and uniform degree over all nodes. Each curve indicates results with different periods for the execution of the "network evolution phase".

Discussion

The first thing that it is possible to note from those figures is that the more often the evolution phase takes place, the better results we get. We can also note that firm Pct and firm Wealth increases as the size of the intra-organizational firm (degree) increases. A high degree means that nodes have a large number of neighbors which gives to NTs more chances to find skills for the task. Moreover, in the SO model, a large network degree indicates that during the Network Evolution phase, the peer sampling service has a larger view from which to pick the random node; in particular degree 20 gives approximately a view of the whole network (Jelasity et al. 2003, [46]). In figures 5.8 and 5.9 we grouped the results obtained with the three rewiring algorithms but only those obtained for $c = 1$. We can note how in general the hierarchy gives better results than the two self-organizing mechanism in terms of Pct. Interestingly we found that CSLAC2 gives better results than CSLAC and that it performs even better than the hierarchy when the network degree is 20. While in CSLAC nodes prefer making new links with nodes having high utility (selected nodes cannot refute a link) in CSLAC2, new links are made in a more accurate way: for example when a NT selects a NS, it will try to link to him only if it has the right skill. We believe that these more accurate rules are the reason why CSLAC2 performs better than CSLAC. With CSLAC2 we gave to the nodes a greater intelligence: they can decide to wire or not to nodes basing on their individual preferences. This opportunity makes them perform better. In terms of firm wealth, we can note how hierarchy is costly. Of course also here CSLAC2 performs better than CSLAC for the same reasons mentioned before.

5.6.2 Dynamic environment

This section contains the results obtained with the dynamic environment. Here NTs every 20 cycles receive a new task which is different from the previous one.

Figures 5.10 and 5.11 indicate respectively the Pct and the Wealth of the firm

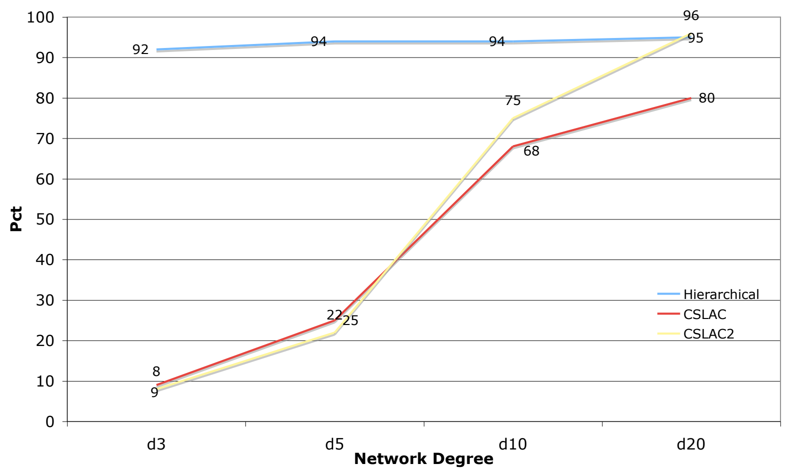


Figure 5.8: Resume: Percentage of completed task in different periods. Static environment. Networks with undirected links and uniform degree over all nodes. Each curve indicates results with different "network evolution phase" algorithms. Note that the Hierarchical approach outperforms the self-organzing approaches over most of the chart.

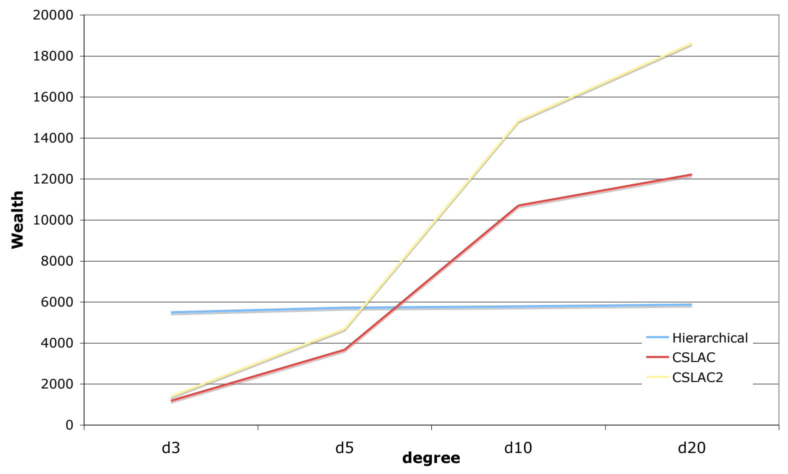


Figure 5.9: Resume: Wealth in different periods.Static environment. Networks with undirected links and uniform degree over all nodes. Each curve indicates results with different "network evolution phase" algorithms. Note that the self-organized approaches outperform the Hierarchical organization when the degree increases above about five.

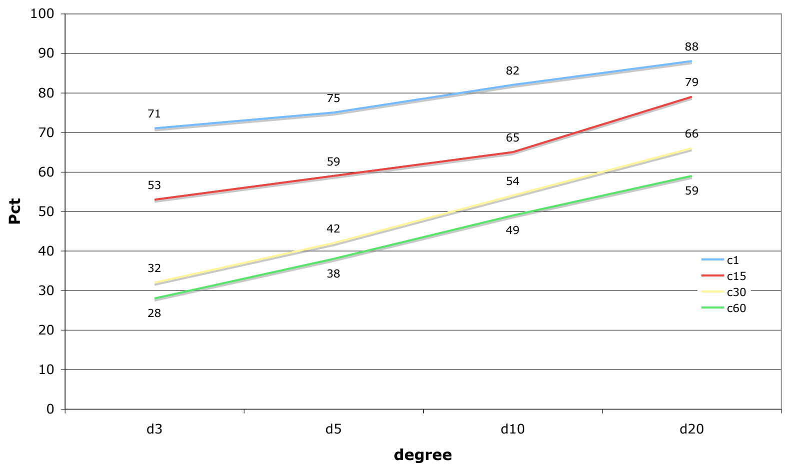


Figure 5.10: Hierarchical algorithm: Pct in different periods. Dynamic environment: here manager nodes receive randomly generated tasks over time rather than the same tasks (as previously). The hierarchical organization does less well than previously.

with different rewiring periods with the **Hierarchical** model. Also here we can note how better results are obtained when $c = 1$.

Figures 5.12 and 5.13 indicate respectively the Pct and the Wealth of the firm with different rewiring periods with the **CSLAC** algorithm. Also here we can note how better results are obtained when $c = 1$.

Figures 5.14 and 5.15 indicate respectively the Pct and the Wealth of the firm with different rewiring periods with the **CSLAC2** algorithm. Also here we can note how better results are obtained when $c = 1$.

Discussion

The first thing that it is possible to note from those figures is that the more often the evolution phase takes place, the better results we get. Also in the dynamic task environment, high degree entails better results. In figures 5.16 and 5.17 we grouped the results obtained with the four evolutionary algorithms but only in the case in which $c = 1$.

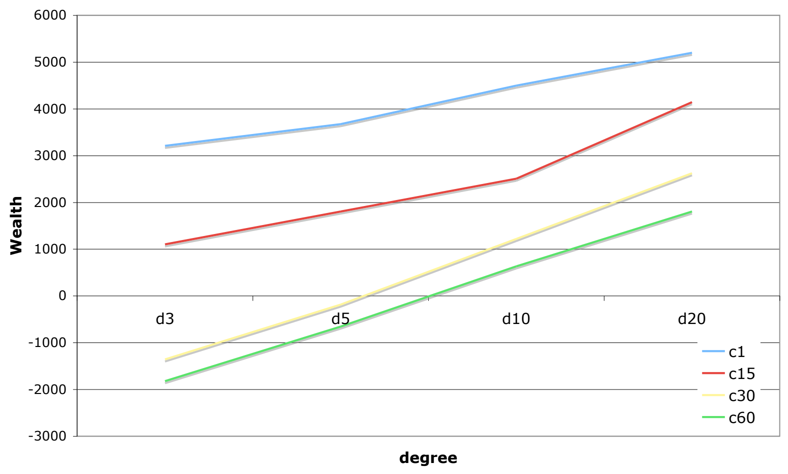


Figure 5.11: Hierarchical algorithm: Wealth in different periods. Dynamic environment: here manager nodes receive randomly generated tasks over time rather than the same tasks (as previously). The hierarchical organization does less well than previously.

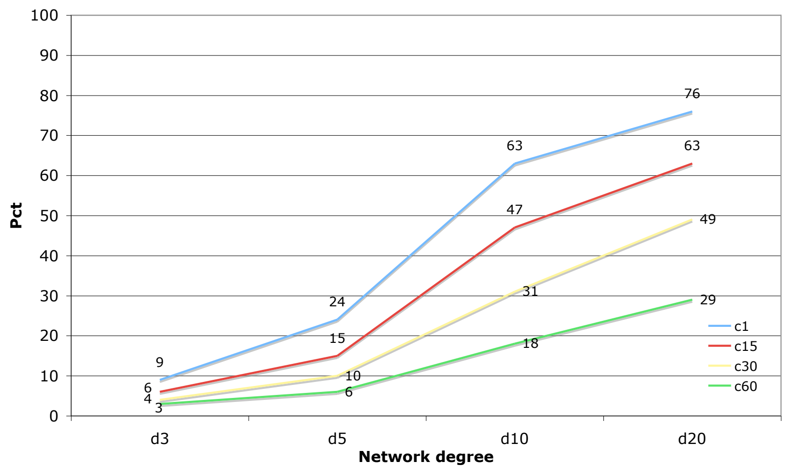


Figure 5.12: CSLAC algorithm: Pct in different periods. Dynamic environment: here manager nodes receive randomly generated tasks over time rather than the same tasks. Each curve indicates results with different periods for the execution of the “network evolution phase”.

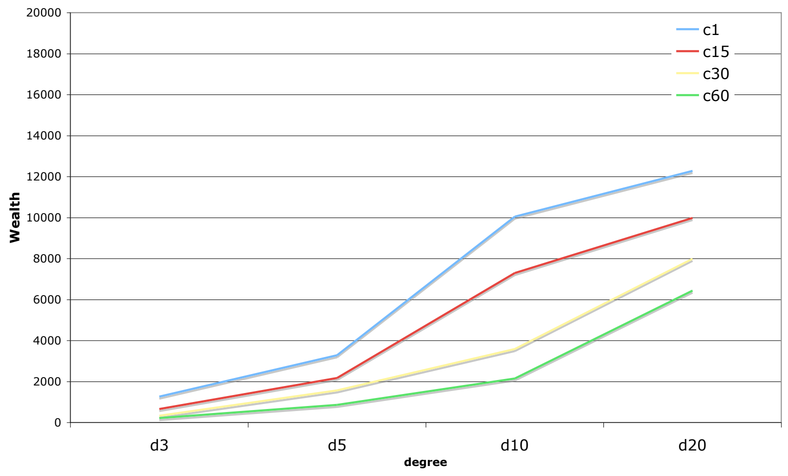


Figure 5.13: CSLAC algorithm: Wealth in different periods. Dynamic environment: here manager nodes receive randomly generated tasks over time rather than the same tasks. Each curve indicates results with different periods for the execution of the "network evolution phase".

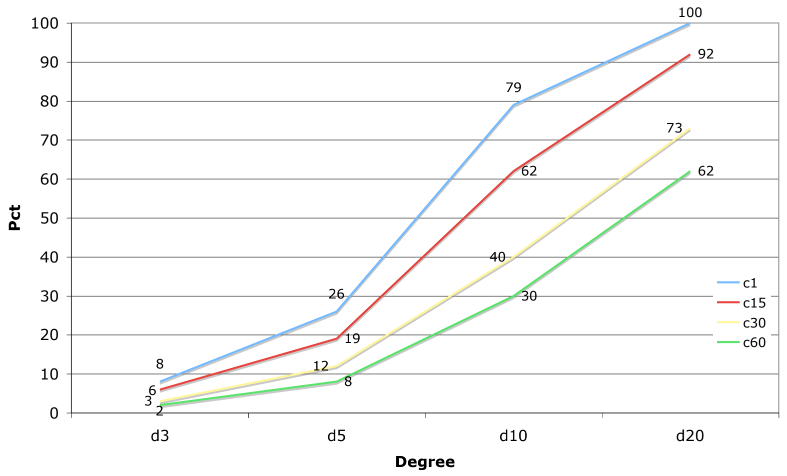


Figure 5.14: CSLAC2: Percentage of completed task in different periods. Dynamic environment: here manager nodes receive randomly generated tasks over time rather than the same tasks. Each curve indicates results with different periods for the execution of the "network evolution phase"

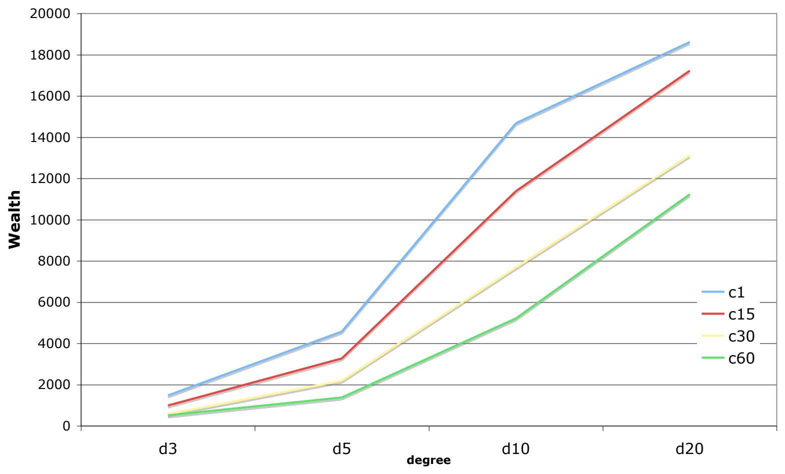


Figure 5.15: CSLAC2: Wealth in different periods. Dynamic environment: here manager nodes receive randomly generated tasks over time rather than the same tasks. Each curve indicates results with different periods for the execution of the "network evolution phase"

We can note how in terms of Pct, here the hierarchy gives worse results than in the static environment: this means that authority is less effective when the task rapidly changes. On the other hand CSLAC performs better here, while CSLAC2 gives more or less the same results. We think that in an environment in which tasks rapidly change (typical of a knowledge economy) a self-organizing method can give good results even though also here we can note how these good performances are obtained as the size of intra-organizational networks increases (this latter measured as the network degree). In terms of firm wealth, we can note that hierarchy costs much more than SO. Among the two SO algorithms we have seen that CSLAC2 performs better than CSLAC. We think this is due to the fact that CSLAC2 keeps the network more connected than CSLAC, allowing NTs to have a larger set of NSs which will compete one another decreasing their acceptance threshold (β). Figures 5.18 and 5.19 show the snapshot of the network at the end of the simulation for the dynamic environment with network degree 10; we can note how with CSLAC2 (figure 5.19) the network results more connected. As we have seen, CSLAC2 tends

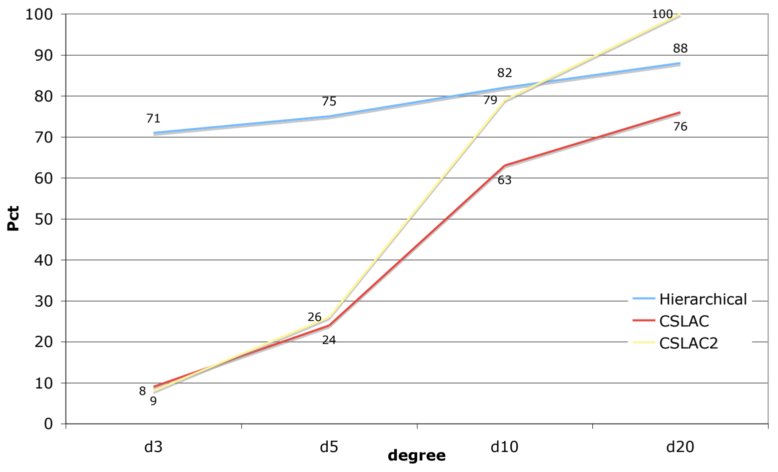


Figure 5.16: Resume: Percentage of completed task in different periods. Dynamic environment: here manager nodes receive randomly generated tasks over time rather than the same tasks. The hierarchical organization does less well than previously. The self-organising peer protocols do better when the degree is more than about ten. Essentially, since tasks are random over time manager nodes (NT) in the network are less able to benefit from hierarchy because they cannot predict which other nodes to recruit.

to keep as more ties as possible, preferring links with highly connected nodes and performing less disconnections than CSLAC; in force of this, often also old links are kept, forming a more interconnected network.

5.7 Conclusion

The work presented in this chapter contributes to the area of studies that is concerned with emergent organizational problems in knowledge economy. In addition we present an attempt to bridge studies in computer science, dealing with the nature and the mechanisms of P2P networks evolution and organization. To represent individual decision-making of agents embedded in organizational networks, we adapted an algorithm, the SLAC algorithm, developed to study cooperation in P2P networks.

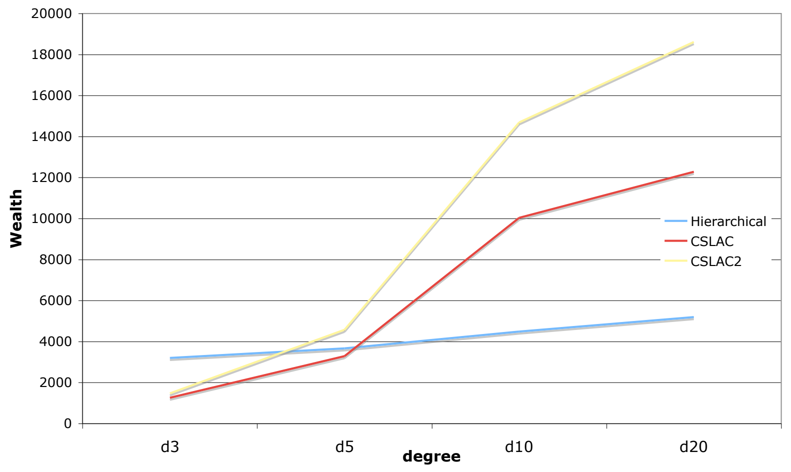


Figure 5.17: Resume: Wealth in different periods. Dynamic environment: here manager nodes receive randomly generated tasks over time rather than the same tasks. Hierarchy again performs poorly due to the requirement to give payoff to all nodes even when they are not productive.

In this chapter, we explored under what circumstances networks of specialist may self-organize. In this respect, we believe that the P2P network is a useful concept to address novel forms of organization in a knowledge economy. On these lines, we investigated how agents, with individual incentives and decision-making rules, interact locally and give raise to global organizational structures. We studied how different individual decision-making rules lead to different emergent network of skills, with different performances.

However, simulation experiments tease out problems and trade offs in applying self-organizing mechanisms to the government of a firm's skill endowment.

