Complex systems and health systems, computational challenges

Zifan Liu

THÈSE

Pour obtenir le titre de

Docteur en Informatique

présenté par

Zifan LIU

à l'Université de Versailles Saint-Quentin-en-Yvelines

Complex Systems and Health Systems, Computational Challenges

présentée et soutenue publiquement le 11 février 2015

Commission d'examen

Rapporteurs :	Joel SALTZ	Stony Brook University, USA
	Michel DAYDÉ	Université de Toulouse, France
Examinateurs :	Didier GUILLEMOT	Institut Pasteur, France
	Alain BUI	Université de Versailles, France
	Christophe CALVIN	CEA, France
	Michel KERN	INRIA, France
Directrice de thèse :	Nahid EMAD	Université de Versailles, France
Co-directeur de thèse :	Michel LAMURE	Université de Lyon 1, France
	Soufian BEN AMOR	Université de Versailles, France

Acknowledgments

Without the influence and support of all people around me, I would not be able to achieve the doctorate degree in compute science.

I would like to first thank the members of my dissertation committee - not only for their time and patience, but for their insightful comments and questions during my defense. I am indebted to Prof. Michel Daydé and Prof. Joel Saltz, who spent their time to review my thesis to help me improve and finalize it.

I would like to thank all my teachers and professors, whose truly altruistic effort to promote a solid understanding of numerical linear algebra and pretopology. My special gratitude goes to my advisor Prof. Nahid Emad, for the support, guidance and help all along my three years' Ph.D, to Prof. Sofian Ben Amor and Prof. Michel Lamure for their helpful and insightful discussion on computational epidemiology and pretopology.

Intellectually stimulating discussions and long hours of party with Florian, Makarem, Feng, Langshi, Cihui, Fan and France made my life in Paris extremely enjoyable. Also, I am thankful to my friends from my church, including Ming, Zhiren and many, for a great time sharing and praying together.

Special thanks goes to my wife Xue, for taking a moment of her time to proofread and edit my dissertation. I am truly thankful to my parents (Yi and Yaxian), who managed to make my childhood as carefree as it was possible. Words cannot express my infinite gratitude to my wife Xue for being a great support and for sharing the moments of joy and happiness during all time we are together. Thank you !

Table des matières

Table des figures

Liste des tableaux

Résumé

Le calcul des valeurs propres intervient dans des modèles de maladies d'épidémiques et pourrait être utilisé comme un allié des campagnes de vaccination dans les actions menées par les organisations de soins de santé. La modélisation épidémique peut être considérée, par analogie, comme celle des virus d'ordinateur qui dépendent de l'état de graphe sous-jacent à un moment donné. Nous utilisons PageRank comme méthode pour étudier la propagation de l'épidémie et d'envisager son calcul dans le cadre de phénomène petit-monde.

Une méthode multiple de "implicitly restarted Arnoldi iterations" a été proposé par Shahzadeh Fazeli et al. (MIRAM). Dans cette thèse, Une mise en œuvre parallèle de cette méthode est proposé pour calculer le vecteur propre dominant de matrices stochastiques issus de très grands réseaux réels. La grande valeur de "damping factor" pour ce problème fait de nombreux algorithmes existants moins efficace, tandis que MIRAM pourrait être prometteuse. Nous proposons également dans cette thèse un générateur de graphe parallèle qui peut être utilisé pour générer des réseaux synthétisés distribués qui présentent des structures "scale-free" et petit-monde. Ce générateur pourrait servir de donnée pour d'autres algorithmes de graphes également.

MIRAM est mis en œuvre dans le cadre de trilinos, en ciblant les grandes données et matrices creuses représentant des réseaux sans échelle, aussi connu comme les réseaux de loi de puissance. Hypergraphe approche de partitionnement est utilisé pour minimiser le temps de communication. L'algorithme est testé sur une grille nationale Grid5000. Les expériences sur les très grands réseaux tels que Twitter et Yahoo avec plus de 1 milliard de nœuds sont exécutées. Avec notre mise en œuvre parallèle, une accélération de 27$\times$ est réalisé par rapport au solveur séquentiel.

Mots clés : épidémie, PageRank, Théorie de graphe, Loi de puissance, IRAM, Grande donnée, Hypergraphe partitionnement.

Abstract

The eigenvalue equation intervenes in models of infectious disease propagation and could be used as an ally of vaccination campaigns in the actions carried out by health care organizations. The epidemiological modeling techniques can be considered by analogy, as computer viral propagation which depends on the underlying graph status at a given time. We point out PageRank as method to study the epidemic spread and consider its calculation in the context of small-world phenomenon.

A multiple method of implicitly restarted Arnoldi iterations has been proposed by Shahzadeh Fazeli et al. (MIRAM). In this thesis, a parallel implementation of this method is proposed for calculating dominant eigenpair of stochastic matrices derived from very large real networks. Their high damping factor makes many existing algorithms less efficient, while MIRAM could be promising. We also propose in this thesis a parallel graph generator that can be used to generate distributed synthesized networks that display scale-free and small-world structures. This generator could serve as a testbed for graph related algorithms.

MIRAM is implemented within the framework of Trilinos, targeting big data and sparse matrices representing scale-free networks, also known as power law networks. Hypergraph partitioning approach is employed to minimize the communication overhead. The algorithm is tested on a nation wide cluster of clusters Grid5000. Experiments on very large networks such as twitter and yahoo with over 1 billion nodes are conducted. With our parallel implementation, a speedup of 27$\times$ is met compared to the sequential solver.

Keywords: Epidemic, PageRank, Scale free networks, Power law, IRAM, Big data, Hypergraph partitioning.

Chapitre 1

Introduction

1.1 Motivation

Dynamic complex systems appear in many areas such as physics, biology, and computer networks etc. In the domain of health research, quick response and effective control of widely spreading health crises stay a big challenge for public health officials as well as scientists. The 1918 flu pandemic in the U.S. has caused more lives than those due to World War I. According to the World Health Organisation (WHO), infectious diseases account for more than 13 million deaths a year. Spurred by the rapid development in big data techniques, it has now become possible to deal with this problem more efficiently. While different models have been proposed to simulate epidemic spread, efficient computational methods to handle large-scale epidemic outbreak have not received adequate attention.

In order to simulate the epidemic spread, such as A/H1N1 outbreak in France, computational epidemiology plays an important role. It is an interdisciplinary area which is based on developing large scalable models over computers. For example, Network Dynamics and Simulation Science Laboratory (NDSSL) has proposed a parallel simulation model Simdemics [15][16], designed to scale to the entire United States (300 Million people). This solver generates roughly 300 GB of data and is expected to increase as more details are added. One run takes 3000 cpu hours on a 1.5TF machine of 448 cores and one experiment takes 100 to 300 runs. As a result, one to four experiments per year could be expected. In fact, the traditional method uses

multi-agent system (MAS) and requires hundreds of experiments and computes the expected outcome by averaging. The simulations based on MAS model become increasingly heavy especially when facing with a large population because more details about the population should be used. MAS simulation is communication-bound so that it is not very well adapted for modern parallel machines. Can we find a model that avoids these repeated runs as in MAS approach ? And can we transform its simulation into a computationally intensive problem ? How to solve it on modern supercomputers ? How to establish efficient interventions based on this model ? In this study, we address these problems by proposing a computational model using sparse matrix operations. Based on this model, the intervention as vaccination can be further formulated as a very large sparse eigenvalue problem. From a computational perspective, solving the underlying numerical problems is difficult due to the size and the structure of social networks in question.

On the other hand, most of the existing work in this area now has focused on considering the effect of underlying social network structure on epidemic dynamics by using tools from statistical physics [11]. The new trend of complex network-based models recognize the individual-level randomness and network topology as significant factors on the dynamic of epidemics, which introduces stochastic aspects in the modeling [19][78]. Adaptive network models can be seen as an offspring of this tendency, the idea of which is that individuals usually attempt to reduce their risk of infection by eliminating contact with contagious individuals [18][48][53]. The objective of this approach is to model infection spreading in a population with evolving contacts. We propose in this study a mathematical model of dynamic-network epidemic spread based on stochastic pretopology, which is an extension of the theory of random graphs.

1.2 Related Work

1.2.1 Modelling of epidemic spread

An epidemic is said to arise when new cases of a certain disease occur in excess of normal expectancy, the result of which is a very high mortality in a population during a short period of time. There are several severe epidemic outbreaks in the human history, where we could cite :

• The plague of Justinian during the age of the Eastern Roman Empire, which killed as many as 25 million people across the world.

• The black death in Europe in the years 1382-50 CE, which resulted in the deaths of an estimated 75 to 200 million people.

And more recently,

• The 1918 flu pandemic, occurred during January 1918 - December 1920, killed 50 to 100 million people in the world.

• The HIV/AIDS pandemic, from the year of 1981, has caused over 30 million deaths until now in global.

In the first place, it is necessary to distinguish the following three terms : endemic, epidemic and pandemic. The major defference between these terms is the the proportion of infected people in the population.

• Endemic : in epidemiology, an infection is said to be endemic in population when that infection enters a steady state without the need of external inputs. Central America is an endemic zone of dengue fever. Chickenpox is endemic in the UK. The Thailand is an endemic zone for yellow fever. In general, within a geographical zone, an endemic develops according to geographical and climatical conditions, social-economical factors as well as alimentation of individuals.

• Epidemic : an epidemic occurs when new cases of a certain disease, in a given population, and during a given period, substantially exceed what is expected based on recent experiences. Many examples could be cited : the epidemic of spain flu, the epidemic of SIDA, of obesity and of lung cancer.