Choice between hierarchical and market-based control mechanisms may be a very subtle endeavour. In our computer simulation experiments, as task environment becomes dynamic, market-based mechanisms increasingly gain effectiveness. Deciding which mechanism better suits entails careful evaluation of two aspects. First, comparing alternative mechanisms requires the evaluation of costs. Cost of hierachical mechanism depends on salary paid and other costs of labor implied by

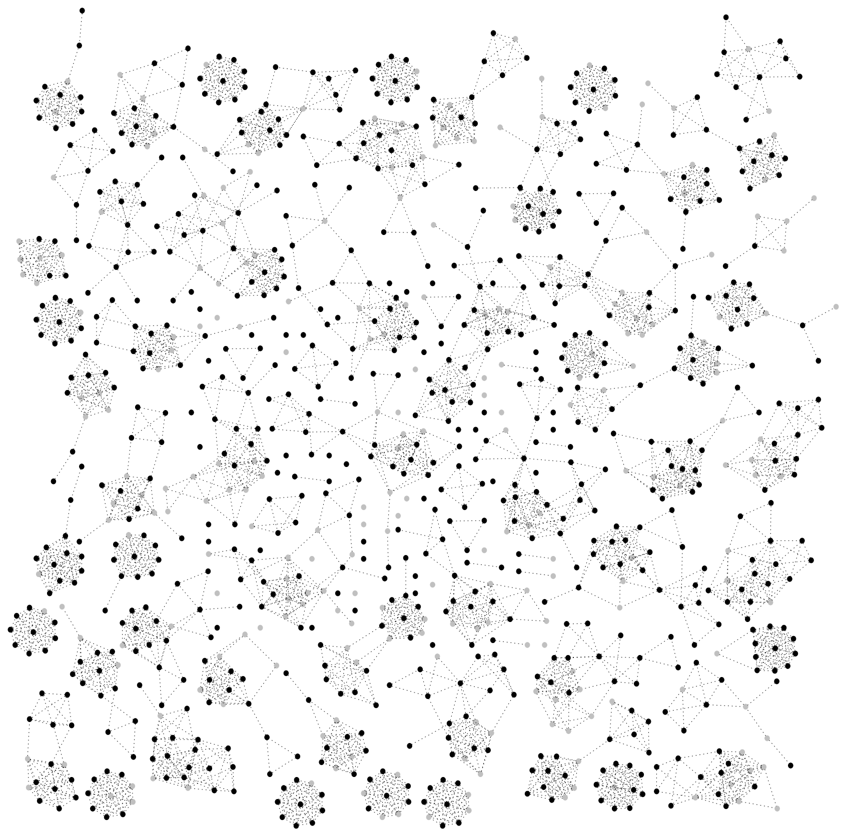


Figure 5.18: Snapshot of the network for the dynamic task environment with degree 10. Gray nodes are managers (NT), black nodes are employees (NS). CSLAC1 tends to create separate clusters and several disconnected nodes.

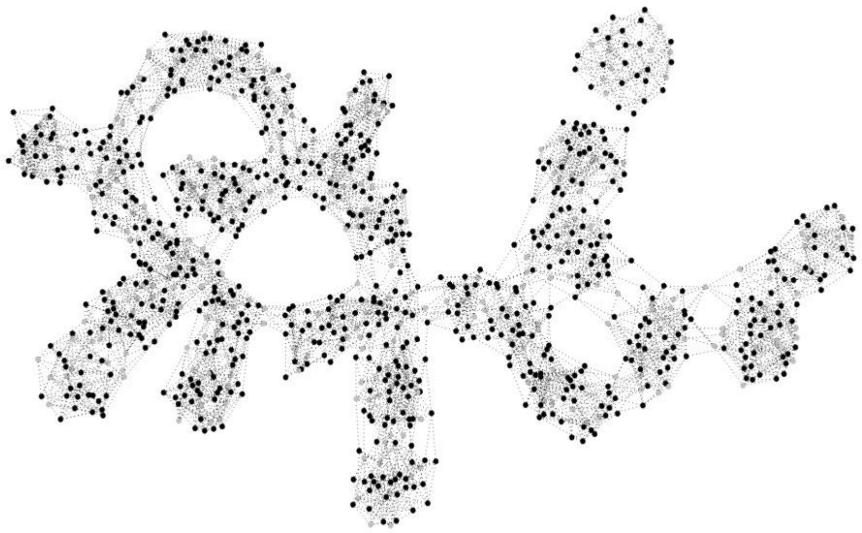


Figure 5.19: Snapshot of the network for the dynamic task environment with degree 10. Gray nodes are managers (NT), black nodes are employees (NS). CSLAC2 tends to keep clusters connected.

the employment relationship. Cost of market-based mechanism depends on tools and infrastructures built in order to create and maintain intra-organizational networks. In addition, maintaining large intra-organizational may have the disadvantage of stimulating dysfunctional competition among project-managers for skills of professionals. Second, performance of market-based mechanisms strongly depend on network features. Size, stability and communication flows within networks are key aspects to address in order to evaluate relative desirability of non-hierarchical control mechanisms.

Simulation experiments also highlighted area for future work. First, we suspect that presented results change drammatically as the required size for teams changes. In the simulations, relationship between skills needed to complete a task and size of a node's degree is crucial to explain performances of CSLAC algorythim.

Second, the work did not consider team-based learning. We can hypotize that social capital and shared mental models emerge within a team that makes it easier to communicate and coordinate, and that creates inertia in inter-groups movements.

Team based learning may generate a force that increases effectiveness of market-based mechanisms by counteracting erosion of a team's skill endowment. Team learning will be studied in chapter 7.

Table 5.1: Rewiring step

Connection	**Action**	**Response**
NT –> NT	1. Do not wire to it 2. Connects to all its neighbors 3. Select D using degree criteria	Do not take action because no wiring is asked for
NT –> NS	1. If NS has the required skill a. Wire b. Connects to the neighborhood of NS and define D on the basis of the degree criteria 2. If NS hasn't the required skill a. Copy the neighbourhood of NS and define D on the basis of the degree criteria	A NS node always accepts a wiring from a NT node
NS –> NT	1. Always wire 2. Connects to all its neighbors 3. Start dropping D using the degree criteria	1. If NS holds the required skill always accept 2. Else not accept
NS –> NS	1. Consider the new node and its neighbourhood 2. Explore the list of N nodes (own links plus links of the new nodes plus the new node) 3. Select D on the following basis: a. First take NT b. Second use degree criteria	Not necessarily the NS that takes action is asking to wire to the selected node (it may have decided to wire to its neighbor but not to it, on the basis of degree). If the acting node is asking for wiring, the responding NS node decides using the degree criteria.

Table 5.2: Imitation step

Type of connection	**Action**
NT –> NT	If the linked node is richer, copy its α
NT –> NS	No coping activity
NS –> NT	No coping activity
NS –> NS	Each cycle search own neighborhood and sets its beta at -0,1 in respect to the lower beta of a NS node holding the same skill included in its neighborhood

Table 5.3: Experiments performed

Evolutionary Algorithm	**Description of the algorithm**
Hierarchical	NTs decide which NS must join a certain group basing on the skill
CSLAC	Nodes rewire to another node on the basis of the Utility Nodes bargain the margins and imitate strategies
CSLAC2	Nodes rewire to another node basing first on the kind of node (NT or NS); second on the degree of the node; Nodes bargain the margins and imitate strategies

Chapter 6

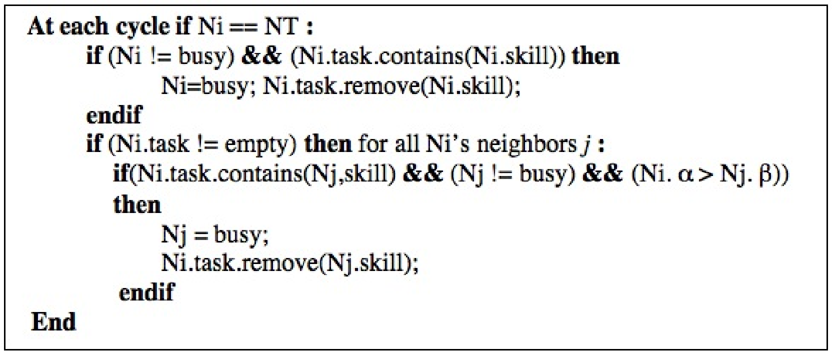
Self-Organizing mechanisms for task allocation

In this chapter we focus on the evaluation of six self-organizing mechanisms for the evolution of the network and the formation of teams. The six mechanisms we propose, derive from those seen in the previous chapter. The basic model is the same as before (see chapter 4) but here we articulate more the individual decision making of each single node and we separate the imitation step from the rewiring step. We also propose three different task environments (see Mollona and Marcozzi, 2008 [66, 67]).

6.1 Model Overview

In the previous chapter we compared two different decision making processes: we called them "Hierarchical model" (HI) and "Self-Organizing model" (SO). The aim was to understand which one of the two performs better in some specific environment. We found that the HI outperforms the SO, but for high network degree the SO model gives similar results in terms of completed tasks and better results in terms of wealth. In this chapter we want to focus on the SO model only, of which we propose six different versions (we will describe them in details later in this section).

The core of the model is the same we have described in chapter 4. We have a number of NT which periodically receive some task to be completed according



```
At each cycle if Ni == NT :
    if (Ni != busy) && (Ni.task.contains(Ni.skill)) then
        Ni=busy; Ni.task.remove(Ni.skill);
    endif
    if (Ni.task != empty) then for all Ni's neighbors j :
      if(Ni.task.contains(Nj,skill) && (Nj != busy) && (Ni. $\alpha$ > Nj. $\beta$))
      then
        Nj = busy;
        Ni.task.remove(Nj.skill);
      endif
End
```

Figure 6.1: Pseudo code of the Interaction phase.

to some specific task environment (see section 6.2). Hence also here we have an "interaction phase", where NTs try to complete the task, and a "network evolution phase" (rewiring) where nodes move according to some preferences. In addition, here we have an "imitation phase" which is independent from the other two phases and in which nodes copy more successful nodes' α and β values[1].

In the next subsections we describe the three phases in details.

6.1.1 Interaction phase

Periodically NTs receive a task to be completed. To achieve this they start looking for employees among their immediate neighbors. If they find a node with the right skill which is not busy, they offer him a certain commission (α); if this offer is greater than the nodes acceptance threshold (β), the job will be assigned and the node will get its share of prestige; the node manager instead, will get its prestige only when the entire task is completed. Figure 6.1 shows the pseudo-code of this algorithm.

6.1.2 Network Evolution phase

The network evolution phase is composed of two consecutive steps: a *selection step* and a *wiring step*. In the selection step periodically a node i selects a node j basing

[1]Actually this phase was present also in the previous model, but it was incorporated in the evolution phase.

on one of the three following criteria: "prestige" criteria, "degree" criteria and "skill" criteria. In all these three, periodically a certain node i selects a node j from a portion of the network (with the usual peer sampling service described in appendix B): then, if the prestige criteria is applied, the node with lower prestige (wealth) wires to the one with high prestige (according to 'closed' team or 'open' team rewiring mechanism); if the degree criteria is applied, the node with lower degree wires to the one with higher degree; finally, with the skill criteria, four possibilities may take place:

- a NT selects a NT: no wiring;
- a NT selects a NS: if NS have a skill needed by NT, the wiring step takes place, else no wiring action;
- a NS selects a NS: no wiring;
- a NS selects a NT: the NS tries to wire to NT (NT may refuse in case it does not need that skill).

After the selection, the wiring step takes place: we implemented two different versions, one called "Closed team rewiring" and one called "Open team rewiring". The idea of the closed team rewiring comes from a protocol called SLAC (Hales, 2002 [37]; Marcozzi et. al., 2005 [59]; Hales and Arteconi, 2006 [38]) which aims to produce cooperation in P2P systems (we have seen it in the previous chapter). Basically when the rewiring phase takes place, a certain node i selects a node j from a portion of the network: if some conditions are respected (those stated by the selection step), node i drops all its links and then connects to j and to all js neighbors. With this method, only the selected node must cope with the selection criteria. The idea behind this is that a certain node can work exclusively in one team. In the open team rewiring mechanism, if the selection criteria is respected node i links to node j but in this case it does not isolate from its original neighbors; it will keep the links to those nodes respecting the selection criteria and then will link to the j's neighbors which will cope with the selection criteria too. Hence, since for each selection criteria we have two rewiring criteria, in total we have six different evolution phases which are indicated in table 6.1.

Table 6.1: Network Evolution mechanisms: Prestige, Degree and Skill are the three selection criteria; Closed team and Open team indicate the type of rewiring

	Prestige	**Degree**	**Skill**
Closed Team	T1	T2	T3
Open Team	T4	T5	T6

6.1.3 Imitation phase

The imitation phase takes place at each cycle with a certain probability (0.9) and is independent from the interaction phase and the network evolution phase. When executed by a NS, if it is connected to a NT, it check if such NT is linked to other NSs with its same skill: in this case NS lower its β of a 0.1 constant, in order to be more competitive. When executed by a NT, say i, it selects a random NT, say j (using the usual peer sampling service). If the prestige of j is greater than the prestige of i, then i copies its α.

6.2 Experimental settings

We performed several experiments with all the versions we have mentioned above. Simulations were carried on Peersim [105] (see appendix A), using the Newscast protocol (Jelasity et al., 2003 [46] – see appendix B) for the management of the overlay topology. Newscast is a protocol that builds and maintains a continuously changing random graph. The generated topology is very stable and provides robust connectivity. We use this protocol to implement a service to pick a random node which is used in the Network Evolution phase. Hence we have two networks evolving in parallel: one which evolves following the Newscast protocol (random network) and one which evolves following one of the six network evolution mechanisms described in section 6.1.2 . In our simulations the time is divided in cycles and in each cycle each node performs the specific actions described in the previous sections. In each experiment we checked how the described algorithms influence the Percentage of completed tasks (Pct). At cycle zero our network and the network provided by Newscast are identical; then our network evolves according to the network evolution mechanism. With this mechanism periodically a certain node selects a random node and we implemented two different ways of doing this which we called Randomized Service 1 and 2:

- **Randomized Service 1** – RS1 – (Newscast view): node a selects a random node among the Newscast instance of its neighborhood. Doing this means that node a picks the random node from a network which is different from the network of nodes composing its actual working team (FirmNet view); the random node is picked from the Newscast layer of the system architecture (see figure 4.4); RS1 is the peer sampling service the we have used so far;

- **Randomized Service 2** – RS2 – (FirmNet view): node a selects a random node among its current neighborhood. This means that the node wants to strengthen the links with one of the individuals working in its team and its links; the random node is picked from the FirmNet layer of the system architecture (see figure 4.4).

The main difference between these two mechanism is that with the second method if node a is isolated, it has no chances of finding a random node and then to recover from this situation; on the other hand with Newscast, node a will find a random node because Newscast keeps the network connected. The idea behind these two different selection methods is that with RS1 we want to mimic a situation in which the new component / components of a certain team are selected from a kind of social network which is different from that one representing the current team;

with RS2 instead we want that the selection is made according to the acquaintances inside the team. The configuration we adopted in the experiments is the following:

- Network size (N): 1000;
- Network degree: $d \in \{3, 5, 10, 20\}$;
- Rewiring probability: 0.9 every cycle; - Initial network topology: random; - Skill initialization: all the nodes were initialized with a random skill taken from a set of 5 elements ($S_i \in \{1, 2, 3, 4, 5\}$)
- α and β initialization: a random value between 0 and 1;
- Payoffs: in some experiments the reward given by the entire task is 3 (1 for each job); in some experiments each task has a different total value (25% with 3 – 25% with 6 – 25% with 9 – 25% with 12).

Moreover we tested each configuration in three different task environment:

- **Env1**: same task to each NT; every 20 cycles the task is replaced with a new one identical to the previous one; same payoff to each task (static task environment);

- **Env2**: different task to each NT; every 20 cycles the task is replaced with a new different one; same payoff to each task (dynamic task environment);

- **Env3**: different task to each NT; different payoff to each task; every 20 cycles the task and the payoff are replaced with new different ones (dynamic task environment).

Hence we tested a totally static environment and two dynamic environments. In the next two subsection we respectively give the results obtained with the randomized service 1 and 2 on these three mentioned cases.

6.2.1 Randomized Service 1: results

Figure 6.2 indicates the Percentage of Completed Tasks in “Env1”, (static environment). We can note how all the mechanism give optimum results when degree is 20. The T6 algorithm always gives better results than the others, even when degree

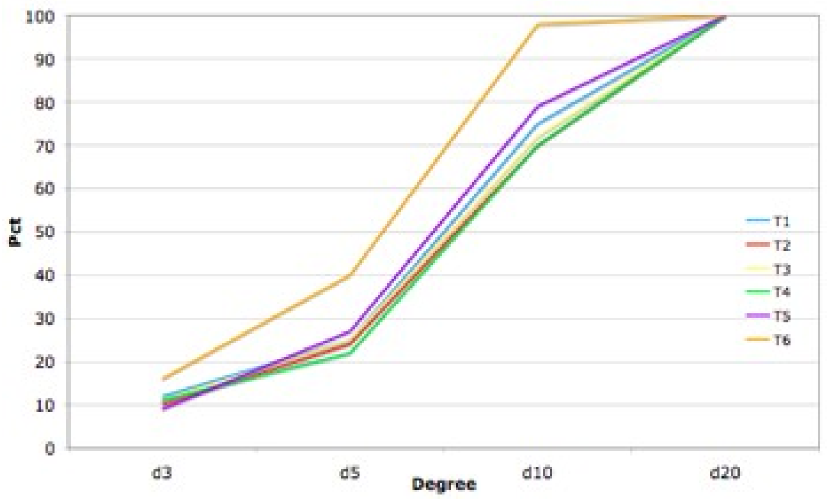


Figure 6.2: Percentage of completed tasks (Pct). Randomized Service 1. Env1, here each NT in the Network receive the same task. Each task has also the same prestige. The random node for the Network evolution phase is given by Newscast. Mechanism T6 always gives best results. Everyone gives good results when degree is 20.

is very low. Also T5 gives good results: this means that choosing the open team rewiring is good when the node selection criteria is based on skill and degree; on the other hand we can see how such criteria gives bad results when the selection criteria is based on the prestige. As the size of the intra-organisational networks increases (the network degree), all the algorithms improve. Figures 6.3 and 6.4 show similar results than figure 6.2. We can note how all the mechanisms behave in a similar way than in Env1 but T1 which performs worse, indicating that these algorithms are quite robust even when there is a high task dynamism. With all these experiments we have seen that T6 performs much better than the others. We think this is due to the fact that T6 gives incentives to keep the network as much connected as possible and gives to the nodes more intelligence than the others, allowing NTs to choose 'good' NSs and to have a larger set of NSs which will compete one another decreasing their acceptance threshold (β).

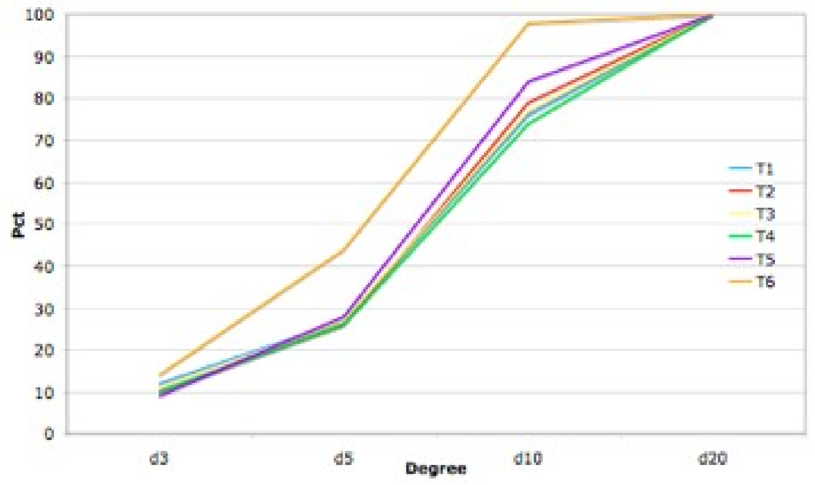


Figure 6.3: Percentage of completed tasks. Randomized Service 1. Env2, here different NTs receive different task. Every 20 cycles each NT receives a new task different from the previous one. Each task has the same prestige. Mechanism T6 always gives best results. Everyone gives good results when degree is 20.

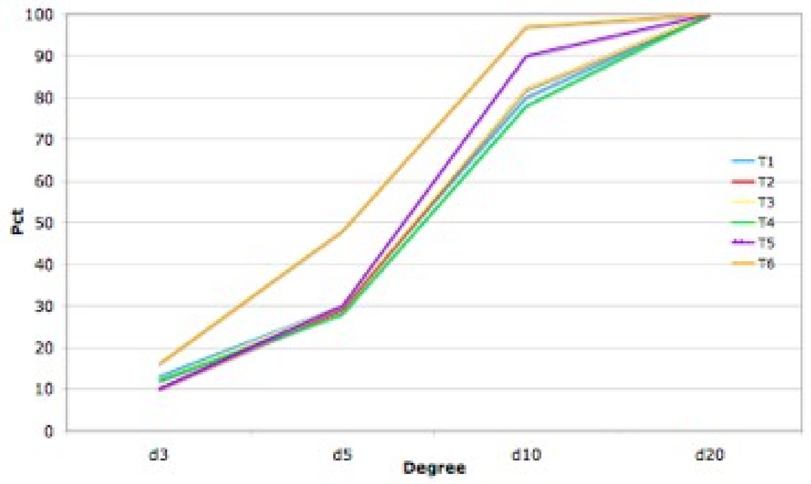


Figure 6.4: Percentage of completed tasks. Randomized Service 1. Env3, here different NTs receive different task. Every 20 cycles each NT receives a new task different from the previous one. To each task is associated a different amount of prestige and every 20 cycles also the payoff changes. Mechanism T6 always gives best results. Everyone gives good results when degree is 20

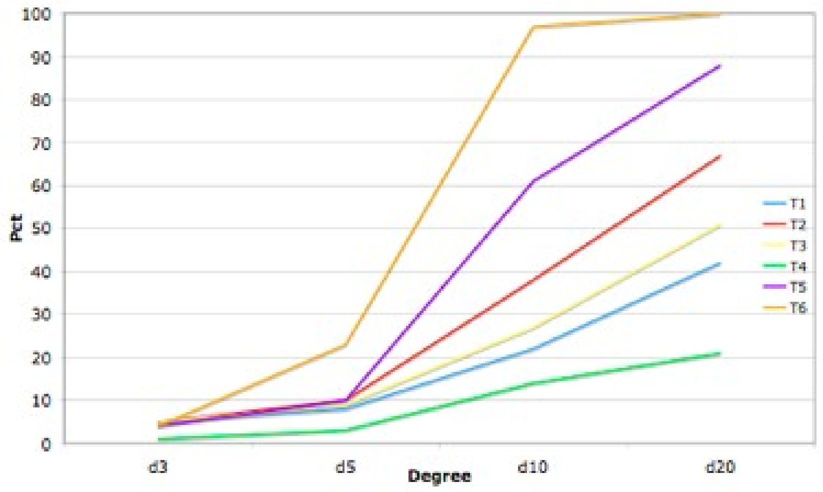


Figure 6.5: Percentage of completed tasks. Randomized Service 2. "Env1, here each NT in the Network receive the same task. Each task has also the same prestige. Mechanisms T6 gives good results which are similar than the case with Randomized Service 1. All the other mechanisms give worse results than the same with RS1

6.2.2 Randomized Service 2: results

This section describes the results obtained when the random node used by the SO algorithms is picked with the Randomized Service 2. This means that each node when executing one of the six network evolution algorithms (table 6.1) picks a random node directly from its current view so that the more links a node has, the more chances it has to join a new neighborhood. Figures 6.5, 6.6 and 6.7 show similar results even if they relate to different scenarios. We can note how when the network degree is low, 3 and 5, the performances are very bad. This is because when the degree is low, nodes may have no neighbors or just few neighbors, hence the rewiring phase may not take place or may not be able to form a good team. With the Randomized Service 1, the situation is different because even when a node has no links, it picks the random correspondent from the Newscast instance of its neighborhood which is always connected. With degree 10 and 20 the T6 algorithm gives very good performances, better than the other five algorithms.

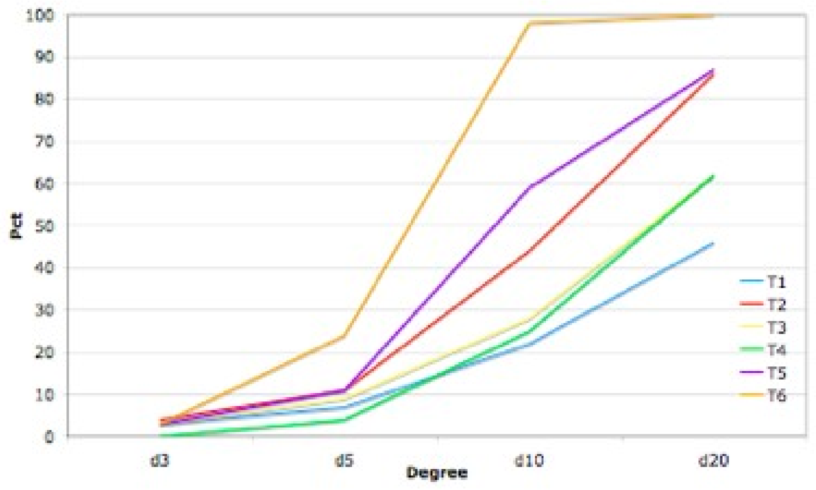


Figure 6.6: Percentage of completed tasks. Randomized Service 2. "Env2, here different NTs receive different task. Every 20 cycles each NT receives a new task different from the previous one. Each task has the same prestige. Mechanisms T6 gives good results which are similar than the case with Randomized Service 1. All the other mechanisms give worse results than the same with RS1.

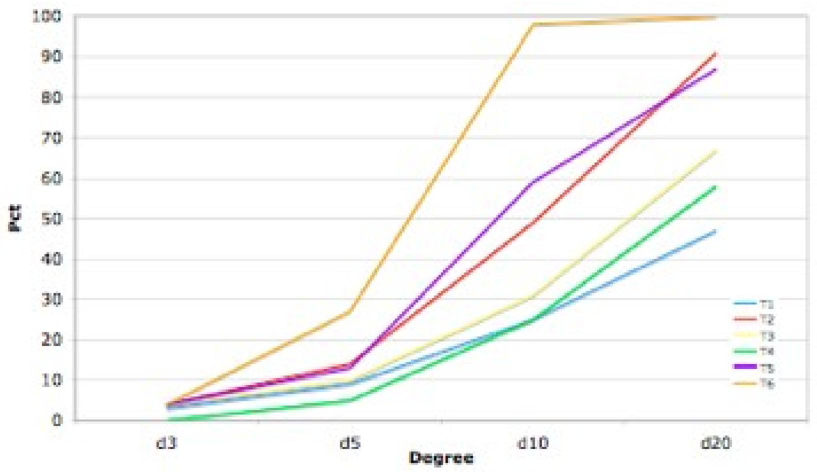


Figure 6.7: Percentage of completed tasks. Randomized Service 2. Env3, here different NTs receive different task. Every 20 cycles each NT receives a new task different from the previous one. To each task is associated a different amount of prestige and every 20 cycles also the payoff changes. Mechanisms T6 gives good results which are similar than the case with Randomized Service 1. All the other mechanisms give worse results than the same with RS1

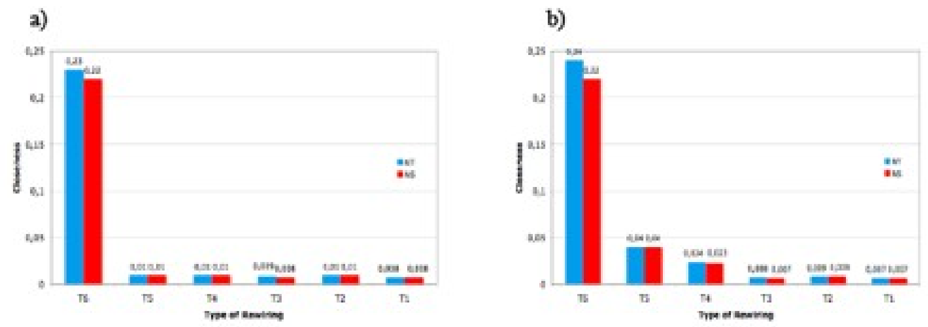


Figure 6.8: Closeness centrality calculated on the network at the end of the simulation with “Env2. Network degree is 5. a) Randomized Service 1. b) Randomized Service 2. Every 20 cycles each NT receives a new task different from the previous one. Each task has the same prestige. Mechanism T6 gives the highest values.

6.3 Structural Analysis

So fa, one of the main findings of our work, has been that the structure of the network matters. We found that as the network degree increases, performances of the firm increases. We also noted that the rewiring mechanisms performing better, are those which tend to keep the network as much connected as possible. Thus we decided to do some structural analysis on our networks. We calculated the “closeness centrality every 10 cycles for all our experiments. The results that we obtained are very similar both with the RS1 and with RS2 . We found that for the first 20 cycles, values are always quite high but then they start decreasing and stabilize after few cycles. Figure 6.8 show the results on the network at the end of the simulation (Env2, degree 5); we can note how with T6, the closeness value is much higher than with the other algorithms, this is because here nodes never isolate before adding new links. For all the simulations that we did, we also checked if some “cut vertex [2] appeared during the computation: we found that they never appeared (except for some isolated cases), meaning that the groups formed by our algorithms are always well connected.

[2]A cut vertex is a vertex of a graph such that removal of the vertex causes an increase in the number of connected components

6.4 Conclusion

From the simulation experiments seen in this chapter, emerge a number of facts that delineate areas for further investigation. The first fact which is interesting to highlight is that self-emerging knowledge integration mechanisms originate within organizations and allow firms to coordinate their skills and respond to environmental demand. In addition, in some circumstances, these knowledge integration processes are enough robust to respond to a turbulent environment in which a firm is required to complete tasks that change in regard to both the repertoire of skills needed and their relevance within the organization. Thus, the complex problem of integrating specific skills in response to environmental dynamics rather than requiring top down problem-solving abilities to dynamically partition into teams, calls for the analysis of what organizational context appropriately moulds individual incentives, bridge specialists and convey information. The second fact is that emergence of integration mechanisms is associated to the structure of intra-organizational social networks within which individuals interact. As shown in the graphs reporting simulation experiments, in all our simulations, the higher the connectedness of intra-organizational social networks, the higher the percentage of task completed by a firm and, when each node is able to maintain links with other 20 nodes, self-organized integration mechanisms produce same performances as an hypothetical central authority which has enough authority to impose top down to an individual to move from one position to another one within an intra-organizational social network. A third fact is that individual attitudes and mobility within a social network are key element to produce knowledge integration. In particular, in our work, we compared two individual attitudes. The first attitude characterizes individuals that are extremely mobile and easily terminate all previous links to connect to other nodes; the second attitude characterizes individuals that are willing to make links with new nodes, without losing the good links with old nodes. In general, simulation experiments show that the first attitude has worse performance. The fact that a node makes new links, not necessarily indicates that all the old ones are useless; on the other hand, it is not always possible to keep old links, sometimes moving may indicate the physical movement of thousand of kilometers which inhibits a possible collaboration among past neighbors. However, the model we presented is a very simplified one: we assume that project managers are able to appropriately associate task content to skills. This is not always true, mainly if it is about intellectual task. To cope with this problem, we though of introducing a different categorization of the skills, developing a version of our model in which managers chose employees on the base of the category to which their skill belongs. We also gave employees the ability to learn skills which will be added to those they already have. This process will be described in the next chapter.

Chapter 7

Introducing Learning to FirmNet

In this chapter we introduce a new version of our FirmNet model (Mollona and Marcozzi, *submitted* [69]). Here nodes in the network have the ability to learn some particular kind of skill. We first give an introduction on this new model highlighting the motivations, then we describe our model and finally we report the results of the experiments we conducted.

7.1 Introduction

The model seen in the previous chapters is a very simple one: we assume that project managers are able to appropriately associate task content to skills. Therefore both task and skill content is not ambiguous. Yet, in ambiguous task environments, this may be a strong assumption and project managers may face problems in connecting task content to required skill and to interpret skill content. In addition, if task content is particularly innovative, it may be possible that skills are not immediately available but have to be built. In other words, professionals may hold skills that are not exactly fit to the need but are sufficiently close to those required. These specialists may make an effort to learn how to deploy their skills in order to appropriately cope with required tasks. However, such dynamic fit between skills and task require learning both on the part of specialists and project managers. The described learning process, however, entails a dilemma to deal with. On the one hand, our previous experiments (Mollona and Marcozzi, 2008 [68, 66, 67] – see previous chapters) suggest that for a network of specialists to be able to dynamically respond to evolving tasks it is required that the nodes of the network are able to frequently rewire and to hold an high number of connections, on the other hand, the previously described learning processes require stability and continuity of collaboration with specific nodes. In general, computer simulation is a useful method to understand dynamical process regarding learning (e.g. Carley, 1992 [14]) In this light, the work described in this chapter addresses the mechanisms and the circumstances that better handle the dilemma between dynamic plasticity of an intra-organisational network and the continuity of association among specific nodes required to enact learning processes.

7.2 Learning in FirmNet

The core of our model, remains the same that we have seen in chapter 4. Also here we have an interaction phase, a network evolution phase and an imitation phase. Actually some changes have been done to the interaction phase and we will see them soon. In this new version we also thought of introducing a different categorization of the single skills. So far we have considered a system where each agent could have just one single skill and this skill could vary from 1 to 5. Here we enlarged the set of skills (from 1 to 9) and we considered three different categories in which each skill can fall (see table 7.1).

Table 7.1: Skills' categories

Category	**Skills**
A	1 2 3
B	4 5 6
C	7 8 9

Hence each skill belongs to a category which can be A, B or C. A category can be seen as a particular area grouping certain skills. Category A for example can be a degree in Computer Science and skill 1, 2, 3 can relate respectively to expertise in programming, network administration, web designing. In real life when a particular task is too specific, the manager is not always able to identify the more suitable employee and generally it can assign the job to someone who is only related to that area and doesn't have the exact skill. This person of course has more capabilities to learn the skill than someone that comes from a totally different area.

Coming back to our model, when a NT receives a task, it just knows the categories of the three skills needed for the task, hence when in the interaction phase it selects neighbors, it can base its choice only on the category of the skill held by the NS. Even if the selected node has a skill of the right category, it may not be the required one; in this case the NS has two possible options:

- NS can decide to learn the skill: after a number of cycles (20) it learns the skill which is added to those he already holds. During the 20 learning cycles, the node remains idle and cannot move or accept jobs from other NTs;
- NS can decide not to learn the skill: in this case it may ask (with a certain probability) one of its neighbors to work on such job.

Figure 7.1 shows an example. Each node decide to learn the skill according to its "attitude at learning" (γ); if it doesn't learn, it can decide to pass the job to a neighbor according to its "attitude at passing" (δ). Both γ and δ are initialized at random with a value from 0 to 1.

As for the network evolution phase, we use the same 6 mechanisms we have seen in the previous chapter (see section 6.1.2 and table 6.1). The same happens with the imitation phase (see section 6.1.3).

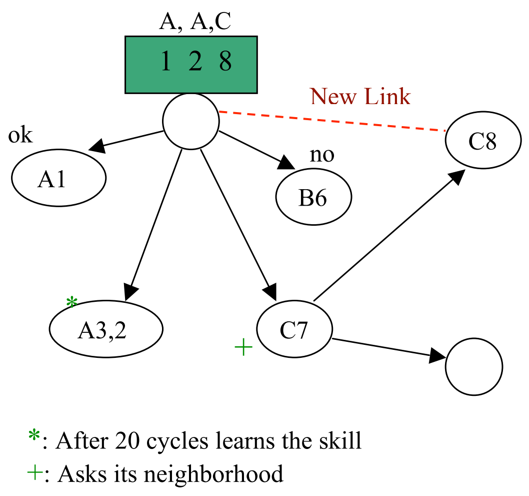


Figure 7.1: Interaction phase with the learning mechanism: node "*" learn the skill; node "+" passes the job to its neighbor who will make a link to the NT.

7.3 Experimental setting

We performed several experiments with this new model, comparing the six rewiring mechanisms that we indicated in table 6.1. Experiments proposed here have been done using the "Randomized Service 1" for the selection of the node in the rewiring phase (this is the usual peer sampling service that we have used in several situations, see section 6.2). The configuration we adopted for the simulations is indicated in table 7.2. For each experiment we performed 10 runs and then took the average. The standard deviation is always very low.

With this configuration we propose five sets of experiments which will be shown in the next subsections. The first is called 'Learning', in which we study the new Learning model comparing the performances of the six SO mechanisms seen in the previous chapter (table 6.1); the second is called 'No Leaning': here we replicate the same experiments proposed in chapter 6 comparing the SO model (without learning) with 2 different HI models, this time using 9 skills in the system instead of 5 as in chapter 6; the third is called 'Evolutionary Learning': the model is the same tested

Table 7.2: Basic configuration

Parameter	**Value**
Newtork size	1000
Number of NTs	250
Number of NSs	750
Number of cycles	500
Skill	$S \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
Commission (NT)	$\alpha \in \{0 \ldots 1\}$
Accept threshold	$\beta \in \{0 \ldots 1\}$
Attitude at learning	$\gamma \in \{0 \ldots 1\}$
Attitude at passing	$\delta \in \{0 \ldots 1\}$
Learnin time	20 cycles
Task environments	Env1, Env2, Env3 (see sec. 6.2)

in ‘Learning’ but here we make the parameters γ and δ evolve according to the wealth of the nodes; the forth is called ‘Lower degree attachment’: here NSs during the rewiring phase prefer attaching to NTs with low degree; finally the last set of experiments is called ‘Wealth performances’ in which we show the average wealth of NSs and NTs in different situations.

In the last three sets of experiments, we also compare two different reward policies: *individual reward* versus *group reward*.