• Pandemic : a pandemic is an epidemic of infectious disease that has spread through human populations across a large region ; for instance multiple continents, or even worldwide. We have as example the case of A/H1N1 influenza, which caused a lot of deaths worldwide. As shown in Figure 1.1, zones in yellow show the cases of infection. In red, the indication of the number of observed cases, with 5 thresholds, 1, 5, 10 50 100.

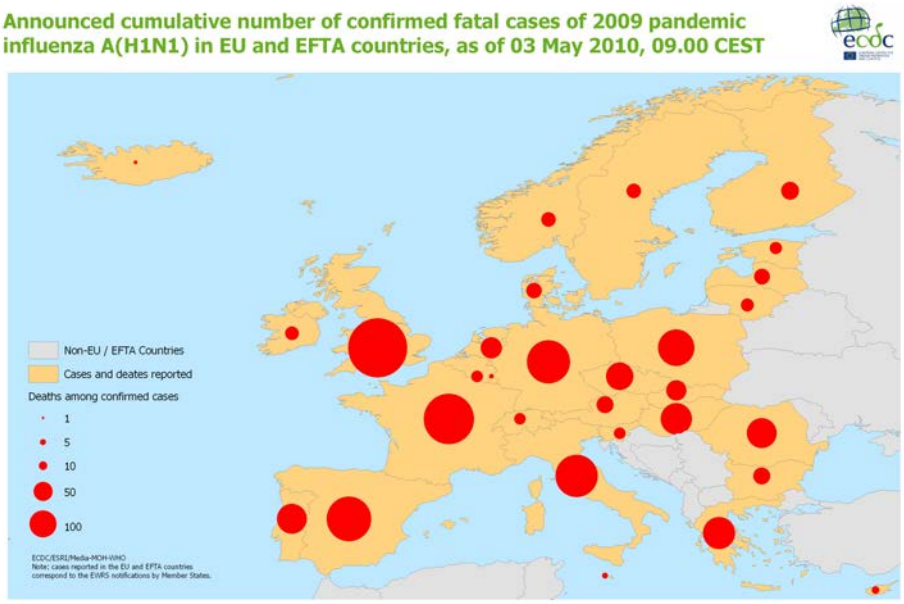


FIGURE 1.1: Illustration of A/H1N1 influenza en Europe, 12 may 2009, ECDC

The diffusion or the propagation of epidemic is defined as the evolution of infection with respect to temporal and spatial aspects. In [22], a first study on pertussis in developing countries is carried out at spatial scale. In this work, Broutin et al. determine the impact of local heterogeneity of the spread of the epidemic of pertussis and its persistence in a spatio-temporal

environment. Then they highlight the impact of the population density on the dissemination and persistence of pertussis by performing a time series analysis.

There are two basic deterministic processes being used in the analysis of epidemic spreading, the susceptible-infective-recovered (SIR) process and the susceptible-infective-susceptible (SIS) process, where the difference is whether or not the disease confers lifelong immunity.

1.2.2 Review of epidemiological models

One common used measure of infectivity is the epidemic threshold λ_c, which is the minimum infectiousness that a disease has to reach in order to invade a network. Each susceptible node is infected with rate ν if it is connected to one or more infected nodes. At the same time, infected nodes are cured with rate δ, defining an effective spreading rate $\lambda = \nu/\delta$. If the value of λ is above the threshold, $\lambda \geq \lambda_c$, the infection spreads and becomes persistent. Below it, $\lambda < \lambda_c$, the infection dies out exponentially fast [89][3].

The homogeneous models (SIR or SIS) assume that the population mixed at random, so that each individual has an equal chance of coming into contact with any other individual [37]. However in real world, it is not rare to find the different mixing rates between the population subgroups [10]. So a direct improvement is to avoid the random-mixing assumption. Models that include underlying network structure achieve this goal by assigning to each individual a finite set of contacts [79][76].

Network-based epidemic models

The difference between various network models depends on how individuals are distributed in space and how connections are formed. The Paul Erdös and Alfréd Rényi model (E-R model) consists of n nodes, joined by edges which are placed between pairs of nodes chosen with equal probability p. By using ideas drawn from percolation theory [77], it is found that there is a non-zero epidemic threshold,

$$\lambda_c = \frac{1}{< k >} \tag{1.1}$$

where $< k >$ is the average connectivity. The degree distribution of this model is Poisson and the epidemic growth rate in such a network is reduced in comparison with the random mixing model [57]. Despite being one of the oldest and best studied models of a network, E-R model differs from real networks in two crucial ways : it lacks network clustering and it has an unrealistic Poissonian degree distribution [80]. By the following, we introduce two other network models to remedy these two inconvenients.

Models based on Small-world networks got into our sight by the work of Watts and Strogatz [103][102]. Small worlds can be formed by adding a small number of random connections to a lattice [1]. And Newman's work [81] gives another effective way of constructing this kind of networks. Random networks display low clustering but short path lengths since there are many long-range links, whereas small-world networks have high clustering and short path lengths. These characteristics have important implications in the context of epidemics : the high level of clustering means that most infection occurs locally, but short path lengths means that epidemic spreading through the network is rapid [103]. By applying the percolation theory to small-world networks to calculate the threshold, it is found that [72] :

$$\lambda_c = \frac{\sqrt{1 + 12\phi + 4\phi^2} - 1 - 2\phi}{4\phi} = 1 - 4\phi + O(\phi^2) \tag{1.2}$$

where, it is assumed that each individual is linked to its two nearest neighbors and on average to ϕ randomly chosen other individuals.

In 1998, a project to map the World Wide Web has revealed a surprising fact that a few highly connected pages are essentially holding the World Wide Web together. Counting how many Web pages have exactly k links showed that the degree distribution followed a power-law. Following researches observed many real world networks that display this phenomenon, while small worlds, random networks have a power-law degree distribution. Scale-free networks can be constructed dynamically by adding new

1. Lattices display high clustering but long path lengths because connections are established between adjacent individuals in a 2-D grid.

individuals to a network one by one with preferential mechanism. The major contribution of this model is the heterogeneity in numbers of contacts and the existence of hubs (the most highly connected nodes in the network) [28]. Hubs in a network play a pivotal role in the spread and maintenance of infection. Research suggests that the simultaneous elimination of as few as 5 to 15 percent of all hubs can crash a system. Despite some practical difficulties, immunizations targeting hubs could be interesting [67][5][29]. Further research indicated the absence of epidemic threshold in scale-free networks [89]. That is, even weakly contagious viruses will spread and persist in the system.

To summarize this subsection, we focus on the assumptions used for each network-based epidemic model and on the existence of epidemic threshold. Recently, an interesting study has proved the close relationship between the epidemic threshold of a network and the largest eigenvalue of network's adjacency matrix, which can subsume many previous known threshold for special case graphs (E-R, BA power-law, homogeneous) [101].

Dynamics on/of networks

In the previous section, we review SIS and SIR models that study the dynamics on networks. And we review models like E-R networks, small-world networks, and BA scale-free networks that study the dynamics of networks, whose topological structure can have strong impact on the dynamics of nodes. In spite of the success of research mentioned above, the dynamics of networks and the dynamics on networks are still studied seperately. The fatal default is that these models do not take into account the ability to adapt the network topology dynamically in response to the dynamic state of nodes in real world networks [18][53][48].

In the case of the spreading of an infectious disease in a social network, people tend to avoid contact with infected individuals during the outbreak. And these decisions may change the global structure of the network. Based on such intuition, scientists use techniques namely "adaptive networks" to investigate the complex interplay between a time evolving network topology and the dynamics of the nodes. The first attempt is by rewiring process. In traditional SIS models, they use p as the probability that a

susceptible becomes infected, r as the rate of recovery from infection. In addtion to that, rewiring process allows each susceptible, with probability w, to break the link to the infected and forms a new link to another randomly selected susceptible. By using the "moment closure approximation" [57], three coupled ordinary differential equations have been developed [49]. They developed also the epidemic threshold as :

$$\lambda_c = \frac{w}{< k > [1 - \exp(-w/r)]} \tag{1.3}$$

By comparing the analytical calculations from ODE with direct numerical simulation of the full model, they have discovered some interesting results : the existence of a lower order epidemic threshold. Between these two thresholds, a region of bistability appeared where the healthy and endemic state are both stable. For the control of real-world diseases, adaptive rewiring can increase the invasion threshold and the persistence threshold. However, increasing the rewiring rate hardly reduces the size of the epidemic in the endemic state.

The above strategy of rewiring makes sense given that the individuals would have knowledge of the disease status of the rest of the population. As for some asymptomatic disease, however, a random link activation-deletion seems more reasonable during the outbreak. Inspired from the macro-ODE-based compartmental model [69][65], M. Taylor *et al.* used ω as per link deletion rate and α as per potential link creation rate, to adapt the dynamic network case [98]. They constrained the local activation of links by the maximum nodal degree M. They further calculated the epidemic threshold and discovered that the value of M as a local contraint limiting the number of contacts per individual can be used to control and prevent the outbreak of an epidemic.

In [99][44], authors assume that the infection probability as well as the rewiring probability between agents is type-dependent. By introducing f_p as a measure of a disease being intratype or intertype and f_w as a measure of a rewiring choice being intratype or intertype, Bing Wang *et al.* have evaluated the impact of *intratype/intertype* infecting and rewiring on the epidemic threshold [100]. They have discovered that consistency between infec-

ting and rewiring modes speeds up the disease spread, while inconsistency contributes to halting the outbreak.