7.3.1 Learning

The first set of experiments that we did, was directed to compare the behavior of the six rewiring mechanisms in a situation in which nodes can learn the skill or pass the job as we described in section 7.2 . Figures 7.2, 7.3, 7.4 show the results of the simulations in the three environments that we take into consideration (see

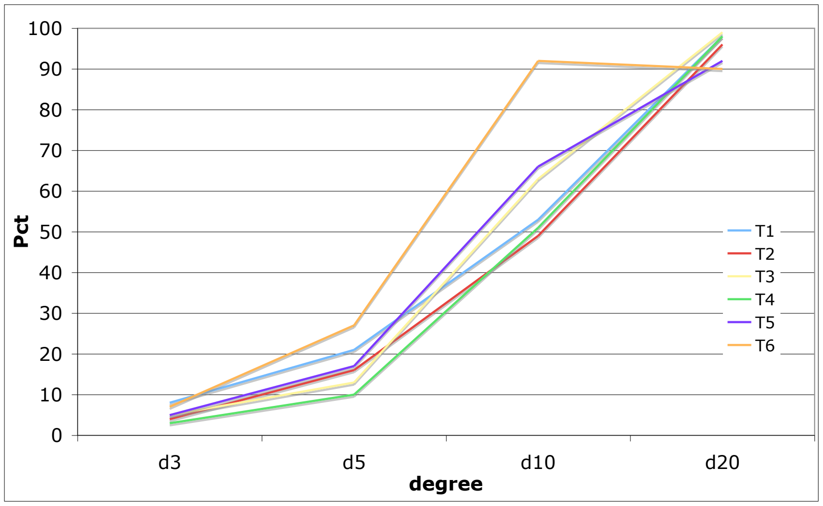


Figure 7.2: Percentage of completed tasks with “Env1: here each NT in the Network receive the same task. Each task has also the same prestige. NSs which don't have the right skill may learn the skill or pass the job according to their strategies (γ, δ).

section 6.2 for the description of the three task environments). We can note that also here T6 gives the best performances in all the task environments. In Env1 (static environment), it's interesting to note how the second one which performs better is T5 meaning that in an environment in which there is scarcity of resources, having a well connected network helps even if the choice of the nodes is not based on the individual preferences. In Env2 and Env3 instead, the second best performing mechanism is T3, indicating that the accurate choice made by the single nodes is preferable when the task environment is dynamic.

7.3.2 No Learning

In order to see if this new Learning model performs better, in terms of Pct, than the classical SO model, we decided to compare the previous results with the same 6 mechanisms without the learning process. For doing this we re-executed the same

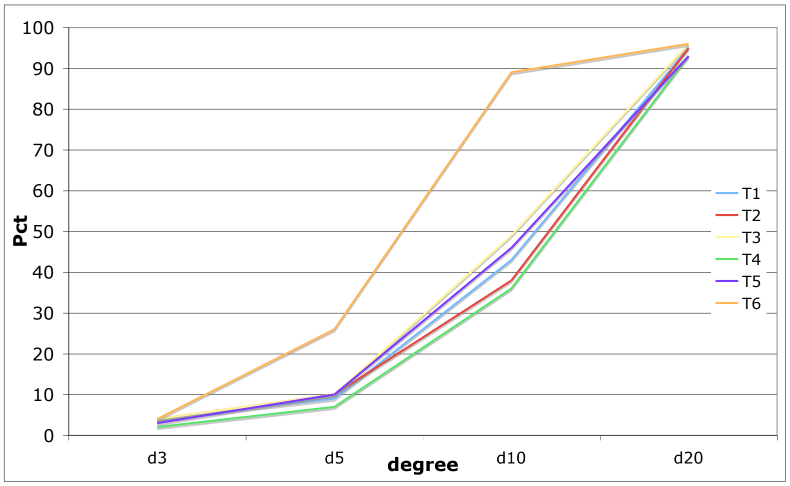


Figure 7.3: Percentage of completed tasks with "Env2: here each NT in the Network receive a different task which changes over the time (every 20 cycles). Each task has also the same prestige. NSs which don't have the right skill may learn the skill or pass the job according to their strategies (γ, δ).

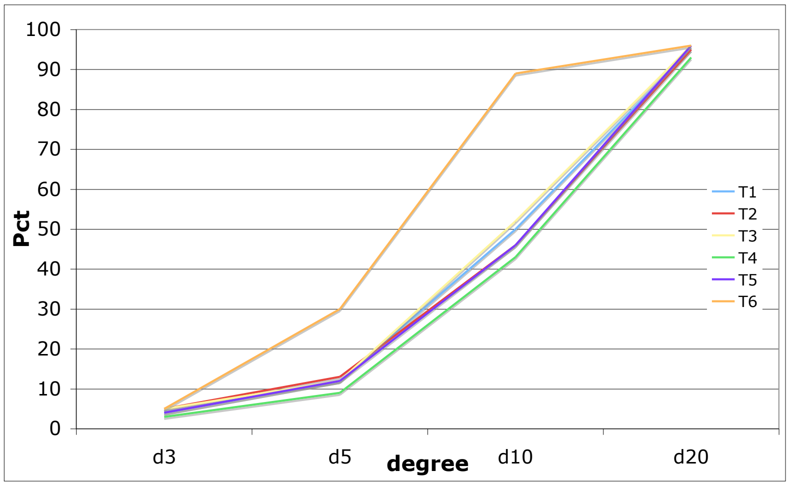


Figure 7.4: Percentage of completed tasks with "Env3: here each NT in the Network receive a different task which changes over the time (every 20 cycles); also the prestige assigned to each task changes every 20 cycles. NSs which don't have the right skill may learn the skill or pass the job according to their strategies (γ, δ).

experiments of the previous chapter but this time with 9 skill in the system. We also compared them with two "hierarchical approaches": we called them G1 and G2. The aim was to use the two Hierarchies as a benchmark for the SO model. Moreover, since we increased the number of skills in the system from five to nine, we wanted to see if this change influenced the performances of the hierarchy. In both hierarchies during the interaction phase NTs ask their immediate neighbors to provide their skill; they can refuse only if busy. NSs get a fixed wage every 20 cycles, NTs are paid only when the entire task is completed. A NS gets free again only when the entire task is completed or every 20 cycles when a new task in assigned and NS are paid. The difference between the two hierarchies is in the network evolution phase:

- G1: periodically NTs pick a random node from the entire network and if this node has the right skill for working on the task and is not busy, a link to him is made (its the same mechanism we have seen in chapter 5);
- G2: periodically NSs check if their skill is currently needed by the NTs to whom they are eventually linked; if not they drop the link to them. After this they make a link to a randomly selected NT in the network. Payment in G2 is the same than in G1: NSs get a fixed wage every 20 cycles, NTs get the payoff only when the entire task is completed.

Figure 7.5 refers to Env1: surprisingly we find that T1 gives similar results than T6 meaning that in a situation in which resources are scarce[1], basing the rewiring on the skill doesn't give much help. Interestingly we can note how the two hierarchical mechanisms don't give good results. It seems that when the number of available skills is high (bigger than 5), hierarchy is not able to manage it. We must point out that Env1 is different from the "static environment" that we tested in chapter 5: here the task is identical for all the NTs, while in chapter 5 we had different

[1]When the task is totally static and composed always by the same 3 jobs, a high number of possible skills (9) leads to a situation of scarce resources because skills are equally distributed in the system and hence there are not enough skills equal to the 3 requested.

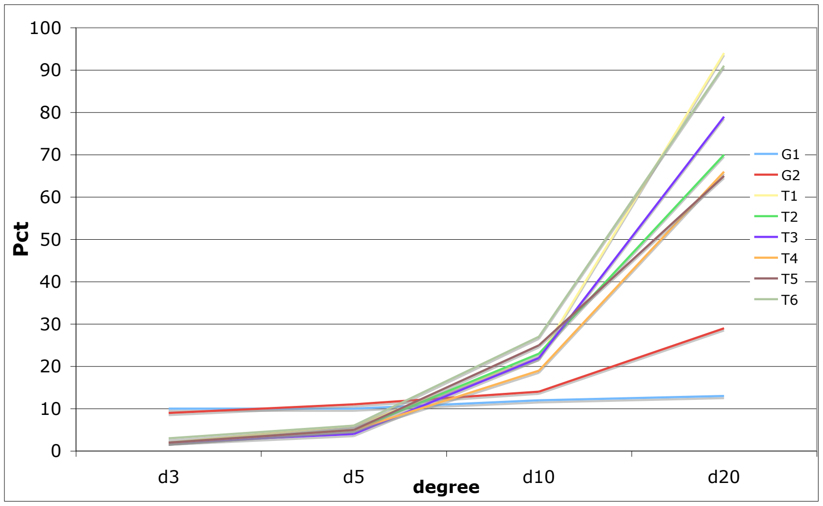


Figure 7.5: Percentage of completed tasks with "Env1: the lines in this plot show the six SO mechanisms and the two HI mechanisms. In no one of these learning attitude or passing attitude are considered.

tasks for different NTs which does not changes over the time. This new situation creates a scarcity of resources from which the two hierarchies are not able to recover. Moreover with hierarchical mechanisms, when a NS gets busy, he will be free again only when the entire task is completed, hence if he is stuck because the NT dosn't find the remaining NS, he cannot go working for another NT. Hence skills, which are not enough, are also idle more often than with the SO mechanisms. On the other hand, when network degree is high, SO mechanisms performs better because a certain NSs, with "good" skills, have the possibility to move among different teams and then satisfy different tasks.

Figure 7.6 refers to task environment Env2: hierarchies performs better than before but they are not still giving good results. Here the task environment is dynamic: tasks are different for each NT and continuosly changing every 20 cycles. Here we don't have scarcity of resources because the high dynamism of the tasks makes the most of the skill involved and hence the two hierarchical algorithms performs better, but worse than HI in previous work with 5 skills (see chapter 5). In

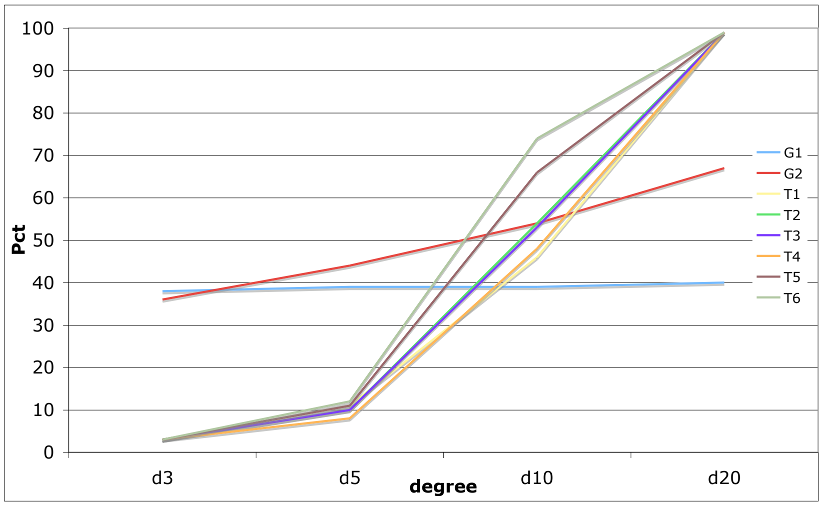


Figure 7.6: Percentage of completed tasks with “Env2: the lines in this plot show the six SO mechanisms and the two HI mechanisms. In no one of these learning attitude or passing attitude are considered.

particular G2 gives good performances: here NSs have the ability to drop the links to NTs which don't need their skill. In a dynamic environment is important that NSs have this ability in order to become always available for other NTs. This particular feature of G2, limits the idleness of NSs created by hierarchical algorithms. With G1 instead, NTs have the ability to pick the useful node which are not released (and then made available for other NTs) when not needed anymore. SO algorithms performs much better than in Env1: the situation of not scarcity of resources and the fact that NSs have the possibility to move among different groups and then satisfy different tasks, more frequently than with HI, are the reasons of these good performances.

Figure 7.7 represents the results with task environment Env3. We can note that they are very similar to Env2 meaning that changing the payoff values doesn't make a big difference.

Comparing the pictures of the results with learning (section 7.3.1) with those without learning (section 7.3.2, we can also find that the learning mechanism plus

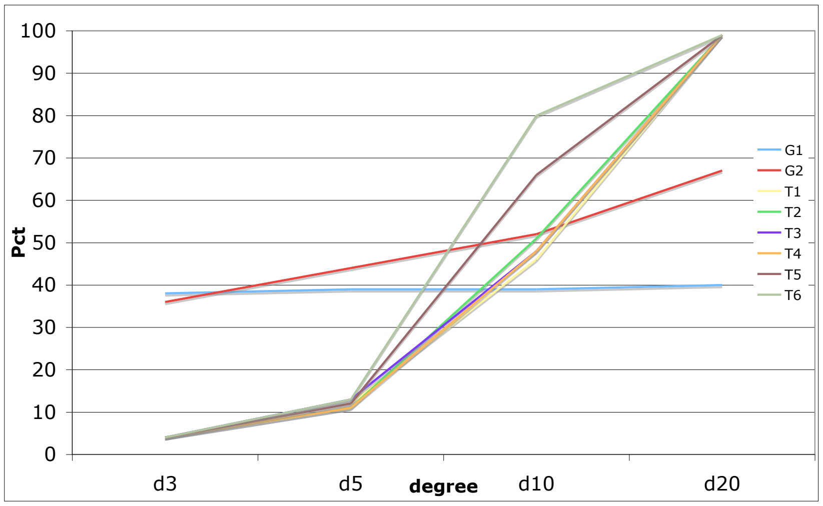


Figure 7.7: Percentage of completed tasks with “Env3: the lines in this plot show the six SO mechanisms and the two HI mechanisms. In no one of these learning attitude or passing attitude are considered.

the passing mechanism gives some improvements in terms of Pct, hence when the set of skills is large this new approach may be a good choice.

7.3.3 Evolutionary Learning

A typical behavior is that of imitating the strategies of people performing better. We already explored this with the rewiring mechanisms T1 and T4 for the teams formation. In this section we want to explore what happens if nodes copies their attitude at learning (γ) and passing (δ) according to their wealth. We introduced a mechanism with which nodes periodically selects a random node (via the usual peer sampling service) and then copy its γ and δ if the selected node is richer. Hence in the learning model we analyze in this section, nodes have the ability to learn a skill, pass a job to a neighbor and imitate (γ) and (δ) values of richer nodes. We performed experiments adopting two different reward policies: an “individual reward” policy and a “group reward” policy.

- **Individual reward**: NSs get their reward soon after they complete a job; the NT will get the payoff only when the entire task is completed; NSs gets busy when recruited but becomes free again as soon as their single job is completed (all the SO algorithms that we have seen so far adopt this reward policy).

- **Group reward**: NSs get their reward only when the entire task is completed; the same happens to the NTs; NSs gets busy when recruited and will become free again only when the entire task is completed.

Figures 7.8, 7.9, 7.10 show the results with the "individual reward" policy and the evolution of γ and δ according to the wealth of the nodes. Figures 7.8 refers to task environment Env1. As we said before, Env1 represents a situation in which the resources to allocate are scarce. We can note that here the SO mechanisms are able to perform better than before because nodes have both the ability to learn new skills (involving a less scarcity of resources) and the ability to pass the job to their neighbors. Algorithm T6 is the one which performs better: with it, we give to nodes a great intelligence, indeed they can decide to make new links on the basis of their individual needs or preferences.

Figure 7.9 indicate performances of the algorithms in "Env2". Here the task environment is more dynamic: tasks are different for each NT and continuosly changing every 20 cycles. Here we don't have scarcity of resources because the high dynamism of the tasks makes the most of the skills be involved. Our six SO mechanisms, here perform worse than in Env1: when a node decides to learn a new skill, he becomes idle for 20 cycles; this may be a problem because every 20 cycles tasks change and the new task may be of a different kind then the one just learned by the NS, hence the leaning effort may result useless. Something similar happens with Env3 (figure 7.10) where the task is dynamic and the reward is dynamic too.

Figures 7.11, 7.12, 7.13 show the results with the "group reward" policy and the evolution of the γ and δ. Something different happens when a "group reward" policy is adopted. In Env1 (figure 7.21), performances are very similar than with the individual reward policy; they are a bit better when degree is small (d3, d5): we think that when degree is small, nodes must cooperate more in order to complete the task (and hence in this case get their payoff), mainly in a situation of scarce resources. Some experiments have highlighted that in this situations the δ value generally increases entailing a kind of cooperative behavior of the nodes.

In Env2 (figure 7.12) , results are different: here the dynamic environment favors the individual reward policy. With the group reward policy, nodes are paid and become available only when the entire task is completed, hence if a NS is busy and its NT doesn't complete the task, it will not be available for other NTs (with the individual reward, nodes are never busy for long time, they are soon available for other NTs). In addition the fact that nodes can learn, adds more idleness in the system, since when a node is learning is idle for 20 cycles. This idleness doesn't help in dynamic task environments. Similar results can be seen in Env3 (bottom of figure 7.13).

We also compared the results obtained when γ and δ don't evolve with those obtained when they evolve and individual reward or group reward is applied. Figure 7.14, refers to the results with task environment Env1. In general we note that results are better when copying of the 'attitudes' takes place. Its interesting to note that often the group reward mechanism gives better results: group reward makes teams more stable, a node cannot move if it has not been paid, hence in a static task environment this seems to give better results.

Figures 7.15 and Figure 7.16 show the results related to Env2 and Env3. Here we can note that in general is the individual reward performing better: in a dynamic task environment more flexibility is preferred and the constrains given by the group reward policy (idleness of NSs until the total task is completed) doesn't help.

7.3.4 Lower degree attachment

An important finding of our previous work (Mollona and Marcozzi, 2008 [67]), is that if nodes prefer attaching to nodes with high degree (large number of actual connections), those nodes could benefit from this having for example more possibility to be recruited if among those links there were some to other NTs. But what happens

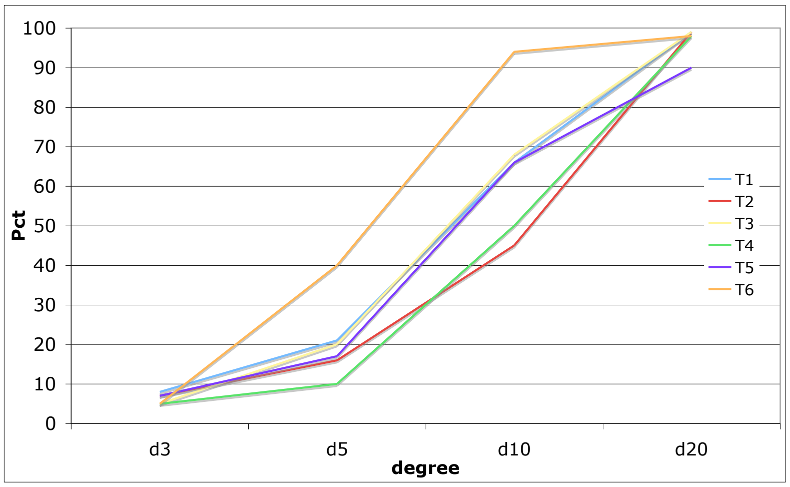


Figure 7.8: Percentage of completed tasks in "Env1. The lines in this plot represents the Pct obtained with the six rewiring mechanisms. Here each NS may learn a skill or may pass a job to its neighbors. Nodes are paid according to the individual reward policy.