1.2.3 PageRank computation

PageRank citation ranking was initially introduced in [82] to bring order to the Web. A page has high rank if the sum of the ranks of its inlinks is high. In other words, rank is propagated through links. To use mathematical formalism, we look for a PageRank vector x, which is the dominant eigenvector of the *Google matrix*,

$$A = \alpha P + (1 - \alpha)ve^T,\ 0 \leq \alpha < 1 \tag{1.4}$$

where the matrix P is a column stochastic matrix, called *transition matrix*, representing the outlink structure of the Web, e is the vector $(1, ..., 1)^T$, α is called the damping factor, and the vector v is the teleportation vector, which ensures the uniqueness of the PageRank vector. A difficulty in PageRank model is caused by the existence of dangling nodes [23]. These nodes will result in one or more columns of zeros in transition matrix P. Several ideas have been proposed to deal with this problem [54][33]. Research by the initial PageRank paper [82] indicates that the PageRank could be calculated by removing the links to dangling pages from the web network. However, theoretically this process might generate new dangling pages and iteratively remove all pages from the network. We simply add an artificial loop with probability 1 to these nodes themselves. By this way, diagonal elements corresponding to dangling nodes in matrix P are filled with 1. This handling can be justified by similar arguments as showed in [33].

Many algorithms [47][46][55][21] have been proposed for computing PageRank, a survey can be found here [14]. A PageRank is the eigenvector associated to dominant eigenvalue of the Google matrix. However, in real world applications, the computation of PageRank has two challenging aspects. First, the matrices involved are very large and relies on a sparse matrix-vector product (MVP) kernel. Suppose z is a vector of p-norm 1, Az can be written as $\alpha Pz + (1 - \alpha)v(e^T z)$ where $e^T z$ is a scalar. So the MVP of A is expressed as MVP of a sparse matrix P plus a vector. Otherwise, any direct computation using A is bottlenecked by memory on large networks. In fact,

the Google matrix A becomes a dense matrix due to the part $(1-\alpha)ve^T$. For above reason, algorithms based on MVP might be advantageous. Secondly, the damping factor α is generally very high. For example, in the model of epidemic spread, the virus has the probability $1-\alpha$ to jump randomly from an infected individual to any other individual through some unusual contact. Intuitively, this event happens rarely and for disease spread α must be very close to 1. This is in fact an argument in favor of using Arnoldi-type methods, as opposed to the power method.

In this thesis, we justify the choice of implicitly restarted Arnoldi method (IRAM) [95] in PageRank computation. We discuss some improvements over it to address the two difficulties stated in the previous paragraph. The model of parallelization used is so general that it could be adapted for any modern (possibly future) parallel architecture. Our numerical results show that : (i) the strategies proposed could accelerate the convergence of IRAM for matrices derived from real applications ; (ii) the PageRank-like infection vector could be used as an ally of vaccination campaigns in the actions carried out by health care organizations.

1.3 Contribution

Modeling of epidemic spread benefits a lot from network research to understand infection evolution in a population. Most of the existing work in network based epidemic model tries to answer questions of how a virus will propagate in a real network and how to establish efficient interventions to control the disease.

This work contributes in many aspects as :

• understanding the impact of social graph structure on propagation of virus. Our work addresses the important question about time evolution of infection through a stochastic model, which depends on matrix operations targeting modern supercomputers.

• identifying individuals most likely to spread the disease and establishing interventions accordingly,

• providing an efficient and "light" computation solver for widely spreading epidemics or pandemics while avoiding repeated runs in computer simulation.

• providing a framework that considers the divers relationships among individuals in a population. We argue that the dynamic of underlying contact networks is also taken into consideration.

We demonstrate that PageRank can be computed using numerical methods based on sparse matrix vector product and propose to use implicitly restarted Arnoldi method (IRAM). Our numerical results are quite encouraging. The proposed algorithm is capable of handling very large graphs. Additionally, it is found in Experiment 6.5 that the number of shifts used in IRAM could help to accelerate the convergence of the dominant eigenpair on these graphs. We are conducting experiments for *yahoo* graph.

Chapitre 2

Classical approaches for modelling epidemic spread

The first epidemiological considerations reach back to the 19th century and were conducted by a doctor in London, John Snow. In the course of a cholera epidemic in London Snow could detect the source of the outbreak ? a contaminated well in Broad Street ? by analysing the spatial and social environments of infected people.

2.1 Ordinary differential equation (ODE) systems

The first model for simulating spread of diseases is the SIR model (1927). In its initial form, SIR model is a compartmental differential-equation model that structures the infectioned population in terms of age-of-infection, while using simple compartments for people who are susceptible (S), infected (I) and recovered (R). Two parameters α and β describe the probability that in the case of interaction a susceptible individual becomes infected by a contagious (infected) individual, respectively that an infected individual recovers during one time unit (this leads to a geometric distribution of the duration of infection). The population (S, I, R) can thus be described by the

following system of ordinary differential equations (ODE).

$$\begin{aligned} \frac{dS(t)}{dt} &= -\alpha S(t)I(t), \\ \frac{dI(t)}{dt} &= \alpha S(t)I(t) - \beta I(t), \\ \frac{dR(t)}{dt} &= \beta I(t) \end{aligned} \tag{2.1}$$

One major drawback of these approaches is, that they only reflect a perfectly homogeneous population. If one tries to include different sub-populations or spatial distributions the complexity of the models is quickly growing beyond reason.

2.2 Cellular Automata

Cellular automata models consist of cells on a grid that may change colours at discreet times to represent different states. A cell's state is determined by a set of rules and the state of its neighbours, and therefore the neighbourhood of a cell must be specified. This thesis studies two-dimensional cellular automata models, an example of which is Conway's game of life. Cellular automata have long been used to study biological systems such as this project does [104]. The cellular automaton model is based on the Kermack-McKendrick model. Each cell will have a certain probability of becoming infected :

$$P_{infect} = 1 - (1 - p)^R$$

where p is the probability that a infected cell will transmit the disease to a healthy cell, and R is the number of cells surrounding the healthy cell. Each cell will also have a certain probability of recovering from the disease :

$$P_{recover} = q$$

and the values of p and q are between 0 and 1. P_{infect} and $P_{recovered}$ are then compared to random numbers. If the probability is more than the random number, then the cell will become infected or will recover, otherwise the cell will stay in the same state. For the common cold, cells that are recovered will become susceptible again after a set of updates. The neighbourhood used

in this project is the Moore Neighbourhood. The number of neighbours, n, around a cell is given by the following equation :

$$n = (2r + 1)^2$$

where r is the range. This project uses a Moore Neighbourhood with range 1, giving nine possible neighbours for each cell. The state of each cell will be updated accordingly and displayed with the following colours : green if the cell is susceptible, red if the cell is infected, and black if the cell is recovered. The number of susceptible, infected, and recovered cells is then plotted as a function of time.

2.3 Multi-agent systems

Multi-agent system (MAS) combines agents, environments, interactions and organization. Appeared in the 1990s, the multi-agent system follows the object-oriented programming because it is difficult to handle. Currently MAS allows to transform objects into autonomous computing entities. The basic concept of multi-agent simulation is to represent the behavior of individuals, or to constitute an abstraction of the real world. We can cite various applications of multi-agent system as is in the field of transport, telecommunications, simulation, game theory, information systems, robotics, cooperative systems, interactive games , modeling of complex networks, etc. To sum up, MAS = Agents + Environment + Interactions + Organization.

2.3.1 Agent

The term "agent" is preferred to the term "object" since the term agent involving human abilities. An agent is an entity that thrives in an environment that has certain features :

- Autonomy : the agent has a behavior that is related to his own experience. In the literature, the concept of "autonomy" is associated with at least four concepts :

(a) Design in MAS.

(b) Environment in which it operates.

(c) Objectives.

(d) Motivations.

- Communication : agents interact with each other using a protocol. They can share information or send messages.

- The agents evolve with their environment although they have a partial representation of it.

Thus, the agent will perform actions based on the representation of their environment.

- Agents have knowledge about themselves and other agents.

- Agents have the ability to learn. They can change their behavior based on what they perceive.

- Agents have goals.

- Agents are mobile.

There are various types of agents, such as reactive agents and cognitive agents.

- Reactive agents are defined as agents with no representation of their environment. They have little intelligence which indicates that the shares previously carried out by agents of this type are forgotten. They work by reactions of "stimulus-response". The prime example that can be cited in this case is the ant. One advantage of using the reactive agents in a

simulation is that they can be many in quantity.

- Cognitive agents are smart. They have a very good representation of their environment, have a memory and a collective or individual goal. The human being is a good example of cognitive agent. This type of agent is used to solve more complex problems. We can see that the design of an agent is more or less complex depending on its nature. The use of a system composed solely of reactive or cognitive agents does not lead to a specially balanced state.

- Hybrid agents combine the characteristics of reactive agents and cognitive agents.

We can therefore note that it is crucial to define the behavior of the agent. In general, MAS are made from three models :

1. Eco-agent defined by Ferber [42], is a reactive agent that aims to achieve a state of satisfaction. When it is impossible to reach this state due to another agent, it is attacked by eco-agent.

2. The BDI (Belief Desire Intention) agent [87], is to represent the behavior of a cognitive agent. It has beliefs and desires accordingly. It will try to satisfy them.

3. The POA Agent, Agent Oriented Programming is based on intentional postures according Y. Shoham in [94].

2.3.2 Environment

The environment is defined as the place where agents evolve and interact. It is modeled as a reactive agent. Most often, in the MAS, the environment is spatial. In case where the environment changes during the simulation, we say that the system is dynamic otherwise static.

2.3.3 Interactions

In MAS, interactions play an important role. They include all actions that may arise between the different agents and the environment. There are two types of direct interactions in MAS :

- Those generated by the communication from one agent to another.

- Those concerning the action of an agent over the environment. Indeed, any action of an agent has an impact on the environment and therefore modifies it.

There are also indirect interactions between agents called "interactions brought by the environment." Indeed, the perception changes caused by other agents in the environment constitutes an indirect interaction.

2.3.4 Organization

The organization encompasses all actions of all components. It allows to structure the behavior of agents and their interactions. An agent has freedoms according to the definition of all its actions and the entities to which actions are directed.

Two objectives of MAS are :

- "The creation of distributed artifacts capable of performing complex tasks by cooperation and interaction."

- "Theoretical and experimental analysis of self-organization mechanisms that occur when several autonomous entities interact."

2.4 Conclusion

To summarize this chapter, the classic approach as ODE or cellular automata is mathematically robust that can be easily used to give global information on the epidemic spread. However, they don't take into account the underlying network structure, which proves to be an important factor during the propagation process. Recently, multi-agent system has gained a lot of attention and is widely used in epidemic models. In this work, we use multi-agent system in the simulation experiment to compare with the proposed model of epidemic spread.

Chapitre 3

Propositions of modelling epidemic spread

We study the SIS process of an epidemic spreading over an undirected network. We assume a connected network $G = (N, E)$, where N is the number of nodes in the network and E is the set of edges. Table 1 lists the parameters used.

n	Size of the population
β	Virus infection rate
δ	Virus curring rate
t	Time stamp
$p_{i,t}$	Probability that node i is infected at t
$\tilde{p}_{i,t}$	Probability that node i receives infections

TABLE 3.1: Table of symboles

3.1 Time evolution

3.1.1 simulation based on matrix vector multiplications

Time is assumed to be discrete. During each time interval, the probability that a node i is infected at time t is $p_{i,t}$. The probability that an individual i

will receive infections from his contacts at time t is

$$\tilde{p}_{i,t} = 1 - \prod_{j:\ contacts\ of\ i} (1 - \beta p_{j,t-1}) \tag{3.1}$$

where $(1 - \beta p_{j,t-1})$ represents the probability that the contact j did not pass the infection to i. where $(1 - \beta p_{j,t-1})$ represents the probability that the contact j didn't pass the infection to i. Ignoring second order terms, we have

$$\begin{aligned} \tilde{p}_{i,t} &\approx \sum_{j:\ contacts\ of\ i} (\beta p_{j,t-1}) \\ &= \beta \sum_{j:\ contacts\ of\ i} p_{j,t-1} \end{aligned} \tag{3.2}$$

An individual i will be healthy at time t if

1. i was not infected at time $t-1$ and did not receive infections from his contacts, or
2. i was infected at time $t-1$ but was cured at time t while did not receive infections from his contacts, or
3. i received infections from his contacts but an curring event happened after infections. Since an curring event might happen before or after all infections to i, we consider the probability that this event happend after all infections to be $\frac{1}{2}$.

As a result, the healthy probability of i at time t is

$$\begin{aligned} 1 - p_{i,t} =& (1 - p_{i,t-1})(1 - \tilde{p}_{i,t}) \\ &+ \delta p_{i,t-1}(1 - \tilde{p}_{i,t}) \\ &+ \frac{1}{2}\delta \tilde{p}_{i,t} \end{aligned} \tag{3.3}$$

It follows that

$$p_{i,t} = p_{i,t-1} + \tilde{p}_{i,t} - p_{i,t-1}\tilde{p}_{i,t} - \delta p_{i,t-1} + \delta p_{i,t}\tilde{p}_{i,t} - \frac{1}{2}\delta \tilde{p}_{i,t} \tag{3.4}$$

Ignoring second order terms, we have

$$\begin{aligned} p_{i,t} &\approx p_{i,t-1} + \tilde{p}_{i,t} - \delta p_{i,t-1} - \frac{1}{2}\delta \tilde{p}_{i,t} \\ &= (1 - \delta p_{i,t-1}) + \beta(1 - \frac{1}{2}\delta) \sum_{j:\ contacts\ of\ i} p_{j,t-1} \end{aligned} \tag{3.5}$$

Let p_t be the column vectors $\begin{bmatrix} p_{1,t} \\ p_{2,t} \\ \vdots \\ p_{n,t} \end{bmatrix}$, then

$$p_t = [(1-\delta)I + \beta(1-\frac{1}{2}\delta)A_{dj}]p_{t-1} \tag{3.6}$$

where A_{dj} is the adjacency matrix of the underlying network. Write $A = [(1-\delta)I + \beta(1-\frac{1}{2}\delta)A_{dj}]$, we call it transition matrix, (9) becomes

$$p_t = Ap_{t-1} = A^t p_0 \tag{3.7}$$

and here p_0 is the initial condition vector for the outbreak of epidemics. Entries in this vector will depend on various factors for a specific disease, such as age, sex, vaccination, etc. Moreover, different scenarios of the outbreak as well as different intervention strategies can be simulated by changing the values in p_0.

An example of 5 individuals has been given in Figure 3.1.

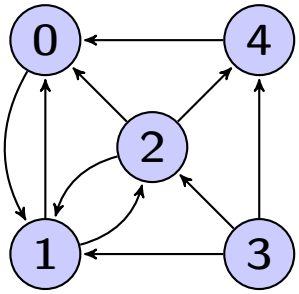


FIGURE 3.1: Small social network of 5 individuals

The set $V = \{0,1,2,3,4\}$ and the set $E = \{0 \to 1, 1 \to 0, 1 \to 2, 2 \to 0, 2 \to 1, 2 \to 4, 3 \to 1, 3 \to 2, 3 \to 4, 4 \to 0\}$. In matrix formulation, these links give the transition matrix A_{dj} as :

$$A_{dj} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \tag{3.8}$$

Assuming that $\beta = 0.2$ and $\delta = 0.24$, we have :

$$A = \begin{pmatrix} 0.76 & 0.176 & 0.176 & 0 & 0.176 \\ 0.176 & 0.76 & 0.176 & 0.176 & 0 \\ 0 & 0.176 & 0.76 & 0.176 & 0 \\ 0 & 0 & 0 & 0.76 & 0 \\ 0 & 0 & 0.176 & 0.176 & 0.76 \end{pmatrix} \tag{3.9}$$

Suppose that each individual stands equal chance to be infected at the outbreak, we have

$$p_0 = \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{pmatrix} \tag{3.10}$$

After the first time step, the infection probability for the population becomes

$$p_1 = Ap_0 = \begin{pmatrix} 0.2576 \\ 0.2576 \\ 0.2224 \\ 0.152 \\ 0.2224 \end{pmatrix} \tag{3.11}$$

The objective here is to simulate the epidemic spread using sparse matrix vector multiplications.

3.1.2 Comparison with agent based stochastic simulations

Agent based stochastic simulations (ASS) have the advantage that you can put all kind of detail into the model at individual level. The simulation proceeds by establishing a set of rules to "guess" all its random parameters. Because of the almost infinite complexities that can be added into an ASS, this can be very slow to simulate epidemics in a large population. Even worse, the result of ASS depends on averaging over repeated runs, which will take large amount of time to ensure the quality.

As shown in the previous paragraph, using matrix operations, however, can avoid these repeated runs. In fact, instead of guessing the value of all

random parameters each time, we estimate and use the probability associated with these uncertain elements. Better still, individual level detail could be handled by using weighted networks. We assume that the weight w_{ij} for a link will depend on the characteristics of individuals i and j (such as sex, age, etc.) as well as that of the viruses. In Figure 3.2, we show a modified version of our example of 5 individuals' network. Concerning the computa-

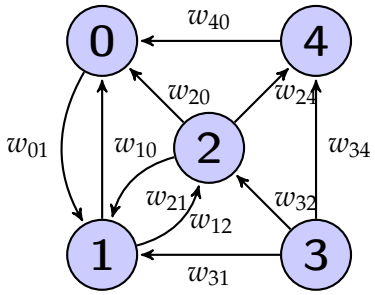


FIGURE 3.2: Weighted version of a small social network of 5 individuals

tion, we simply modify the adjacency matrix of the network to be

$$A_{dj} = \begin{pmatrix} 0 & w_{10} & w_{20} & 0 & w_{40} \\ w_{01} & 0 & w_{21} & w_{31} & 0 \\ 0 & w_{12} & 0 & w_{32} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{24} & w_{34} & 0 \end{pmatrix} \tag{3.12}$$

And the rest of the computation stays the same.

3.2 Containment and mitigation

Here we study the simulation of the effect for different interventions against epidemic spread.
- Pharmaceutical interventions : vaccines and antiviral drugs.
- Non-pharmaceutical interventions : quarantine, school closing and social distancing.

3.2.1 PageRank-like model : vaccines and antiviral drugs

Computational epidemiology arises recently as an interdisciplinary area setting its sight on developing and using computer models to understand

and control the spatiotemporal diffusion of disease within populations [68]. Here we focus on networked epidemiology, which seeks to understand the interplay between individual behaviour and dynamical process on social networks. In other words, this approach investigates the influence of the network topology on epidemic spread.