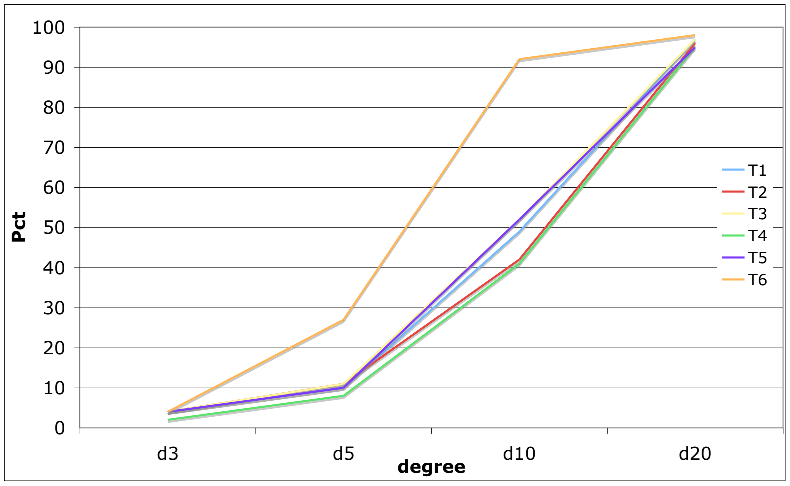


Figure 7.9: Percentage of completed tasks in "Env2. The lines in this plot represents the Pct obtained with the six rewiring mechanisms. Here each NS may learn a skill or may pass a job to its neighbors. Nodes are paid according to the individual reward policy.

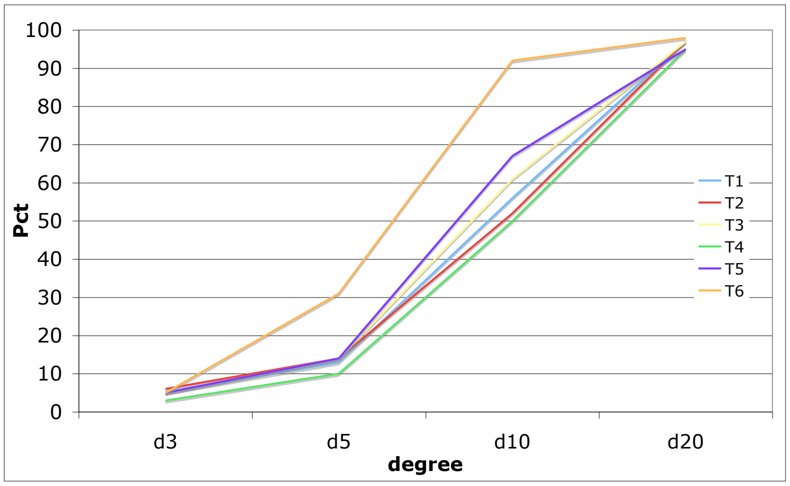


Figure 7.10: Percentage of completed tasks in “Env3. The lines in this plot represents the Pct obtained with the six rewiring mechanisms. Here each NS may learn a skill or may pass a job to its neighbors. Nodes are paid according to the individual reward policy.

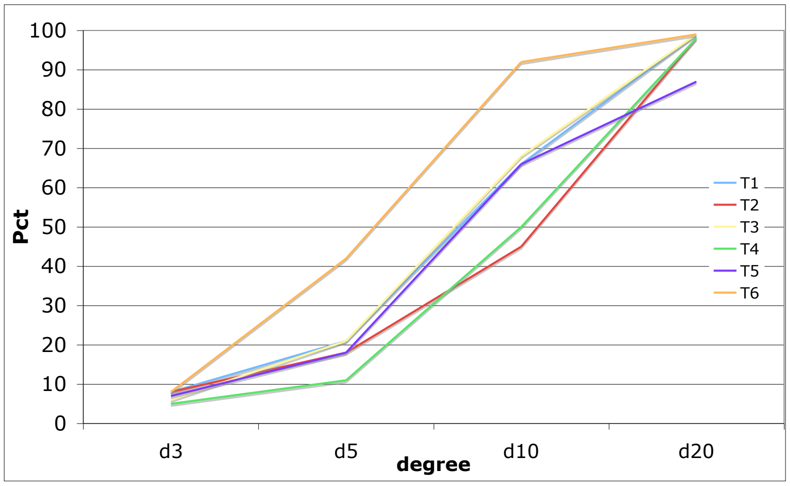


Figure 7.11: Percentage of completed tasks in “Env1. The lines in this plot represents the Pct obtained with the six rewiring mechanisms. Here each NS may learn a skill or may pass a job to its neighbors. Nodes are paid according to the group reward policy.

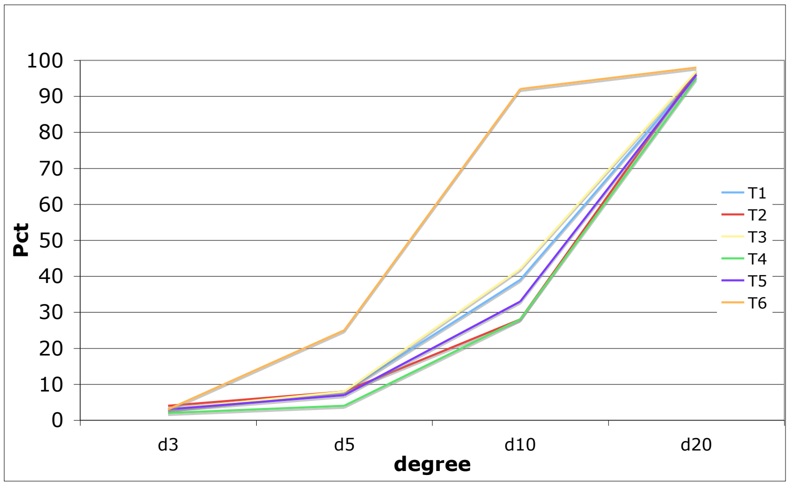


Figure 7.12: Percentage of completed tasks in "Env2". The lines in this plot represents the Pct obtained with the six rewiring mechanisms. Here each NS may learn a skill or may pass a job to its neighbors. Nodes are paid according to the group reward policy.

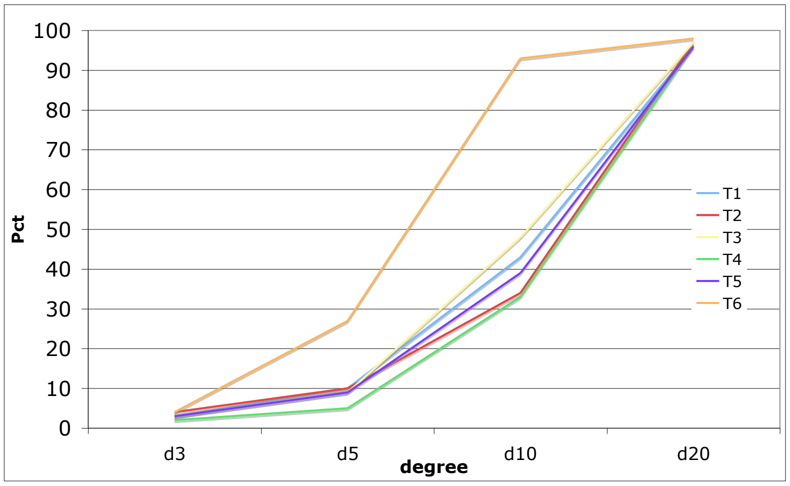


Figure 7.13: Percentage of completed tasks in "Env3. The lines in this plot represents the Pct obtained with the six rewiring mechanisms. Here each NS may learn a skill or may pass a job to its neighbors. Nodes are paid according to the group reward policy.

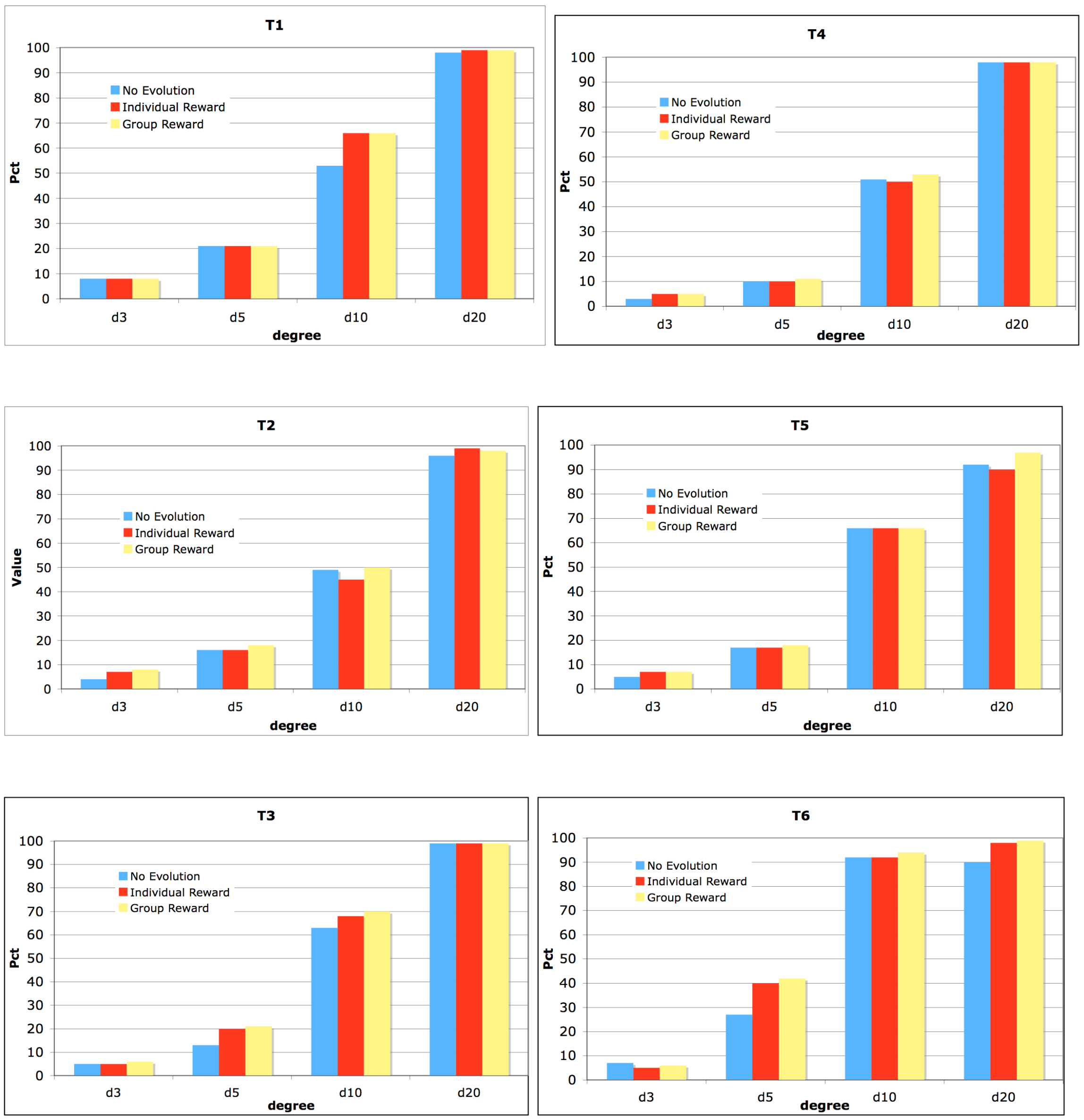


Figure 7.14: Percentage of completed tasks in "Env1. This figure represents what happens for each single rewiring mechanisms, comparing the Pct obtained when "no evolution" takes place, when "evolution" takes place, in this last case with both individual and group reward policy.

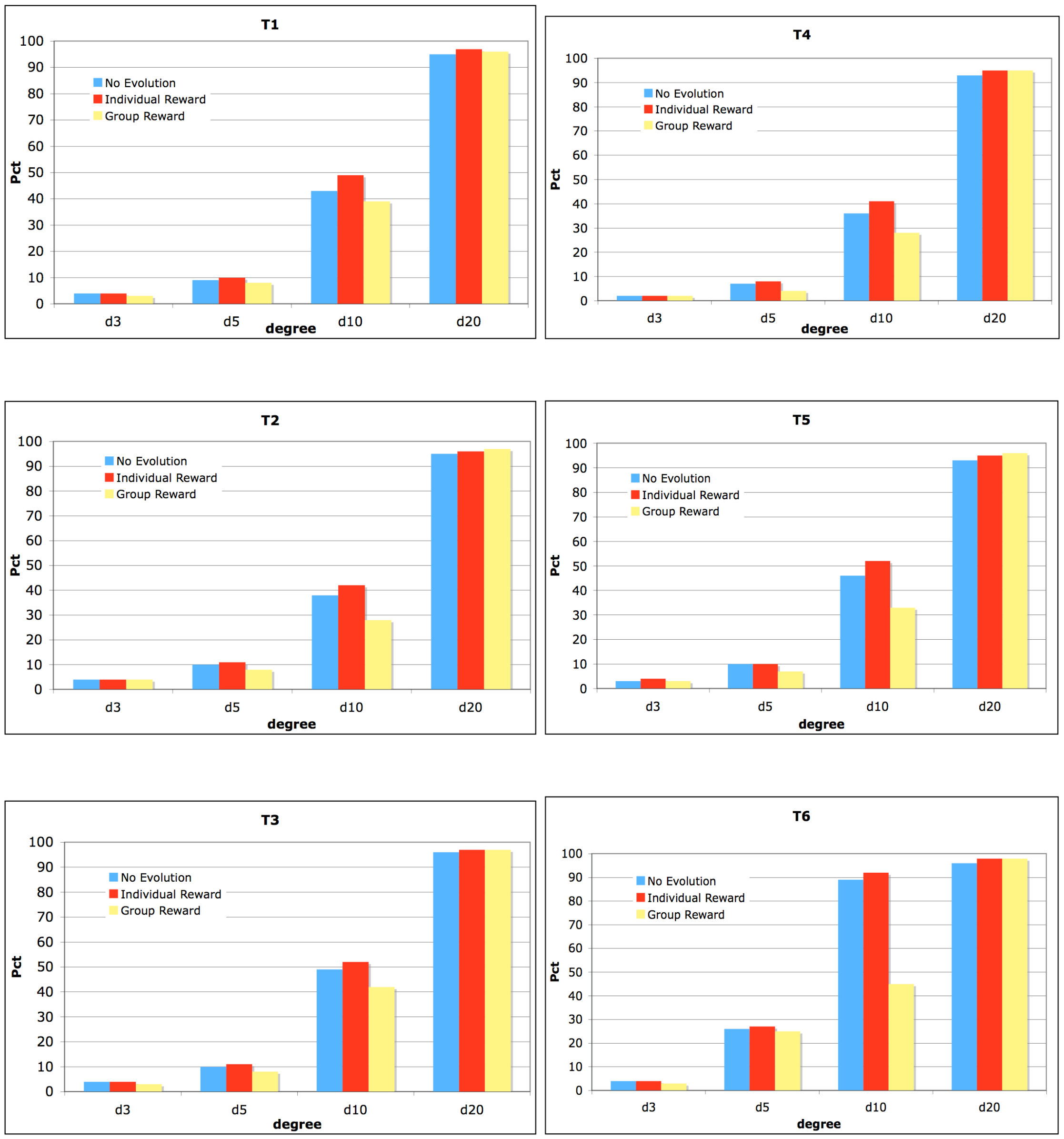


Figure 7.15: Percentage of completed tasks in "Env2. This figure represents what happens for each single rewiring mechanisms, comparing the Pct obtained when "no evolution" takes place, when "evolution" takes place, in this last case with both individual and group reward policy.

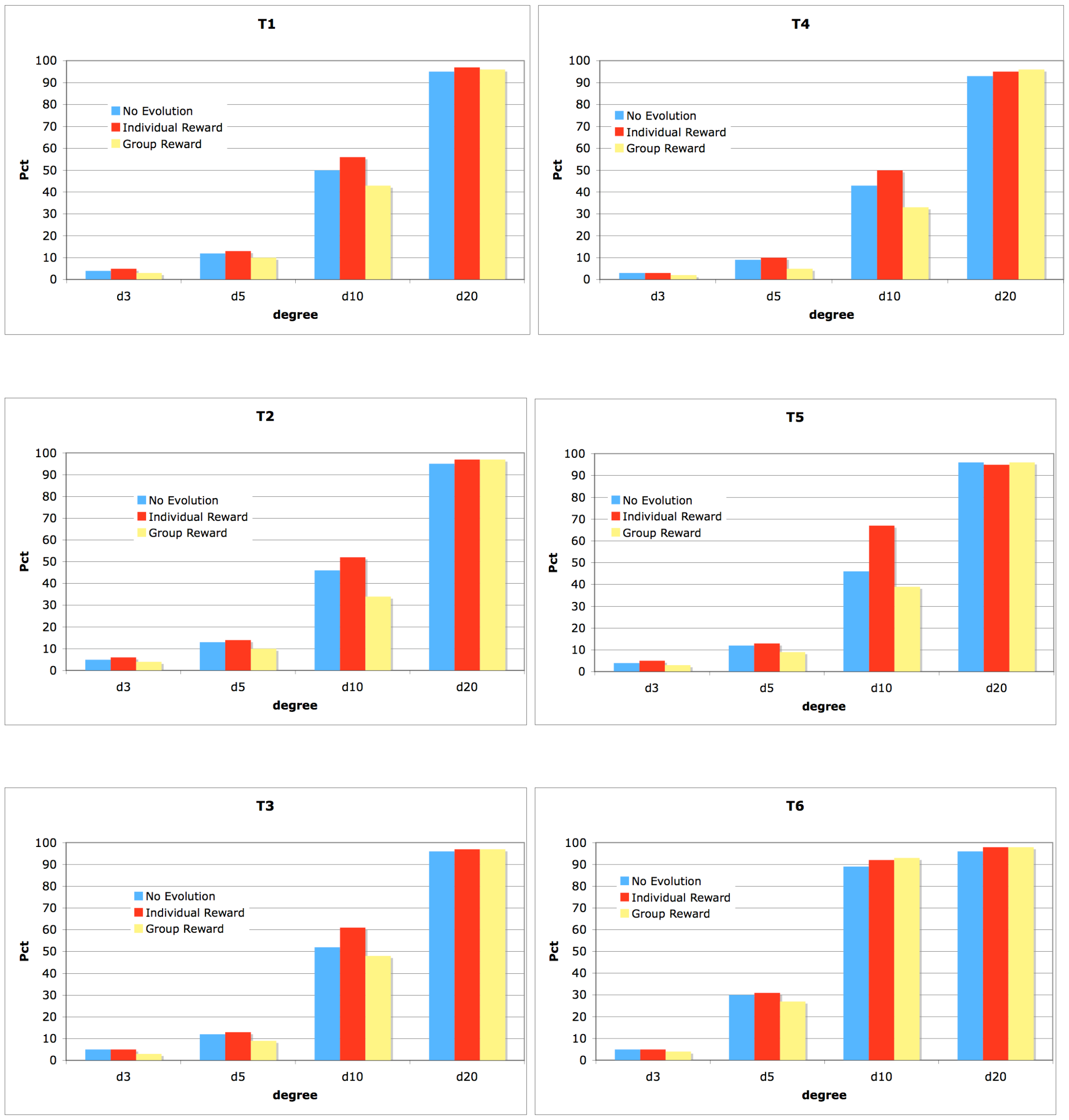


Figure 7.16: Percentage of completed tasks in “Env3. This figure represents what happens for each single rewiring mechanisms, comparing the Pct obtained when “no evolution” takes place, when “evolution” takes place, in this last case with both individual and group reward policy.

if NSs decide to link to NTs with low degree? In theory this may be a good move because if a NS links to a NT with low degree, the NS will have more chances to be recruited and NTs with scarce links will have more chances to complete the task. We implemented a new mechanism called T7 in which NSs have the ability to select a random NT (with the peer sampling service) and if this NT has a degree smaller than the degree of the NT to which it is currently linked, NS will drop the current NT and will connect the the new NT.