Agent based stochastic simulations (ASS) put all kinds of details into the model at an individual level. The simulation proceeds by establishing a set of rules to “guess" all its random parameters. This is the most common approach to simulate epidemics in a large population. Network Dynamics and Simulation Science Laboratory has proposed a parallel simulation model “Simdemics" [16][15], designed to scale to the entire United States (300 million people). A similar work can be found in [25], where an individual based influenza simulation model “FluTe" has been proposed. According to the numerical results of these studies, ASS are useful to help establish different pharmaceutical interventions as well as social distancing measures. Furthermore, from a computational point of view, ASS may easily scale up to simulate millions of people in a very efficient way. Nevertheless, one inconvenience of ASS approach is that its result depends on averaging over repeated runs, which could take large amount of time to ensure the quality.

An interesting work [101] has proved the close relationship between epidemic threshold of a network and the largest eigenvalue of network's adjacency matrix, which can subsume many known thresholds for special case graphs (scale-free, homogeneous, etc.). Rather than using ASS, this work employs matrix analysis to study the epidemic spread. In [71], authors have presented some empirical results on the potential usefulness of PageRank for establishing effective vaccination strategies. In [27], authors have proved that by using PageRank vectors, any infection will die out quickly and this process is independent of the size of the whole network. Although the idea to use PageRank vectors in epidemic studies is not new, we have not found any previous studies on discussing the computational aspects of PageRank-like epidemic models. Another novelty of the present work lies in the application of very large real networks for such models.

For pharmaceutical interventions, only a group of people will be vaccinated at the beginning of an outbreak. Our simulations are conducted by

cutting all of the outlinks belonging to an individual in this group. Then the propagation of virus proceeds by time step and these vaccinated individuals could be considered as the dead ends during the epidemic spread.

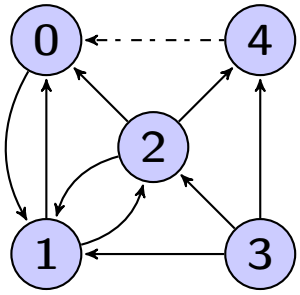


FIGURE 3.3: Small social network of 5 individuals with individual 4 vaccinated

As, show in Figure 3.3, if we vaccinate the individual 4 at the out break, the final transition matrix A becomes

$$A = \begin{pmatrix} 0.76 & 0.176 & 0.176 & 0 & 0 \\ 0.176 & 0.76 & 0.176 & 0.176 & 0 \\ 0 & 0.176 & 0.76 & 0.176 & 0 \\ 0 & 0 & 0 & 0.76 & 0 \\ 0 & 0 & 0.176 & 0.176 & 0 \end{pmatrix} \tag{3.13}$$

The column corresponding to this vaccinated individual reduces to a $0-$column. And the individual 4 will thus never be infected during the course of epidemic.

In order to efficiently establish the vaccination strategy, we propose to make use of Google's pagerank model [82] by analogy. An individual in a social graph is analogous to a webpage in a web graph. The common concept between PageRank model and the proposed epidemic model is the random walk. In PageRank model, the surfer (or walker) starts from a random page, and then selects one of the outlinks from the page in a random fashion. Each page has two states as being visited by surfer or not. The PageRank (importance) of a specific page represents the probability that the surfer is present at this page. In the proposed epidemic model, the virus could be viewed as a walker and its propagation could be viewed as a path that consists of a succession of random steps. Each individual has two states

as being infected or not. The pagerank (importance) of a specific individual represents the probability that the virus reaches this individual during the course of epidemic. To use mathematical formalism, let $G = (V, E)$ be a directed graph with individuals set V and outlinks set E. The graph might be directed. That means, if there is a link $i \to j$ in graph G where $i, j \in V$, $j \to i$ is not necessarily true. For example, blood disease could only happen from the donators to the acceptors. Suppose that the graph G has n individuals with degree $d = (d_1, d_2, ..., d_n)$, where d_j is the number of links individual j has to other individuals. At each time step t, the virus has a state $s_t \in V$, indicating which individual it is on at time t. If $s_t = i$, then at time step $t + 1$, the virus moves to one of his neighbors j chosen uniformly at random from all of i's neighbors.

$$P_{j,i} = P[s_{t+1} = j | s_t = i] = \begin{cases} \frac{1}{d_i} & \text{if } i \longrightarrow j \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

The probability $P(s_t = i)$ for $i \in V$ depends only on s_{t-1} and not on $s_{t-2}, s_{t-3}, ...$, so that $\{s_t\}$ is a Markov chain. $\{s_t\}$ is characterized by its initial state and a stochastic matrix P, given by $P_{j,i} = P(s_t = j | s_{t-1} = i)$ with $P_{j,i} \in [0, 1]$ for all $i, j \in V$ and $\sum_{j \in V} P_{j,i} = 1$. According to Frobenius theorem, $\lambda = 1$ is one of the eigenvalue of the stochastic matrix P and is the biggest eigenvalue. Thus, there is a stationary distribution for the final state of epidemic spread. Let x_i be the probability that individual i is infected during epidemic and we write the stationary distribution as $x = (x_1, x_2, ..., x_n)$ for the whole population. This infection vector x is independent of the starting distribution and has the relationship : $Px = 1 * x = x$. To sum up, the infection vector x's implication in social graph is similar to that of PageRank vector in web graph. The problem consists, as a result, to find the dominant eigenvector with 1 as eigenvalue for the stochastic matrix P of the social graph. In our example,

$$P = \begin{pmatrix} 0 & 1/2 & 1/3 & 0 & 1 \\ 1 & 0 & 1/3 & 1/3 & 0 \\ 0 & 1/2 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 0 \end{pmatrix} \tag{3.15}$$

The first difficulty with this model is the existence of dangling individuals [23], which containing no outlinks. These individuals will result in one or more columns of zeros in transition matrix P. For our example in Figure 3.1, if we delete the link $0 \rightarrow 1$, then the first column of matrix P will contain only zeros. Several ideas have been proposed to deal with this problem [54, 33]. Research by the initial pagerank paper [82] indicates that the pagerank could be calculated by removing the links to dangling pages from the web graph. However, theoretically this process might generate new dangling pages and iteratively remove all pages from the graph. In the context of epidemic spread, dangling individuals could be considered as deadends for virus' random walk process. So an idea is to add a loop with probability 1 to these persons themselves. By this way, diagonal elements corresponding to dangling individuals in matrix P are filled with 1. We adopt this simple solution.

The second difficulty with this model is the problem of non-unique rankings. The phenomenon of "small-world" reveals the clustering effect in social networks. Since very few links exist between clusters, some isolated clusters will break the strong connectivity of graph. It can be shown that the transition matrix P will not yield a unique ranking vector x with such isolated clusters [23]. The common solution is to add a jumping vector to the random walk process :

$$B = \alpha P + (1 - \alpha) v e^T \tag{3.16}$$

where B is disease transition matrix, v is the teleportation vector, e is the vector $[1, ..., 1]^T$ and α, the damping factor, is a positive parameter smaller than 1. The virus has a small probability $(1 - \alpha)$ to jump from any individual to any other individual in a social graph. This would happen, for example, when an infected person (virus carrier) meets someone outside his normal contacts. Considering the preferential attachment of social scale-free networks [10], we choose v to be proportional to individuals' degree and normalizes it by "1-norm". In short, there is a small probability that an individual establishes a new temporary link with someone who already has

many links.

$$v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}, v_i = \frac{d_i}{\sum_{i=1}^{i=n} d_i} \tag{3.17}$$

All column sums of the new transition matrix B are 1, so B is still a stochastic matrix with dominant eigenvalue equal to $1 : Bx = x$. In our example, $d = (1, 2, 3, 3, 1)$, $\sum_{i=0}^{i=4} d_i = 10$. By taking $\alpha = 0.9$, we have :

$$\begin{aligned} B &= \alpha P + (1-\alpha) v e^T \\ &= 0.9 * \begin{pmatrix} 0 & 1/2 & 1/3 & 0 & 1 \\ 1 & 0 & 1/3 & 1/3 & 0 \\ 0 & 1/2 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 0 \end{pmatrix} \\ &+ 0.1 * \begin{pmatrix} 1/10 \\ 2/10 \\ 3/10 \\ 3/10 \\ 1/10 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}^T \\ &= \begin{pmatrix} 0.01 & 0.46 & 0.31 & 0.01 & 0.91 \\ 0.92 & 0.02 & 0.32 & 0.32 & 0.02 \\ 0.03 & 0.48 & 0.03 & 0.33 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.01 & 0.01 & 0.31 & 0.31 & 0.01 \end{pmatrix} \end{aligned} \tag{3.18}$$

If we vaccinate individual number 4 at the out break, the final transition matrix A becomes

$$B = \begin{pmatrix} 0.01 & 0.46 & 0.31 & 0.01 & 0 \\ 0.92 & 0.02 & 0.32 & 0.32 & 0 \\ 0.03 & 0.48 & 0.03 & 0.33 & 0 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0 \\ 0.01 & 0.01 & 0.31 & 0.31 & 0 \end{pmatrix} \tag{3.19}$$

The column corresponding to this vaccinated person reduces to a $0-$column. And he will thus never be infected during the course of epidemic. Further insight is given by numerical experiments in chapter 8.

It must be noticed that the proposed model is based on SIS epidemiological process where, at a given time, each individual can be susceptible (S) or infected (I). So we suppose that individuals recover with no immunity to the disease, that is, individuals are immediately susceptible once they have recovered. For example, for flu disease, people may very unfortunately get ill again once recovered. Furthermore, the characteristic of individuals as well as that of virus are not taken into account in the example. But they can be taken into account by using weighted networks. Infection vector x could help health officials to decide the relative importance of different agents in a population facing an epidemic. This is especially useful when the resource of vaccination are limited during the beginning phase of an urgent outbreak. Priorities should be given to those individuals with bigger ranking in x. In addition, a fast computation of this vector could be expected thanks to the efficient implementation of eigenvalue algorithm.