Experiments have shown results different from what we expected: figure 7.17 compares the performances of T6 and T7 with different reward policies (IR, GR) in the three task environments (Env1, Env2 and Env3). In Env1 (top of figure 7.17) we note that T6 gives good results both with the group reward and individual reward policy. We think that T7 gives worse performances because the algorithm is too simple: in T7, NS just link to NTs with low degree but they don't know anything about the real needs of the NT; on the other hand, with T6 nodes have more advantages and more intelligence, hence they are more able to adapt the structure of the network to the task environment. What is interesting from Env1 is the difference of Pct between T7 with IR and T7 with GR. T7 with GR gives better results in particular with d10: we think this may be due to the fact that with IR, nodes are free to move as soon as they have completed their job, hence it may happen that a NS who has just completed a job moves to another NT only because this second NT has a lower degree; this second NT may not actually need such NS and the previous NT will not be able to complete the same task anymore during the next round. On the other hand with GR, nodes have less chances to move toward 'wrong' NTs, because they will result busy during all the time needed for the completion of the entire task.

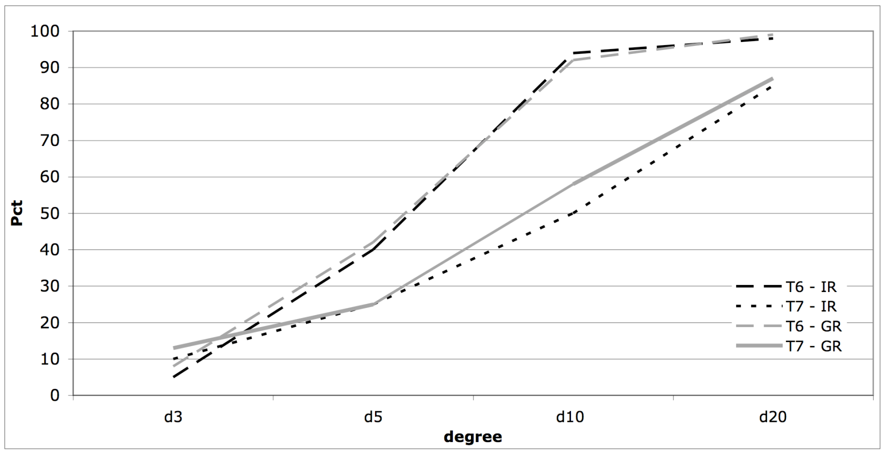
In Env2 and Env3 (center and bottom of figure 7.17), results are similar to Env1 but we note that there is not much difference between T7-GR and T7-IR: we think this is due to the fact that in Env1, if a NT needs a certain NS, it will need its forever, so if it leaves for linking to a NT which doesn't need it, this decreases the global performances; in Env2 and Env3 instead, a NTs don't need certain skills forever, hence moving after providing the skill, is not always a problem.

7.3.5 Wealth performances

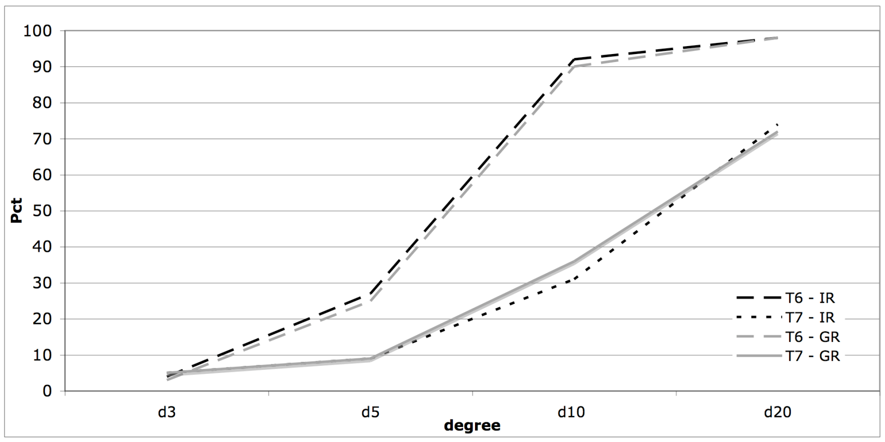
The last set of results that we show, relates to the performances of the nodes in terms of Wealth. We measured for mechanisms T6 and T7 the average wealth of NTs and NSs in all the three task environments and with both individual and group reward policies.

Figure 7.18 shows the average wealth of T6 and T7 with IR in task environment Env1. Interestingly we can note that with mechanism T6, the average wealth of NTs increases as the size of the intra-organizational network increases (degree), while the average wealth of NSs increases until degree 10 and decreases when degree is 20. This trade-off is more evident when task environment becomes dynamic as in figures 7.19 and 7.20. When degree is 20, the Newscast protocol (which is the one providing the peer sample in the network evolution phase), gives a view of the whole network; this global view, plus the high number of links for NTs, gives NTs the possibility to be linked to the highest possible number of "good" NSs. This thing, for NTs means that they have a lot of neighbors among which to choose the cheapest NS for the completion of the task; for NSs means that they have lot of competitors with which to compete for a job, hence they have to lower their β which results with an average lower wealth for NSs. Another interesting thing is that in all the three task environments, the average NSs wealth is higher with T7 than with T6 when degree is 3 and 20. Even though linking with NTs with low degree doesn't improve global Pct (for the reasons we explained above – section 7.3.4), this seems to be a good move for the NSs wealth: when degree is very low, NSs are linked to only very few NTs and hence there are more probability not to be useful for their task (and hence not be paid), hence the T7 mechanism helps them to free themselves more easily then with T6; when degree is 20, the fact that NSs links to NTs with less links than before, put them is a situation with less competition, which on one hand disadvantages NTs (in terms of number of task completed) but on the other hand advantages the NSs (in terms of wealth).

ENV1



ENV2



ENV3

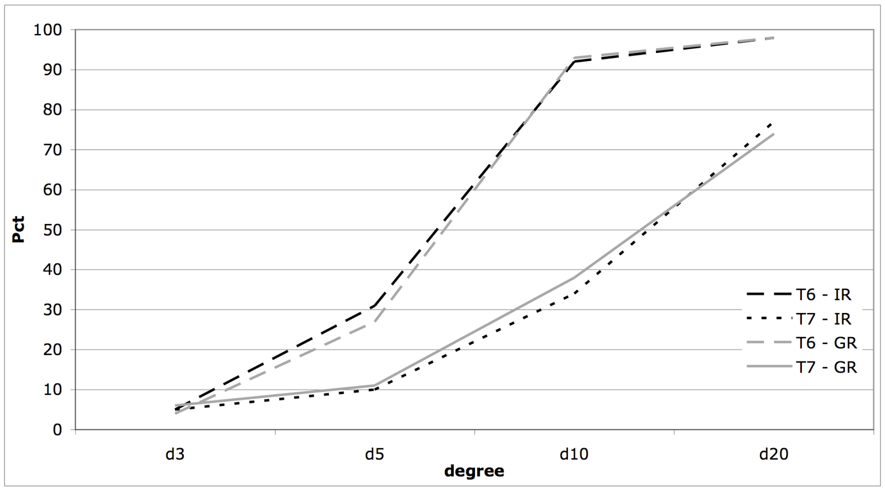


Figure 7.17: Percentage of completed tasks (Pct). The figure shows the performance of mechanisms T6 and T7 in three different task environment (Env1, top – Env2, center – Env3, bottom). In each plot are indicated behaviors of the two mechanisms with "individual reward" and "group reward" policy.

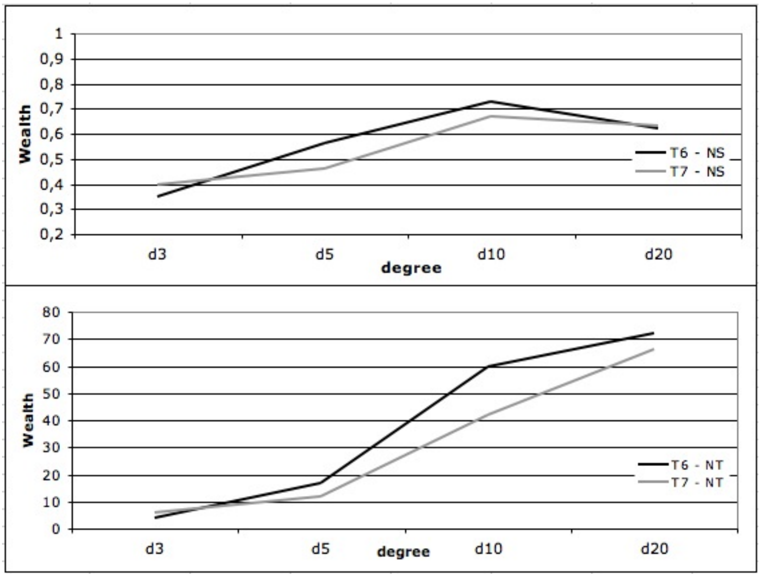


Figure 7.18: Wealth in Env1. Comparison of the average wealth of NS (top) and the average wealth of NT (bottom) with the T6 and T7 mechanisms. Here an "individual reward policy" has been used.

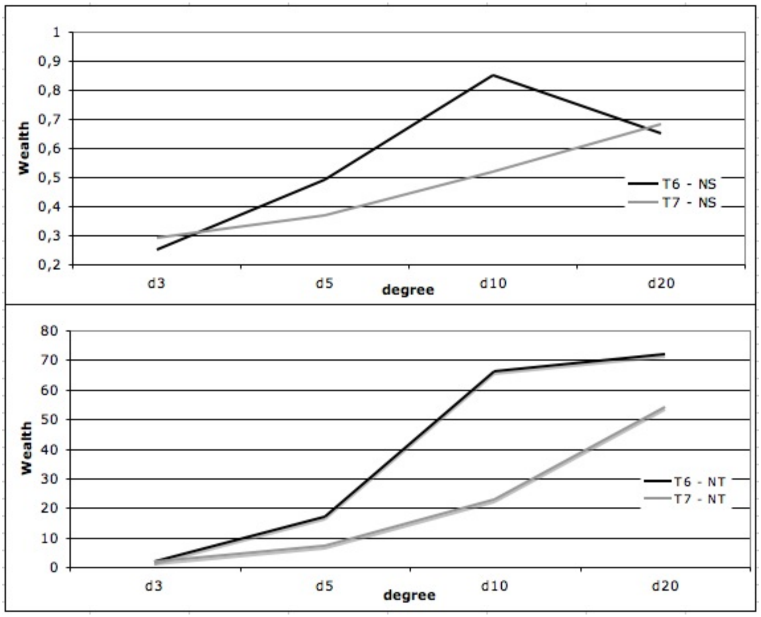


Figure 7.19: Wealth in Env2. Comparison of the average wealth of NS (top) and the average wealth of NT (bottom) with the T6 and T7 mechanisms. Here an "individual reward policy" has been used.

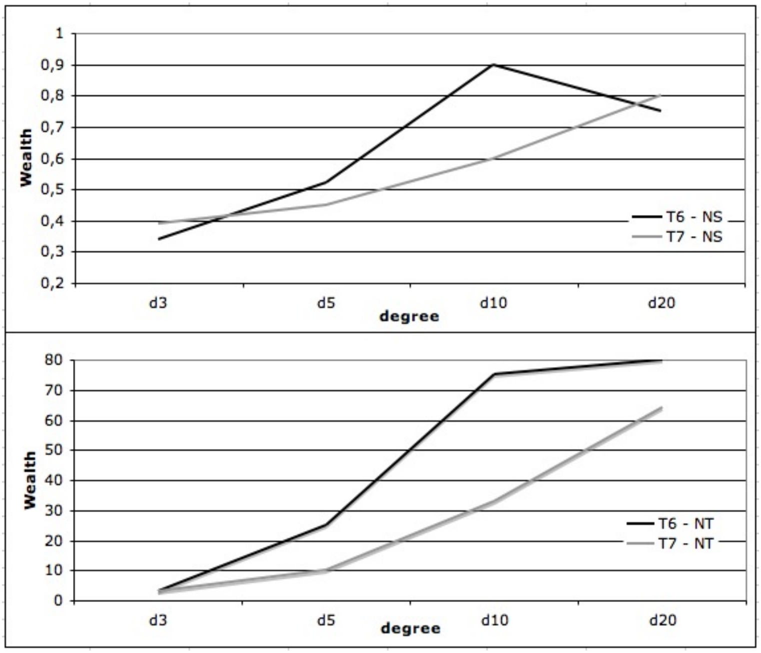


Figure 7.20: Wealth in Env3. Comparison of the average wealth of NS (top) and the average wealth of NT (bottom) with the T6 and T7 mechanisms. Here an "individual reward policy" has been used.

Figures 7.21, 7.22 and 7.23, indicates the same experiments we have just described, but in a situation in which a group reward policy is adopted. Also here the more interesting results are given by the NSs wealth. We can note that in general also here NS using the T6 mechanism increase their average wealth when degree grows from 3 to 10, but it decrease with degree 20. We think that also here applies the same reason we gave for the IR policy. Interestingly, only in Env1 (figure 7.21) when degree is 3 and 20, we have that the average NSs' wealth is greater with T7 than T6. This doesn't happen with Env2 (figure 7.22) and Env3 (figure 7.23): with GR policy, nodes involved in task can move only when entire task is completed. Hence here there is less movement which could help NSs earn more; in particular, when the task environment is dynamic a larger number of nodes can be involved in the completion of tasks and then become idle.

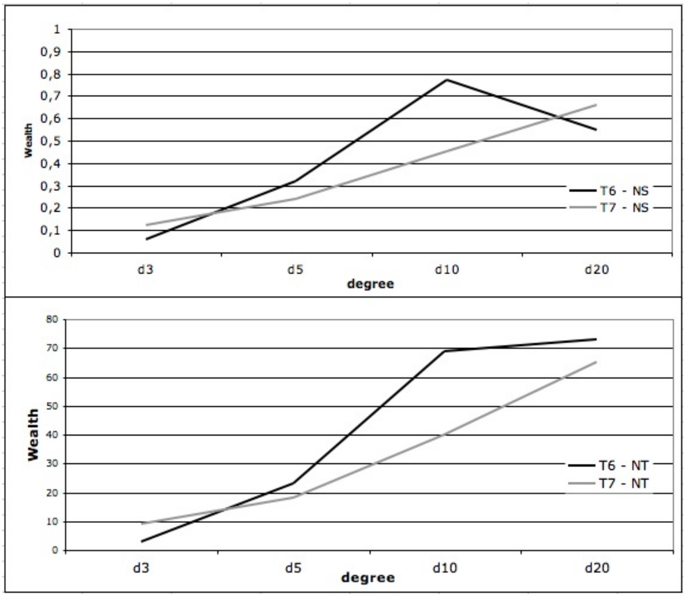


Figure 7.21: Wealth in Env1. Comparison of the average wealth of NS (top) and the average wealth of NT (bottom) with the T6 and T7 mechanisms. Here a “group reward policy” has been used.

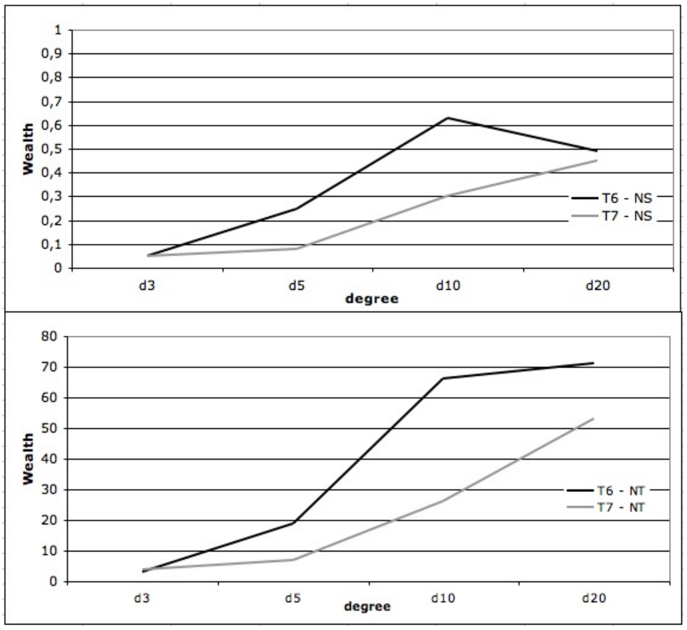


Figure 7.22: Wealth in Env2. Comparison of the average wealth of NS (top) and the average wealth of NT (bottom) with the T6 and T7 mechanisms. Here a “group reward policy” has been used.

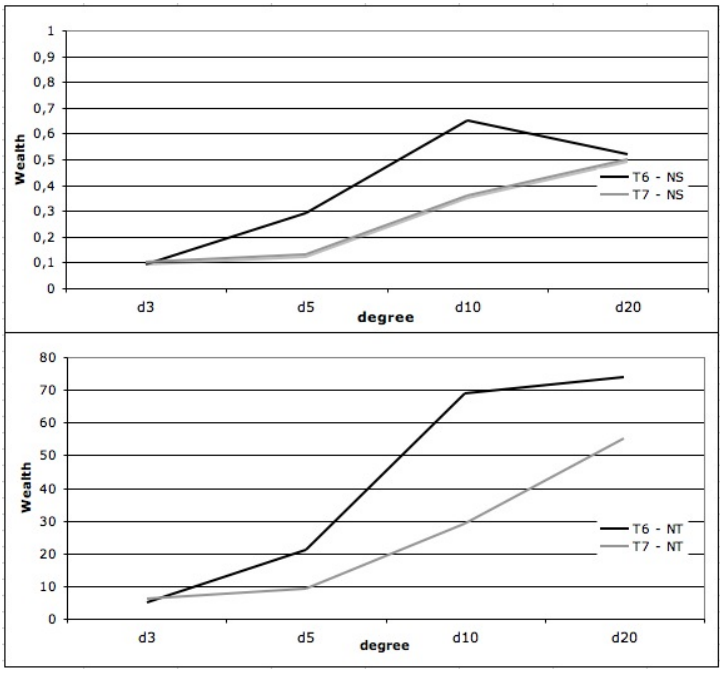


Figure 7.23: Wealth in Env3. Comparison of the average wealth of NS (top) and the average wealth of NT (bottom) with the T6 and T7 mechanisms. Here a "group reward policy" has been used.

7.4 Conclusion

The work presented in this thesis contributes to the area of studies that is concerned with the issue of coordination and skills allocation in knowledge-based production processes. In addition, our paper presents an attempt to bridge studies in computer science, dealing with the nature and the mechanisms of P2P networks evolution, to organization studies. To represent individual decision-making of agents embedded in organizational networks, we adapted an algorithm, the SLAC algorithm (Hales, 2002 [37]; Marcozzi et. al., 2005 [59]; Hales and Arteconi, 2006 [38]), developed to study cooperation in P2P networks. In this respect, we believe that the P2P network is a useful concept to address novel forms of organization in a knowledge economy. We presented a model of an organization as a network of specialists, endowed with idiosyncratic skills, and 'product managers', that maintain a contact with external clients and receive tasks to be completed from these latter. Specialists and product managers are peers, because they cannot adopt any hierarchical mechanisms to regulate their interactions. We explored under what circumstances this organization, without hierarchical authority, successfully self-organizes to complete tasks. In particular, we investigated how global network structures able to complete tasks can emerge from local interaction. Simulation experiments teased out problems in applying self-organizing mechanisms to the government of a firms skill endowment. In our computer simulation experiments, we found that as task environment becomes dynamic, market-based mechanisms substitute for hierarchical control to increasingly gain effectiveness. Yet, we found that effectiveness of market-based mechanisms depend on assumptions concerning individual incentives, on the ability of specialists to learn contiguous skills and on three aspects of organisational morphology: (1) connectivity of individual nodes within the organisation network, (2) reward policies and (3) fluidity of teams boundaries. In particular, we mention six key findings emerging from our study. First, for a sufficient degree of connectivity (by degree we refer to the number of connection maintained by each node), self-organizing mechanisms prevail over hierarchical mechanisms of control.

Second, ability of specialists to extend their know-how to learn contiguous skills increases performances of the network as the repertoire of diverse skills required to complete incoming tasks gets larger. Third, performances of the network greatly improve as we assume that product managers are able to search within the network specialists bearing specific skills. In this case, fluidity of teams boundaries increases performances of the network. Fourth, when nodes have individual incentives that suggest them to connect to other nodes that are simply richer of more connected, performances of the network decrease. In this case, rigidity of teams boundaries improves performances of the network. Fifth, group-based reward mechanisms abate networks performances when content of tasks to be completed changes dynamically but the value attached to each task remains stable. Finally, for low or high degree of nodes connectivity, a problem of diverging incentives emerges between specialists and product managers. Specialists may have an incentive to search connection to product managers that are scarcely connected to find structural holes in the network and exploit bargaining power. When specialists are able to exploit structural holes, they increase their personal wealth at the cost of decreasing the ability of the firm to complete tasks.

Chapter 8

Discussion

In this thesis we proposed a novel approach to study the problem of task allocation in firms working in a knowledge based economy. Our work contributes to the studies in organization theory and the aim was to study some plausible self-organizing mechanism through computer simulation, developing hypothesis which can be tested with future research. We designed different approaches to our problem, approaches which have been experimented with the FirmNet model (Mollona and Marcozzi, 2008 [68, 66, 67]; Mollona and Marcozzi, *submitted* [69]). Given the increasing knowledge intesity of jobs, the role of firms as integrators of know-how and skill, becomes crucial but as economic activity becomes mainly intellectual the traditional hierarchical control may result ineffective (Hodgson, 1999 [44]). Sometimes managers in a firm may not be able to assign the right skill to a particular job and individuals must be able to self-elige for jobs leading to an auto-formation of team. Given this particular nature of firms in a knowledge economy, we decided to draw inspiration from Peer-to-Peer networks to design companies. In particular we looked at gossiping, using it directly to have a peer selection mechanism (Newscast). We considered this as a network of acquaintances which is independent from the network of links related to the teams in the firm. P2P networks are decentralized, have shared ownership reducing costs of owing the system and are self-organized leading to the emergence of global order without the presence of a central authority. In our opinion these characteristics can be found also in knowledge-based firms, that's why we designed our FirmNet model looking at P2P networks.

An important help in understanding the dynamics underlying the emergence of teams, came from the study of social modeling through the means of computational sociology modelling and simulations techniques, especially from the tag systems, the translation of which in an explicit network topology lead initially to the SLAC protocol (Hales, 2002 [37]; Marcozzi et al. 2005, [59]). SLAC is the protocol from which we took inspiration for developing the mechanism we described over the thesis, mechanism which we adopted for the organization of the individuals in the firm. We developed an agent-based model using Peersim, a P2P network simulator [105]. We started with a model describing a basic situation and then we incrementally added components and variables to articulate more the scenario of our investigation.

The first step that we did was to compare two different approaches: a hierarchical one versus a marked-based one (we called this self-organizing, SO). The SO approaches also included some evolutionary routines with which nodes were coping the strategies of those performing better. The idea was to understand the boundaries of the applicability of authority in an environment in which skills are mainly intellectual. We found that hierarchy gives better results in terms of tasks completed than the market-based mechanism but its structure has more costs. Moreover we found that as the environment of the task becomes more dynamic, percentage of task completed decreases showing similar results as the SO approach. Hence our first finding was that a stable environment matches the best with a centralized form of organization while a dynamic environment demands an organization with high individual autonomy.

From this first work we also noted that the structure of the network matters and that very clustered networks perform better. Following this line we focused more on the self-organizing approach, articulating more the mechanisms related to the movement of the nodes in the network for the formation of teams. We tested six different mechanism divided in two categories: “closed team rewiring” and “open team rewiring”. In the first, all the rewiring mechanisms implied that a node who wants to join a team has to leave the old one; the second permitted to a node to stay in different teams. Here we also tested three different task environments: static with static task payoff, dynamic with static task payoff and dynamic with variable task payoff. We found that in general the “open” approach gives better results; this indicates that keeping old good ties creates a good starting point when a new team must be assembled for a new task.

Basing on the fact that in a knowledge economy the nominal supervisor is not always able to connect skills with jobs and that the needed skills are not always available and hence must be built, we introduced another version of our model in which some learning mechanisms are present. We also increased the number of skills available in the system, finding that even in a static environment, when the range of skills increases and hence the organization process becomes more arduous, a hierarchical approach results less effective than a self-organizing one. We gave to the nodes the ability to learn the skill needed by the manager or to pass the job to one of its neighbors. We found that these mechanisms enhance the performances of the firm respect to the cases in which such possibilities were not taken into account. We also tested the same approach introducing some kind of “strategies” evolution, letting nodes copy the attitudes at learning the job and attitude at passing the job of the nodes with higher wealth. We also introduced a novel rewiring mechanism with which employees tend to make new ties with managers linked only to few neighbors. Until this point we said that having a large number of links is a good thing. We found that highly clustered networks gives better results than low clustered ones and that employees may have a preference for connecting to managers with high degree. What happens if employees try to link to managers with only few links? We found that this is too simple and is able to give some advantages only to the single specialist adopting it (in terms of wealth); in terms of completed task the firm performs worse. Finally we compared “individual” and “group” reward policies finding that in general group reward gives better results when the task environment is static.

We are aware that the contribution presented in this work and summed up above, has lot of limitations leaving open questions and suggesting directions the work should be brought on. First of all we did not consider any kind of costs related to the connection between the individuals in the firm. We argued that in general the more clustered the network is and the more links a nodes have, the better results can be obtained. But what happens if maintaining links implies a cost? We assumed that since the skills are intellectual, communications between specialist take place mainly over electronic devices like the internet which implies very low costs. Sometimes those communications may need higher costs like for example traveling. We think that this point must be investigated.

Another point that we did not take into account is the honesty of the individuals: we said that links are made according to some particular rules, like for example making a link to a rich node, or to a node with high degree or to a node with a certain skill. But what happens if agents lie and declare a fake information? How can the system (firm) react to this fault?

In this work we also did not take into account the problem of specialists turnover. The problem of turnover has been widely studied both in organization sciences (Dineen and Noe, 2003 [25]; Marks et al., 2001 [58]; Zoethout et al., 2007 [104]) and in computer sciences (Rhea et al., 2004). Nodes in the network may suddenly leave and be replaced by others bearing different skills. We expect that dynamic task environments should not suffer of this turnover while static task environments could see a massive worsening of performances. We think that to comprehend the process of adaptation of teams and newcomers, computer simulation may be a good tool.

Finally, in this work we have always considered a situation in which project managers were defined *a priori* and never changed over the time, implying a kind of authority which decided who has to be manager. Novel forms of organization (Benkler, 2006 [7]), propose that in some cases also project managers should be defined at run-time, according to some feature of the individual which may be its current capabilities or its current position in the network (its centrality). In this respect we are working on a model bearing inspiration from the literature on SuperPeer networks (Yang and Garcia-Molina, 2003 [102]; Jesi et al., 2007 [49]). In this new model, tasks may be initially assigned at random with uniform probability (as in our model) to a certain number of nodes in the network (these nodes called SuperPeer, SP). Then the more task a SP has completed, the more probability it has to receive a new task in the next round. Anyway the number of SPs receiving a task in each round is fixed. When new tasks are generated, if there are some busy SPs (because the task is not completed yet), they will be assigned to random nodes in the network (non SP) which will then have the probability to become SP (completing tasks). The development of this new model is still at an early stage, but we think that it can well represent new organizational needs related to the Knowledge-Based Economy.

References

[1] Adler, P. S. (2001), “Market, Hierarchy, and Trust: The Knowledge Economy and the Future of Capitalism”; Organization Science, 12(2): 215-234.

[2] Axelrod, R. (1997) “The complexity of cooperation: Agent-based models of competition and collaboration”. Princeton, NJ: Princeton University Press.

[3] Axtell, R. L. (1999), “The Emergence of Firms in a Population of Agents”; Working paper 99-03-019, Santa Fe Institute: Santa Fe, New Mexico.

[4] Batista, D.M., da Fonseca, N. L. S. (2007) “A Brief Survey on Resource Allocation in Service Oriented Grids”, in Proc. of Globecom Workshops, 2007 IEEE.

[5] Babaoglu, O., Geoffrey Canright, Andreas Deutsch, Gianni Di Caro, Frederick Ducatelle, Luca Gambardella, Niloy Ganguly, Mark Jelasity, Roberto Montemanni and Alberto Montresor (2005) “Design Patterns from Biology for Distributed Computing”.Proceedings of the European Conference on Complex Systems 2005, Paris.

[6] Benkler, Y. (2006) “The Wealth of Networks”. Free book available at $h$ttp://www.benkler.org/

[7] Benkler, Y. (2002) “Coases Penguin, or, Linux and The Nature of the Firm”. Available at $h$ttp://www.benkler.org/

[8] Brown, J. S. and P. Duguid (2001), “Knowledge and organization: A social-practice perspective”, Organization Science, 12(2): 198-213.

[9] Bredin, J., Kots, D. and Rus, D. (1998) Market-based Resource Control for Mobile Agents. In *Proceedings of "Autonomous Agents"* Copyright 1998 by Association of Computing Machinery (ACM), pp. 197-204

[10] Bruun, C. (2006) "Advances in Artificial Economics" Springer 2006, ISBN 3540372474, pages 294.

[11] Brusoni, S., Prencipe, A. and K. Pavit (2001), "Knowledge specialization, organizational coupling, and the boundaries of the firm: why do firms know more than they make?"; Administrative Science Quarterly, 46: 597-621.

[12] Burgeelman, R. A. and Mittman, B. S. (1994) "An intraorganizational ecological perspective on managerial risk behaviour, performance, and survival: individual, organizational, and environmental effects". In J. A. C. Baum and J. V. Singh (Eds.), Evolutionary dynamics of organizations. Oxford University Press

[13] Burns, T. and G. M. Stalker (1961), "Management of Innovation", London: Tavistock Publications.

[14] Carley, K. M. (1992), "Organizational Learning and Personnel Turnover", Organization Science, 3(1): 20-46.

[15] Carley, K. M. and Z. Lin (1997), "A Theoretical Study of Organizational Performance under Information Distortion", Management Science, 43(7): 976-997.

[16] Cascio, Wayne F. and Aguinis, Herman(2008) "Staffing Twenty-first-century Organizations". The Academy of Management Annals, 2:1,133 – 165

[17] Chattoe, E. (1998) "Just how (un)realistic are evolutionary algorithms as representations of social processes?". Jounal of artificial social sciences simulation, 1(3): 2.1 – 2.36.

[18] Cohen, A. M. (1962), "Changing Small-Group Communication Networks", Administrative Science Quarterly, 6(4): 443-462.

[19] Cohen. A. M., Bennis, W. G. and G. H. Wolkon (1962), “The Effects of Changes in Communication Networks on the Behaviors of Problem-Solving Groups”, Sociometry, 25(2): 177-196.

[20] Cohen, M. D., March, J. G and J. P. Olsen (1972), “A Garbage Can Model of Organizational Choice”, Administrative Science Quarterly, 17(1): 1-25.

[21] Cohen, A. M., Robinson, E. L. and J. L. Edwards (1969), “Experiments in Organizational Embeddedness” , Administrative Science Quarterly, 14(2): 208-221, Laboratory Studies of Experimental Organizations.

[22] Davis-Blake, A. and B. Uzzi (1993), “Determinants of Employment Externalization: A Study of Temporary Workers and Independent Contractors”; Administrative Science Quarterly, 38(2): 195-223.

[23] Davis J. P., Eisenhardt K. M and Bingham C. B. (2007) “Developing theory through simulation methods”. Accademy of Management Review. Vol. 32 No. 2, pp. 480 – 499.

[24] Dickinson, D. L.and R. M. Isaac (1998), “Absolute and relative rewards for individuals in team production, Managerial and Decision Economics”, 19(4/5), Laboratory Methods in Economics: 299-310.

[25] Dineen, B.R. and Noe, R.A. (2003) “The impact of team fluidity and its implications for human resource management research and parctice”, Research in Personnel and Human Resource Management, 22, 1 – 37.

[26] Druschel. P. and Rowstron, A. (2001). “PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility” . The Legion Vision of a Worldwide Virtual HotOS VIII, Schloss Elmau, Germany, May 2001.

[27] Dubin, R. (1976) “theory building in applied areas”. In M. Dunette (Ed.), Handbook if industrial and organizational psychology: 17 – 40.

[28] Epstein, J. M. and R. Axtell (1996), Growing Artificial Societies: Social Science from the Bottom Up. MIT Press and Brookings Institution Press: Cambridge, Massachusetts, and Brookings Press: Washington, D.C.

[29] Faraj, S. and L. Sproull, (2000) “Coordinating expertise in software developments teams”, Management Science, 46(12): 1554-1568.

[30] Forrester, J. W. (1961). “Industrial dynamics”. Cambridge, MA: The MIT Press.

[31] Foss, N. (2005), “Strategy, Economic Organization, and the Knowledge Economy”. The coordination of firms and resources. Oxford University Press.

[32] Gallegati M. and Richiardi M. (2008), “Agent-based Modelling in Economics and Complexity”, in Meyer B. (ed.), ‘Encyclopedia of Complexity and System Science’, Springer, forthcoming.

[33] Gavetti, G., Levinthal, D., and Rivkin, J. W. (2005). “Strategy making in novel and complex worlds: The power of analogy”. Strategic Management Journal, 26, 691712.

[34] Gilbert, N. and Troitzsch, K. G. (2005). “Simulation for the social scientist” (Second ed.). Milton Keynes: Open University Press.

[35] Glance, N. S., Hogg, T. and B. A. Huberman (1994), “Training and Turnover in the Evolution of Organizations”, Organization Science, 8(1): 84-96.

[36] Gordon, D. M. (2001) “Task Allocation in Ant Colonies”, in Segel, L. A. and Cohen, I. R. (eds.), Design Principles for Immune System and Other Distributed Autonomous System, Oxford Univ. Press.

[37] Hales, D. (2002), “Evolving Specialisation, Altruism and Group-Level Optimisation Using Tags”. In Sichman, J. S., Bousquet, F. Davidsson, P. (Eds) Multi-Agent-Based Simulation II. Lecture Notes in Artificial Intelligence 2581, pp.26-35 Springer.

[38] Hales, D. and S. Arteconi (2006), "SLACER: A self-organizing protocol for coordination in peer-to- peer networks". *IEEE Intelligent Systems*, 21(2):29-35.

[39] Hales, D.; Marcozzi, A.; Cortese, G. (2007) "Towards Cooperative, Self-Organised Replica Management". Short paper. In proc. 1st IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO). Boston, MA (USA).

[40] Hart, S. and M. Kurz (1983), "Endogenous formation of coalitions", Econometrica, 51(4): 1047-1064.

[41] Hayek, F. A. (1945), "The use of knowledge in society", The American Economic Review, 35(4): 519-530.

[42] Haveman, H. A. (1993), "Organizational size and change: diversification in the savings and loan industry after deregulation"; Administrative Science Quarterly, 38: 20-50.

[43] Hoegl, M. and H. G. Gemuenden (2001), "Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence", Organization Science, 12(4): 435-449.

[44] Hodgson, G. M. (1999) "Economics & Utopia"; London, Routledge.

[45] Hunt, R. G. (1976) "On the work itself: Observations Concerning Relations between Tasks and Organization Processes". In E.J. Miller (ed.) Task and Organization, Tavistock Institute of Human Relations, London.

[46] Jelasity, M. and M. van Steen (2002), "Large-Scale Newscast Computing on the Internet". *Internal Report IR-503, Vrije Universiteit Amsterdam.*

[47] Jelasity, M and Babaoglu, O. (2005) "T-Man: Gossip-based Overlay Topology Management" In Proceedings of Engineering Self-Organising Applications (ESOA05).

[48] Jelasity, M., Montresor, A. and Babaoglu, O. (2005) "Gossip-based aggregation in large dynamic networks". ACM Transaction Computer systems. Vol. 23:219–252. ACM Press.

[49] Jesi, G. P., Montresor, A. and Babaoglu, O. (2007) "Proximity-aware Superpeer Overlay Topologies". IEEE Transactions on Network and Service Management (TNSM). Number 2, Vol. 4, pp. 74 – 83.

[50] Kim, W. C. and R. Mauborgne (1998), "Procedural justice, strategic decision making, and the knowledge economy", Strategic Management Journal, 19(4): 323-338.

[51] Kirkman, B. L. and B: Rosen (1999), "Beyond self-management: Antecedents and consequences of team empowerment". The Academy of Management Journal, 42(1): 58-74.

[52] Ibarra, H. (1992), "Structural alignments, individual strategies, and managerial action: Elements toward a network theory of getting things done, in Network and Organizations. Structures, Form, and Action", Nohria, N. and R. G. Eccles (eds.), Harvard Business School Press, Boston, MA.

[53] Langley, A. (1999). "Strategies for theorizing from process data". Academy of Management Review, 24(4).

[54] Leavitt, H. J. (1951), "Some Effects of Certain Communication Patterns on Group Performance", Journal of Abnormal and Social Psychology, 46: 38-50.