Dangling individuals

In the proposed epidemic model, a decision must be made to deal with the “dangling individuals". There are several possibilities for their existence. For example, a person with innate immunity against certain disease, a person in quarantine after getting the disease, or someone who dies, etc. There is a difference between a dangling individual and a dangling web page. A dangling page contains no outlinks. However, in most cases, a dangling individual will still have some social connections. They are called dangling because they somehow cannot spread epidemic after getting it. In other words, the outlinks of the dangling individuals will be temporally disabled.

A dangling individual may have high PageRank as normal people. PageRank model computes the score for a person based on individuals that link to it, rather than based on features (such as dangling) of the person. Someone in contact with these dangling individuals contributes to their scores.

Research by the initial PageRank paper [82] indicates that the PageRank could be calculated by removing the links to dangling pages from the web network. However, theoretically this process might generate new dangling pages and iteratively remove all pages from the network. The work by Lee et al. lumps the dangling nodes together into one new state [60]. A rigorous

justification for this approach can be found in [54]. The solution proposed in [33] adds artificial links to the dangling nodes. The idea is to force the transition matrix P to be stochastic.

We simply add an artificial loop with probability 1 to the dangling individuals. The disease will be "trapped" once reaching them. In this way, their corresponding diagonal elements in matrix P are filled with 1. This handling can be justified by similar arguments as shown in [33]. A virtual $(n+1)^{th}$ node is added to a n-sized social network. Let $\mathcal{C}$ denote the set of non-dangling nodes and $\mathcal{D}$ denote the set of dangling nodes. Suppose the size of $\mathcal{C}$ is $|\mathcal{C}| = m$, we have $|\mathcal{D}| = n - m$. Apart from the artificial loops added to dangling nodes, we add new edges $(i, n+1)$ for $i \in \mathcal{D}$ and $(n+1, i)$ for $i \in \mathcal{C}$. We construct a linear system as follows,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \alpha C & 0 & e^{(1)}/m \\ \alpha D & \alpha I & 0 \\ (1-\alpha)(e^{(1)})^T & (1-\alpha)(e^{(2)})^T & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.20}$$

where, if d_j is the out degree of the node j, matrices $C(m \times m)$ and $D((n-m) \times m)$ are defined by :

$$c_{ij} = \begin{cases} d_j^{-1} & \text{if } i, j \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases} \qquad d_{ji} = \begin{cases} d_j^{-1} & \text{if } i \in \mathcal{C}, j \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases}$$

and $e^{(1)}, e^{(2)}$ are column vectors of 1's of conforming dimension.

Theorem 1. *The linear system (3.20) computes the PageRank for dangling nodes as well as non-dangling nodes in the network.*

Démonstration. Rewrite the equation (3.20) as

$$x = \alpha C x + \frac{e^{(1)}}{m} z \tag{3.21}$$

$$y = \alpha D x + \alpha y \tag{3.22}$$

$$z = (1-\alpha)(e^{(1)})^T x + (1-\alpha)(e^{(2)})^T y \tag{3.23}$$

It follows,

$$z = [(1-\alpha)(e^{(1)})^T + \alpha(e^{(2)})^T D]x \tag{3.24}$$

We rewrite the equations (3.21) and (3.24) as

$$\begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} \alpha C & \frac{e^{(1)}}{m} \\ (1-\alpha)(e^{(1)})^T + \alpha(e^{(2)})^T D & 0 \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} \tag{3.25}$$

The matrix in the system (3.25) is a stochastic matrix, so that the vector x corresponds to the PageRank of non-dangling nodes $\mathcal{C}$. The PageRank for dangling nodes can then be computed by

$$y = \frac{\alpha}{1-\alpha} Dx \tag{3.26}$$

□ □

Noticing that, by adding a virtual node, the initial PageRank problem (3.16) can be written as

$$\begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} \alpha P & v \\ (1-\alpha)e^T & 0 \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix}$$

which takes a similar form as (3.25).

Given the limited supplies of vaccines and antiviral drugs, non-pharmaceutical interventions are likely to dominate the public health response to any pandemic, at least in the near term.

3.2.2 Non-pharmaceutical interventions

Quarantine

We simulate the temporary isolation and restriction of the movement of an exposed individual by cutting all his inlinks and outlinks for a period of time. As shown in Figure 3.4, if quarantine is used for individual 4 for 10 time steps, we could simply put the nonzero elements in the last row/column of A to 0 for the first 10 matrix vector multiplications.

$$A = \begin{pmatrix} 0.76 & 0.176 & 0.176 & 0 & 0 \\ 0.176 & 0.76 & 0.176 & 0.176 & 0 \\ 0 & 0.176 & 0.76 & 0.176 & 0 \\ 0 & 0 & 0 & 0.76 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.27}$$

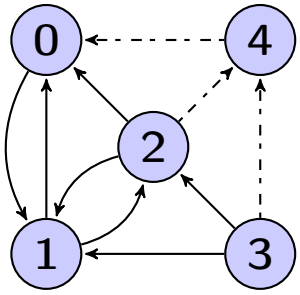


FIGURE 3.4: Small social network of 5 individuals with quarantine on individual 4

School closing

Assuming that the school of the kid 0 and 1 is closed due to the epidemic, then these two children will not be able to get in touch during the epidemic (see Figure 3.5). As a result, the transition matrix A becomes

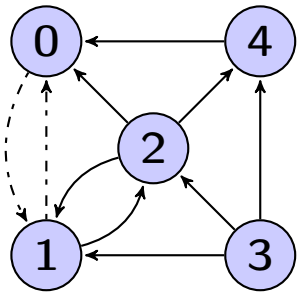


FIGURE 3.5: Small social network of 5 individuals with school closing for kids 0 and 1

$$A = \begin{pmatrix} 0.76 & \underline{\mathbf{0}} & 0.176 & 0 & 0.176 \\ \underline{\mathbf{0}} & 0.76 & 0.176 & 0.176 & 0 \\ 0 & 0.176 & 0.76 & 0.176 & 0 \\ 0 & 0 & 0 & 0.76 & 0 \\ 0 & 0 & 0.176 & 0.176 & 0.76 \end{pmatrix} \tag{3.28}$$

Social distancing

This intervention can be simulated by reducing the infection rate β. If we reduce the infection rate β to 0.1 in the example, the transition matrix A

becomes

$$A = \begin{pmatrix} 0.76 & 0.176 & 0.176 & 0 & 0.176 \\ 0.176 & 0.76 & 0.176 & 0.176 & 0 \\ 0 & 0.176 & 0.76 & 0.176 & 0 \\ 0 & 0 & 0 & 0.76 & 0 \\ 0 & 0 & 0.176 & 0.176 & 0.76 \end{pmatrix} \tag{3.29}$$

3.3 Combination of the two previous approaches : modelling based on numerical algebraic operations

As shown in Figure 3.6, a social network is considered to be a family of different kinds a relationships $\{\mathcal{R}_i\}$. These different relationships will ge-

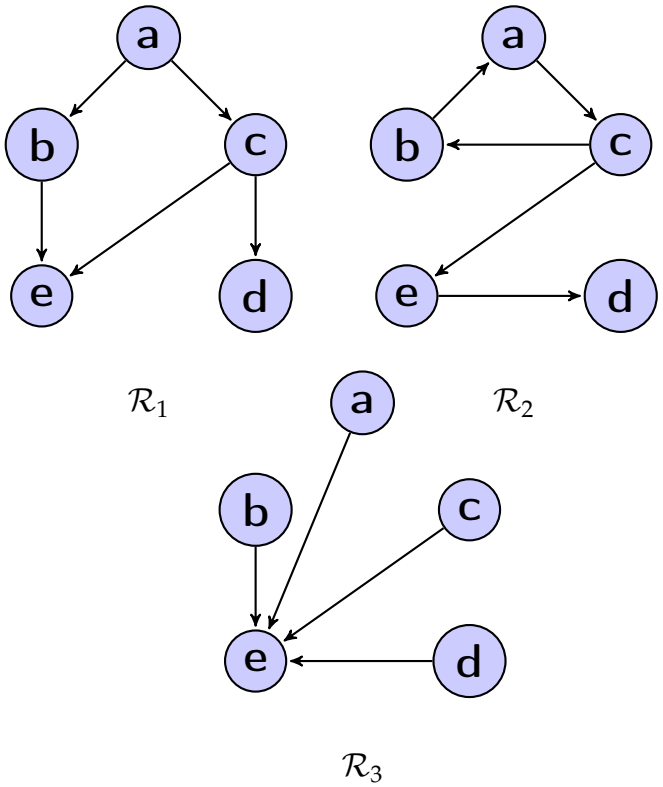


FIGURE 3.6: Social network of 3 relationships for 5 individuals

nerate different diffusion betaviours, based on which the global epidemic dynamics could be inferred. Besides, similar to adaptive networks, a random event that a susceptible node avoids infections by breaking its links

to its infected neighbours while it enhances the connections with other susceptible nodes by creating links to them is considered in our framework as a borelian ω_i. By consequence, one diffusion step on a dynamic network can be viewed as the averaging result of random events on a set of relationships. Lets take a small example of 3 relationships among 4 individuals, which is described by Figure 3.7, Figure 3.8 and Figure 3.9.

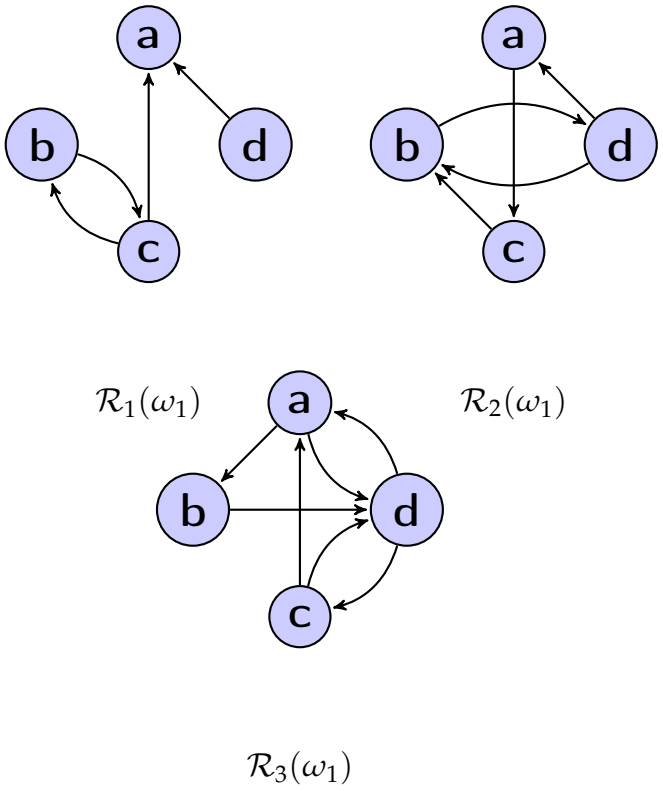


FIGURE 3.7: Stochastic network, random event ω_1

Here, $\Omega = \{\omega_1, \omega_2, \omega_3\}$, $\mathcal{A} = \mathcal{P}(\Omega)$ and $p(\omega_i) = \frac{1}{3}, \forall i = 1, 2, 3$. Let us consider $E = \{a, b, c, d\}$, then

$\Gamma_1(\omega_1, a) = \{a\}, \Gamma_1(\omega_1, b) = \{b, c\}, \Gamma_1(\omega_1, c) = \{a, b, c\}, \Gamma_1(\omega_1, d) = \{a, d\}$

$\Gamma_2(\omega_1, a) = \{a, c\}, \Gamma_2(\omega_1, b) = \{b, d\}, \Gamma_2(\omega_1, c) = \{b, c\}, \Gamma_2(\omega_1, d) = \{a, b, d\}$

$\Gamma_3(\omega_1, a) = \{a, b, d\}, \Gamma_3(\omega_1, b) = \{b, d\}, \Gamma_3(\omega_1, c) = \{a, c, d\}, \Gamma_3(\omega_1, d) = \{a, c, d\}$

$\Gamma_1(\omega_2, a) = \{a\}, \Gamma_1(\omega_2, b) = \{b\}, \Gamma_1(\omega_2, c) = \{a, b, c\}, \Gamma_1(\omega_2, d) = \{a, d\}$

$\Gamma_2(\omega_2, a) = \{a, c\}, \Gamma_2(\omega_2, b) = \{b, d\}, \Gamma_2(\omega_2, c) = \{c\}, \Gamma_2(\omega_2, d) = \{a, c, d\}$

$\Gamma_3(\omega_2, a) = \{a, b\}, \Gamma_3(\omega_2, b) = \{b, d\}, \Gamma_3(\omega_2, c) = \{a, c\}, \Gamma_3(\omega_2, d) =$

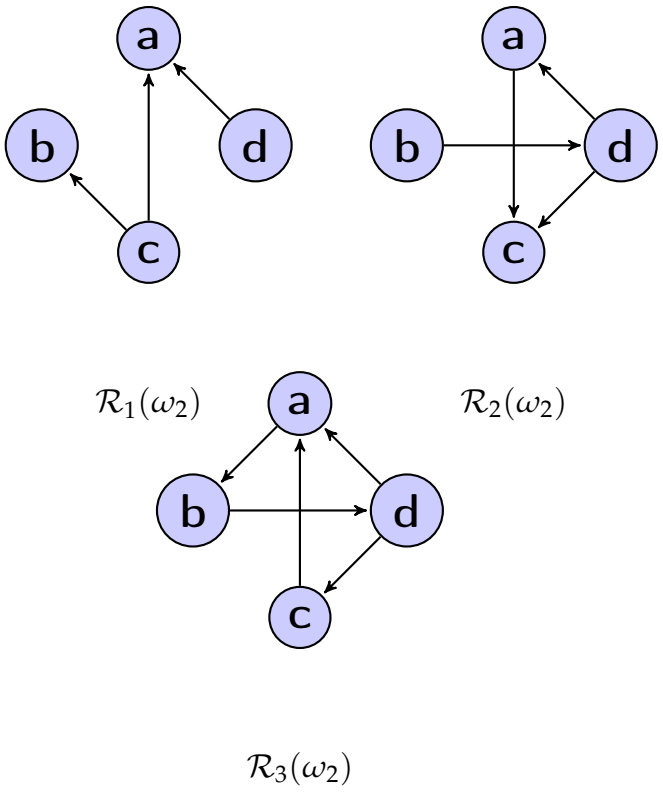


FIGURE 3.8: Stochastic network, random event ω_2

$\{a, c, d\}$
$\Gamma_1(\omega_3, a) = \{a\}, \Gamma_1(\omega_3, b) = \{b\}, \Gamma_1(\omega_3, c) = \{a, c, d\}, \Gamma_1(\omega_3, d) = \{b, d\}$
$\Gamma_2(\omega_3, a) = \{a, c\}, \Gamma_2(\omega_3, b) = \{b, d\}, \Gamma_2(\omega_3, c) = \{c\}, \Gamma_2(\omega_3, d) = \{c, d\}$
$\Gamma_3(\omega_3, a) = \{a, b\}, \Gamma_3(\omega_3, b) = \{b, d\}, \Gamma_3(\omega_3, c) = \{c\}, \Gamma_3(\omega_3, d) = \{a, c, d\}$
Within a specific random event, we could define a different set of parameters (see Table 1) for each of this relationship. Furthermore, with each relationship $\mathcal{R}_i$, we associate a weight parameter w_i to signify its relative importance on the final epidemic propagation.

3.4 Conclusions

In this chapter, we present a PageRank-like model in the first place. The idea is to use a Markov chain to model the spread of disease. With this simplified model, a ranking vector can be derived to help establish efficient vaccination strategy. The model can be easily calculated by numerical linear algebra methods. Indeed, computation plays a more and more important

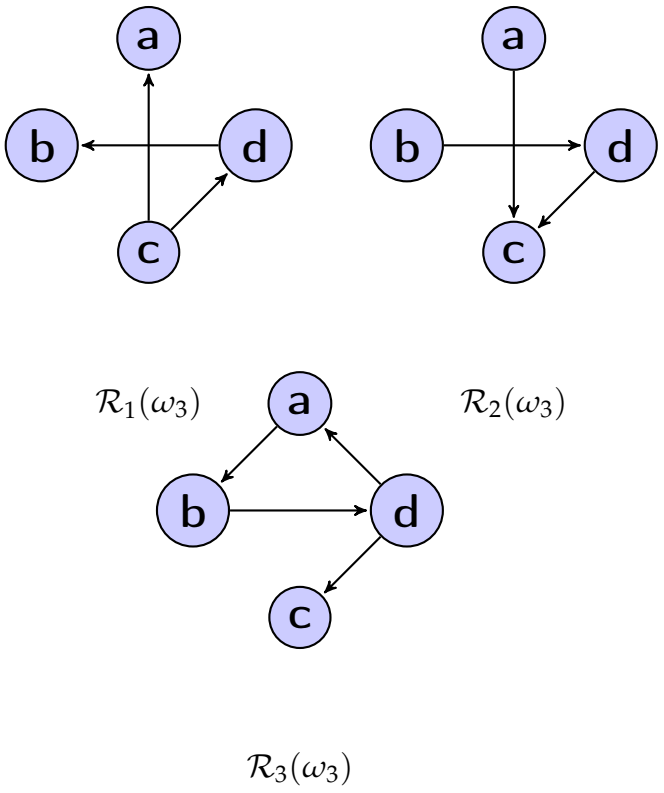


FIGURE 3.9: Stochastic network, random event ω_3

role in epidemiology simulation. However, PageRank-like approach does not consider the evolving structure of the network. In addition, there are various relationships between individuals, which could be either quantitative or qualitative. The complex nature of human contact cannot be modeled by Markov chain alone.

In the second part of this chapter, we propose a multiple stochastic network, which takes into account various kinds of human relationships. Its stochastic property can be used to further explore the dynamic of network structure. This new model is in the same time more realistic and maintains the computational advantages of PageRank-like model. A topological formalism (pretopology) is also presented in the section conclusion of this thesis. A list of new topological parameters make this approach very promising to model some complex social phenomenon. We mention this pretopological modeling technique as a direction of future work.

In the following chapters, we focus on the computational aspects of the proposed models.