[55] Lin. Z. and K. M. Carley (1997), "Organizational Response: The Cost Performance Tradeoff", Management Science, 43(2): 217-234.

[56] Malone, T. W. (1987), Modeling Coordination in Organizations and Markets, Management Science, 33(10): 1317-1332.

[57] March, J. G. (1991) "Exploration and Exploitation in Organizational Learning", Organization Science, vol 2: pp. 71–87.

[58] Marks, M.A, Mathieu, J.E. and Zaccaro, S.J. (2001), “A temporarly based framework and taxonomy of team precesses”, Academy of Management Review, 26 (3), 356 – 376.

[59] Marcozzi, A.; Hales, D.; Jesi, G.; Arteconi, S.; Babaoglu, O. (2005) “Tag-Based Cooperation in Peer-to-Peer Networks with Newscast”. Full paper. In Self-Organization and Autonomic Informatics (I), Volume 135 Frontiers in Artificial Intelligence and Applications, Edited by: H. Czap, R. Unland, C. Branki and H. Tianfield, IOS Press, Netherlands.

[60] Marcozzi, A.; Hales, D. (2008) “Emergent Social Rationality in a Peer-to-Peer System”. Advances in Complex Systems (ACS), Volume No. 11, Issue No. 04. September 2008. World Scientific Press

[61] Marschak and Radner (1972), “Economic Theory of Teams”. New Haven, London, U.K.

[62] M. Masuch and P. LaPotin (1989), Beyond Garbage Cans: An Al Model of Organizational Choice, Administrative Science Quarterly, 34(1): 38-67.

[63] Matusik, S. F. and C. W. L. Hill (1998), “The utilization of contingent work, knowledge creation and competitive advantage”, Academy of Management Review, 23, 4: 680-697.

[64] McGrath, J. E., Arrow, H., and Berdahl, J.L. (2000) “The study of groups: Past, Present and Future”, Personality and social psychology revie, 4, 95 – 105.

[65] Moch, M. K. and E. V. Morse (1977), “Size, centralization and organizational adoption of innovation”; American Sociological Review, 42: 716-725.

[66] Mollona, E. and Marcozzi, A. (2008) “Prestige and Search for Glory. Individual Incentives and Self Emerging Coordination Mechanisms in Knowledge Integration Processes”. In proc. of EURAM 2008. Ljubljana, Slovenia.

[67] Mollona, E and Marcozzi, A. (2008) “Self-Emerging Coordination Mechanisms for Knowledge Integration Processes”. In proc. of ESSA 2008. Brescia, Italy

[68] Mollona, E. and Marcozzi. A. (2008) “FirmNet: The Scope of Firms and the Allocation of Task in a Knowledge-Based Economy”. Journal of Computational and Mathematical Organization Theory (CMOT), Springer. December 2008.

[69] Mollona, E. and Marcozzi. A. (submitted) “Skill Integration in Dynamic Task Environments: Decentralised Decision-Making, Organizational Morphology and Self-Organising Partitioning of Specialised Skills”. Submitted to the EURAM 2009 Conference.

[70] Mollona E. (2008) “Computer simulation in social sciences”. Journal of Manage Governance; Vol. 12, No 2.

[71] Mueller, R.A., Duda, M. R., O’Haire, S. M. (1984) “A survey of resource allocation methods in optimizing microcode compilers”, ACM SIGMICRO, Vol. 15, Issue 4, pp. 285 – 295.

[72] Natter, M., A. Mild, M. Feurstein, G. Drffner and A. Taudes (2001), “The effect of incentive schemes and organizational arrangements on the new product development process”, Management Science, 47(8): 1029-1045.

[73] Ouchi, W. G. (1979) “A Conceptual Framework for the Design of Organizational Control Mechanisms”, Management Science, 25(9): 833-848.

[74] Priem, R. L. and Butler, J. E. (2001) “Is the resource-based ‘view’ a useful perspective for strategic management research?. Academy of Management Reviw, 26: 22 –41.

[75] Repenning, N. (2002) “A simulation-based approach to understanding the dynamics of innovation implementation”. Organization Science, 13: 109 – 127.

[76] Rapoport, A and J. P. Kahan (1984), “Coalition formation in a five-person market game”, Management Science, 30(3): 326- 343.

[77] W. Rice, and Mahon B.(2000). “Peer Networking”. Deutsche Banc Alex. Brown White Paper.

[78] M. Ripeanu; I. Foster and A. Iamnitchi (2002) “Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design”. IEEE Internet Computing, 6(1), February 2002.

[79] Rudolph, J., and Repenning, N. (2002) “Disaster dynamics: Understanding the role of quantity in organizational collapse”. Administrative Science Quarterly, 47(1), 1 – 30.

[80] Podolny, J.M. and Page, K.L. (1998) “Netork forms of Organization”. Annual review of sociology, Vol. 24: 57 –76.

[81] Powell, W. W. (1990). “Neither market nor hierarchy: Network forms of organization”. In Organizational Behavior, 12, 295-336

[82] Rowstron, A.I.T. and Druschel, P. (2001) “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems”. In *Midd leware 01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg,* pages 329-350, London, UK, 2001. Springer-Verlag.

[83] Sah, R. K. and J. E. Stiglitz (1986), The Architecture of Economic Systems: Hierarchies and Polyarchies, The American Economic Review, 76(4): 716-727.

[84] Saxberg, B. O. and J. W. Slocum (1968), “The Management of scientific manpower”, Management Science, 14(8): B473-B489.

[85] Schelling, T.C. (1978). “Micromotives and Macrobehaviour”. W.W.Norton and Company

[86] Seidmann, D. J. and E. Winter (1998), “A theory of gradual coalition formation”, The Review of Economic Studies, 65(4): 793-815.

[87] Simon, H.A. (1996). “The patterned matter that is mind”. In D.M. Steier and T.M. Mitchell (Eds.), Mind matters: A tribute to Allen Newell (Chapter 11). Mahwah, NJ: Erlbaum

[88] Steiner, I.D. (1972) “Group process and productivity”. New York and London: Ac. Press Inc.

[89] Stoica, I., Morris, R., Karger, D., Kaashoek, F. and Balakrishnan, H. (2001). “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications”. Proceedings of the SIGCOMM, pp 149-160.

[90] Strom, D. (2001). “Businesses Embrace Instant Messaging”. January 2001 *e*nterprise.cnet.com/enterprise/09534-7-440339534-7-4403317.html.9534-7-4403317.html.

[91] Tapscott, D. and Williams, A. D. (2007) “Wikinomics (La collaborazione di massa che sta cambiando il mondo)”. Rizzoli Editore.

[92] Von Krogh, G. (2003) Open-Source Software Development. In *MIT Sloan Management Review*, vol. 44, no. 3

[93] Wageman, R. (2001), “How leaders foster self-managing team effectiveness: Design choices versus hands-on coaching”, Organization Science, 12(5): 559-577.

[94] Weick, K. E.(1979) “The social psicology of Organizing”. 2nd ed., New York: McGraw-Hill.

[95] Weick, K. E. (1989) “Theory construction as disciplined imagination”, Academy of Management Review, 14: 516 – 531.

[96] Wilke, H. A. M. and Meertens, R. W. (1994) “Group Performance”, London (etc.): Routledge.

[97] williams, K. Y., and O'Reilly, C.A.III. (1998) "Demography and diversity in organizations: A review of 40 years of research", Research in Organizational Behavior, 20, 77 -. 140.

[98] Williamson, Oliver E. (1985) "The Economic Institution of Capitalism: Firms, Markets, Relational Contracting". The free press, New York.

[99] Williamson, Oliver E. (1991) "The Logic of Economic Organization", 1991, in Williamson and Winter, editors, Nature of the Firm.

[100] Wood, R. E. (1986) "Task complexity: definition of the construct", Organizational Behavior and Human Decision Processes, 37, 60 – 82.

[101] Wright, P. W, and S. A. Snell (1998), "Toward a unifying framework for exploring fit and flexibility in strategic human resource management", The Academy of Management Review, 23(4): 756-772.

[102] Yang, B. and Garcia-Molina, H. (2003) "Designing a Super-peer Network". IEEE International Conference on Data Engineering (ICDE'03).

[103] Zoethout, K. (2006) "Self-Organising Processes of Task Allocation", PhD Thesis, Rijksuniversiteit Groningen.

[104] Zoethout, K., Jager, W, and Molleman, E., (2007), "Newcomers in Self-Organising Task Groups: a Pilot Study", Advancing Social Simulation: The First World Congress, Tokyo, Springer-Verlag.

[105] *http://peersim.sourceforge.net.*

[106] SETI@Home – *h*ttp://setiathome.berkeley.edu/ – Last visited November 2008.

[107] FightAIDS@Home – *h*ttp://www.worldcommunitygrid.org/ – Last visited November 2008.

[108] AfricanClimate@Home – *h*ttp://www.worldcommunitygrid.org/ – Last visited November 2008.

[109] Linux – $h$ttp://www.linux.org – Last visited December 2008.

[110] Wikipedia – $h$ttp://www.wikipedia.org – Last visited December 2008.

[111] Yahoo! – $h$ttp://www.yahoo.com – Last visited November 2008.

[112] Jabber – $h$ttp://www.jabber.org – Last visited November 2008.

[113] Skype – $h$ttp://www.skype.com – Last visited November 2008.

[114] Napster – $h$ttp://www.napster.com – Last visited November 2008.

[115] Kazaa – $h$ttp://www.kazaa.com – Last visited November 2008.

[116] Kazaa – $h$ttp://www.planet-lab.org – Last visited November 2008.

Appendix A

The Peersim System

Peersim is a P2P networks simulator developed at the Department of Computer Science at the University of Bologna. It is written in Java and it is highly modular. We implemented our FirmNet model in Peersim. This appendix gives an overview of it's features.

A.1 Introduction

Evaluating the performance of P2P protocols is a complex task. One of the main problems for their evaluation, is the extremely large scale that they may reach. P2P networks involving hundred of thousands of peers (or more) are not uncommon (e.g., about 5 millions machines are reported to be connected to the Kazaa/Fasttrack [115] network). In addition P2P systems are highly dynamic environments; they are in a continuous state of flux, with new nodes joining and leaving (or crashing).

These properties are very challenging to deal with. Evaluating a new protocol in a real environment, especially in its early stages of development, is not feasible. Distributed planetary-scale open platforms (e.g., Planet-Lab [116]) to develop and deploy network services are available, but these solutions do not include more than about 500 (at the time of writing) nodes. Thus, for large-scale systems, a scalable simulation test bed is mandatory.

The Peersim P2P simulator [105] has been developed with the aim to deal with the previously stated issues. Its first goals are: extreme scalability and support for dynamism. It is a GPL open-source Java based software project. In the following, we provide a brief description of its characteristics.

A.1.1 Peersim Design Goals

The Peersim simulator is inspired by mainly two objectives:

- High scalability: P2P networks may be composed by millions of nodes. This result can be achieved only with a careful design of the data structures involved, trying to avoid (when possible) any overhead. But the memory footprint is not the only problem: the simulator engine must be also efficient.
- Support for dynamism: the simulator must manage nodes joining and leaving the network at any time; this feature has tightly relations with the engine memory management sub-system.

Another important requirement is the *modular* or *component* inspired architecture. Every entity in the simulation (such as protocols and the environment related objects) must be easily replaceable with similar type entities.

The Peersim extreme performances can be reached only accepting some relaxing assumptions about the simulation details. For example, the overhead introduced by the low level communication protocol stack (e.g., TCP or UDP) in not taken into account because of the huge additional memory and CPU time requirements needed to accomplish this task.

A.1.2 Peersim Architecture

As previously stated, Peersim is inspired by a modular and very configurable paradigm, trying to limit any unnecessary overhead. The simulator main component is the *Configurator* entity targeted to read configuration files. A configuration file is a plain ASCII text file, basically composed by key-value pairs. The Configurator is the only not interchangeable simulation component. All the other entities can be easily customized.

In a Peersim simulation, the following three distinct kind of elements can be present: protocols, dynamics and observers. Each of them is implemented by a Java class specified in the configuration file. The network in the simulation is represented by a collection of nodes and each node can hold one or more protocols. The communication between node protocols is based on method calls. To provide a specific kind of service, each component must implement a specific *interface*. For example a protocol has to implement at least the `Protocol` or `CDProtocol` interface to run on Peersim.

Peersim has an utility class package to perform statistic computations or to provide some starting topology configuration based on well know models (such as: random-graph, lattice, BA-Graph,. . .).

The *Simulator* engine is the component that performs the computation; it has to run the component execution according to the configuration file instructions. At the time of writing, Peersim can perform simulation according to the following execution models:

- Cycle based: at each step, all nodes are selected in a random fashion and each node protocol is invoked in turn;
- Event based: a support for concurrency is provided. A set of events (messages) are scheduled in time and node protocols are run according to the time message delivery order.

This thesis work is based on the first simulation model.

A.1.3 FirmNet Implementation Issue

An implementation issue of our model consists in the interleaving of the application level related operations and the rewiring periodically performed in FirmNet. Since the application is performed continuously while the Network Evolution phase (rewiring) takes place periodically, application level operations (Interaction phase) are defined on a per-cycle basis, while rewiring operations will be performed at every simulation cycle. The easiest way to achieve this is to force rewiring not to be executed at any cycle, but only at regular intervals and to be idle the rest of the time. Defining a fixed period of execution though implicitly defines a strong synchronism among nodes, with all of them executing the rewiring at regular intervals and at the same time. To get around this we defined a probability of execution rather than a period of execution in rewiring so that at each cycle the rewiring phase can be executed with some given probability. This simple mechanism destroys the synchronicity between nodes maintaining the average period of execution of rewiring. An N cycles synchronous period can be redefined in an asynchronous way just by defining a probability of execution per cycle equal to $1/N$, so to obtain the result of having each node executing, on average, a reproduction step every N cycles. The parameter describing the probability for each node to rewire at each cycle is called *rewiring probability* .

Through these different synchronization models we are modelling the notion that informations updates for some application tasks might be instantaneous and asynchronous, for example, neighbor nodes might provide requested resources immediately in some application tasks.

Appendix B

The Newscast Protocol

B.1 Introduction

Newscast [46] is a gossip-based topology manager protocol. Its aim is to continoulsy rewire the (logical) connections between hosts. The rewiring process is designed in such a way that the resulting overlay is very close to a random graph. The generated topology is thus very stable and provides robust connectivity. This protocol has been used successfully to implement several P2P protocols, including broadcast [46] and aggregation [48].

As in any large P2P system, a node only knows about a small fixed set of other nodes (due to scalability issues), called *neighbours*. In Newscast, the neighbourhood is represented by a partial, fixed c size view of node *descriptors* composed by a node address and a logical *time-stamp* (e.g., the descriptor creation time).

Referring to the usual gossip scheme (see Figure B.1), the protocol behaviour

```
while(TRUE) do
 wait(Δt);
 neighbour = SELECTPEER();
 SENDSTATE(neighbour);
 n_state = RECEIVESTATE();
 my_state.UPDATE(n_state);
```

(a) Active Thread

```
while(TRUE) do
 n_state = RECEIVESTATE();
 SENDSTATE(n_state.sender);
 my_state.UPDATE(n_state);
```

(b) Passive Thread

Figure B.1: The gossip paradigm.

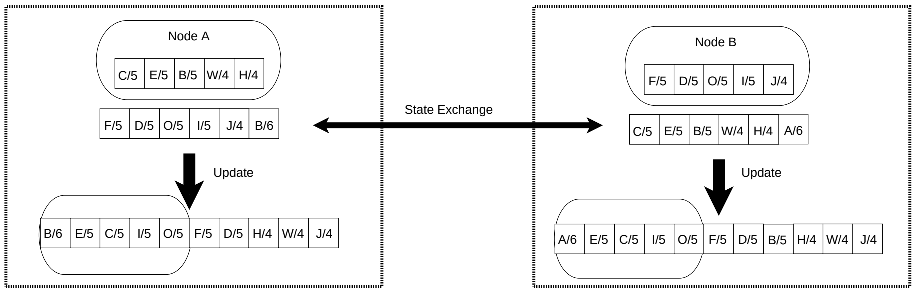


Figure B.2: A Newscast exchange between node A (active) and B. Each node has its own 5 descriptor elements view depicted inside the ellipses. A descriptor is a *node-ID*, *timestamp* pair. After the state exchange node A has received the node B view and viceversa; then each partecipant merges the received view with its own. The result is depicted under the empty arrow: each node has selected the "freshest" descriptors at random and has discarded the others (those inside the ellipse) to obtain new 5 element view. Note that in this basic example, each node sends its entire view; however, the view can be purged by "old" descriptors before sending.

performs the following actions: selects first a neighbour from the local view, exchanges the view with the neighbour, then both participants update their actual view according to the received view. The data actually sent over the network by any Newscast node is represented by the node's own descriptor plus its local view.

In Newscast, the neighbour selection process is performed in a random fashion by the SELECTPEER() method. The UPDATE() method is the Newscast core behaviour. It merges ($\bigcup$ operation) a received view (sent by a node using SEND-STATE()) with the current peer view in a temporary view list. Finally, Newscast trims this list to obtain the new c size view. The node descriptors discarded are chosen from the most "old" ones, according to the descriptor time-stamp. This approach changes continuously the node descriptors hold in each node view; this implies a continuous rewiring of the graph defined by the set of all node views. This behaviour is shown in Figure B.2.

Even though the system is not synchronous, we find it convenient to describe the gossip-scheme execution as a sequence of consecutive real time intervals of length Δ (see the "wait" statement in pseudo-code in Figure B.1), called *cycles* or *rounds*.

The protocol always tends to inject new informations in the system and allows an automatic elimination of old node descriptors using the aging approach. This feature is particularly desirable to remove crashed node descriptors and thus to repair the overlay with minor efforts. In addition, the protocol does not require any clock synchronization, but only that the timestamp of node descriptors in each view are mutually consistent.

The topology generated by Newscast has a low diameter and it is close to a random graph having out-degree c. Experimental results proved that a small 20 elements partial view is already sufficient for a very stable and robust connectivity, regardless of the network size.

Newscast is also cheap in terms of network communication. The traffic generated by the protocol involves the exchange a few hundred bytes per cycle for each peer and is estimated in [46].

A protocol such as Newscast provides a service to pick random nodes from the whole network and we can call it *Randomizer Service* (or peer sampling service). The chance to extract a fresh new node, selected at random from the whole network, is a high desiderable source of information for P2P protocols. We can consider such a service as a *building block* for many P2P protocols. In this vision, the cost effectiveness of Newscast is very useful, because the *Randomizer Service* has to be always-on and run by all peers involved in the overlay.

Such a randomizer service can also be a key component during the initialization phase (*bootstrap*) for any higher level protocol in order to fill its view at the beginning. We use Newscast both as a randomizer service (peer sampling service) and as a bootstrap facility in the Peersim implementation of the FirmNet model. Indeed, at cycle 0 of our simulations, the FirmNet network is identical to the Newscast network; then the former evolves according to the rewiring mechanisms described over the whole thesis and the latter evolves according to the Newscast protocol described in this chapter.