Chapitre 4

Multiple implicitly restarted Arnoldi method

4.1 Krylov subspace methods

Arnoldi in 1951 [7] proposed a method which is a variant of the Krylov projection methods. The Arnoldi method is an efficient technique which permits to compute an approximation of desired eigenpairs of an n-size matrix A by those of an $m \times m$ matrix representing A in an m-size Krylov subspace (with $m \leq n$). The Arnoldi method is best at finding a solution to an eigenproblem with well-separated eigenvalues. A drawback of this method is the expense of too much memory space when m is large. This problem can be remedied by restarting the method as proposed in 1980 by Saad [90]. This approach, called explicitly restarted Arnoldi method (ERAM), allows to restart the Arnoldi projection with a better subspace. Indeed, this approach offers to choose a small Krylov subspace ($m << n$). Then, if the accuracy of the desired Ritz elements computed by Arnoldi method is not satisfactory, ERAM restarts the process using a new Krylov subspace. This new subspace differs from the last one by its initial vector which is formed by an explicit combination of the computed Ritz elements. Despite the simplicity, the formation of a restarting vector for the next iteration using the approximated eigenvectors of the current iteration might not be good. The restarting is difficult because one new starting vector must be defined as an explicit linear combination of wanted Ritz vectors. If this combination is not

carefully chosen, it can lead to a very bad selection for the new starting vector. Moreover, when the starting vectors are complex the cost will increase. Saad proposed some special coefficients for the combination of Ritz vectors such as the weighted linear combination [90]. As he mentioned it, this method may not well work in practice for many eigenproblems [92]. Moreover, the problem of the choice of the size of the subspace remains. An approach based on the Arnoldi projection onto several Krylov subspaces is proposed by Emad et al [35]. The latter are formed with different initial vectors and have different sizes. This technique is called multiple explicitly restarted Arnoldi method (MERAM). It allows to update the restarting vector of each ERAM process by taking into account the eigen-information of interest obtained by all ERAM processes. MERAM improves often the convergence of ERAM but the restarting issues intrinsic to ERAM remain [35].

In order to improve the Arnoldi method, Sorensen has suggested an efficient technique which makes use of the QR algorithm to restart the Arnoldi projection [95]. His approach permits to restart the Arnoldi process with an efficient and numerically stable formulation. This approach which is called implicitly restarting Arnoldi method (IRAM) was analyzed, implemented and validated, among others, in [95, 61, 62, 96, 63, 70]. As in ERAM, implicity restarted Arnoldi method makes use of Arnoldi projection to approximate the desired eigenpairs of a large matrix A. If the accuracy of these Ritz elements is not satisfactory, IRAM applies a QR shifted algorithm on the $m \times m$ matrix which represents A in the projection subspace. As these are the non desired eigenvalues which are chosen for shifts, the upper-left block of the matrix issued from QR algorithm concentrates the information corresponding to the desired eigenvalues. IRAM completes an $m-$size Arnoldi projection starting with the submatrix representing this block whose size is the number of wanted eigenvalues. This is equivalent to restart the Arnoldi process with a new initial vector computed implicitly. Morgan showed that IRAM is much better than the other restarting Arnoldi methods such as explicitly restarted ones [74]. However, the problem of choosing the size of subspace remains.

In restarted Arnoldi methods, in order to improve the quality of the subspace during the iterations, only the initial vector is taken into account. The

idea is to take into account both the initial vector and the size of the subspace. Multiple implicitly restarted Arnoldi method (MIRAM) [35, 39], is based upon the projection of the problem on several Krylov subspaces instead of a single one. These subspaces differ by their size while the subspaces in multiple restarted methods such as MERAM can differ by their size and their initial vectors [35]. MIRAM makes use of Arnoldi method to compute the Ritz elements of a large matrix A in a set of ℓ nested Krylov subspaces $\mathbb{K}_{m_i,v}$ (for $i = 1, \ldots, \ell$) with $\mathbb{K}_{m_i,v} \subset \mathbb{K}_{m_{i+1},v}$. If the accuracy of the desired Ritz elements calculated in none of these subspaces is satisfactory, MIRAM selects the "best" of these subspaces. This subspace is one that contains the "best" current Ritz elements. Then a QR shifted algorithm will be applied to the $m_{best} \times m_{best}$ matrix which represents A in this m_{best}-size projection subspace. As these are the non desired eigenvalues which are chosen for shifts, the leading submatrix issued from QR algorithm concentrates the information corresponding to the desired eigenvalues. MIRAM completes Arnoldi projections on ℓ nested Krylov subspaces starting with this submatrix whose size is the number of wanted eigenvalues. An improved version of this method including the comparison with other methods can be found here [40].

One of the well known problems of the restarted iterative methods is the sensibility of the convergence in the small perturbation on the subspace size. Indeed, they could not converge with a subspace and converge with the same reduced/extended subspace with nearby sizes. MIRAM allows to remedy to this problem by making choice of the "best" size among these subspace sizes. Another advantage of this technique is the better property of convergence with almost the same time complexity relative to IRAM. Our experiments showed a very good acceleration of convergence with respect to the implicitly restarted Arnoldi method.

Notations Throughout this chapter, we use the following notations :

A	$n \times n$ matrix,
H_m	$m \times m$ projected matrix (upper Hessenberg matrix),
V_m	$n \times m$ matrix, orthogonal basis in the Krylov subspace,
f_m	residual vector of lenght n,
k	number of wanted eigenvalues,
$\{\lambda_i^{(m)}\}_{i=1}^m$	eigenvalues of H_m (Ritz values of A),
$\{y_i^{(m)}\}_{i=1}^m$	eigenvectors of H_m,
$\{u_i^{(m)}\}_{i=1}^m$	Ritz vectors of A ($u_i^{(m)} = V_m y_i^{(m)}$),
$\rho_{i,m}$	residual norm $\rho_{i,m} = \|(A - \lambda_i^{(m)} I)u_i^{(m)}\|_2$
$\{\mu_i^{(m)}\}_{i=1}^p$	subset of unwanted eigenvalues of H_m,
$\|A\|_F$	Frobenius norm
$\mathbb{K}_{m,v_1}$	Krylov subspace spanned by $v_1, Av_1, \ldots, A^{m-1}v_1$,
it	number of iterations (number of restarts + 1),
z_n	$n \times 1$ vector $(1, 1, \cdots, 1)^T$
s_n	$n \times 1$ vector $(1, 1, 0.1, \cdots, 0.1)^T$
t_n	$n \times 1$ vector $(1, 1, 0, \cdots, 0)^T$
$H_m(1:k, 1:k)$	the leading $k \times k$ submatrix of H_m.

Dynamic selection of restart parameters in Arnoldi methods have been considered previously. Duff and Scott in [32] developed a subspace algorithm combined with Chebychev acceleration. They select dynamically the size of the subspace and the degree of the Chebychev polynomial at each iteration. Stathopoulos, Saad and Wu proposed in [97] a technique, called thick restarting, that restarts the Arnoldi algorithm with more eigenvectors that is actually required. A dynamic thick restarting scheme which adjusts the number of retained Ritz vectors at each cycle in IRAM is proposed and the question of which and how many eigenvectors to retain is addressed for symmetrical eigenproblems.

Some authors suggested more similar approaches to the one proposed in this paper. Baker et al. proposed in [9] a simple strategy and provide some heuristic explanation for its effectiveness. The authors define a range of subspace sizes whose minimum and maximum values are respectivily m_{min} and m_{max} and they choose, according to some criterion the subspace size m_i of the ith restart in this discrete interval. Their strategy checks the

convergence rate $\| r_{i+1} \|_2 / \| r_i \|_2$ at the end of each restart cycle, where r_i is the residual vector of ith step. The subspace size is initialized by m_{max}. When stagnation is detected, they decrease the restart parameter by a small number d at each cycle until reaching m_{min}. At that point, they increase m_i up to the maximum value m_{max}. For the iterative solution of non-symmetric linear systems by deflated GMRES, Moriya and Nodera proposed in [75] a similar dynamic switching approach for the Krylov subspace dimension. Their strategy consists to combine the deflated GMRES algorithm and the determination of a restart parameter m dynamically. Indeed, in order to decrease the computation cost, the authors propose to begin with a small restart parameter m_s. If stagnation in encountered then the restart parameter is switched to a larger value m_l. When the restart value is m_l and the stagnation disappears then the restart parameter is switched again to m_s. They use as a criterion for stagnation the angle between the residual vector and search vectors which could be easily computed during a run of GMRES. Dookhitram et al. proposed in [31] a comparable approach to accelerate the convergence of IRAM which is based on a relationship between the residual of the current restart cycle of IRAM and the residual in the previous cycle. Despite the similarity, their technique differs from that proposed by Moriya and Nodera for linear systems since unlike the latter, they do not initialize any angle to avoid a problem dependent strategy and also their switching strategy is based on a different relationship between the residual of the current step and the residual of the previous step.

In the approach proposed in this paper, the dynamic determination of subspace size parameter is monitored and *used* inside the current restart cycle while in the methods cited above, this determination is done in current restart cycle for being used in the next restart cycle. Assuming that $E = \{m_{min}, \ldots, m_{max}\}$ is the set of subspace sizes for all of these methods. It can be said that in a given restart cycle, MIRAM uses entire values of this set while other use only a local area of it (usually reduced to a single value).

4.1.1 Arnoldi method and its implicit restarting variant

Let A be a complex non-Hermitian matrix of dimension $n \times n$, v an $n-$size initial guess and $v_1 = v/\|v\|_2$. The well-known Arnoldi process generates an orthogonal basis $v_1, \cdots, v_m$ of Krylov subspace $\mathbb{K}_{m,v_1} = span\{v_1, Av_1, \cdots A^{m-1}v_1\}$ by using the Gram-Schmidt orthogonalization process.

Definition : For $A \in \mathbb{C}^{n\times n}$, a relation of the form $AV_m = V_m H_m + f_m e_m^T$ is called an m-step Arnoldi factorization, where $H_m \in \mathbb{C}^{m\times m}$ is an upper Hessenberg matrix with non-negative subdiagonal elements, $V_m \in \mathbb{C}^{n\times m}$ is a matrix with orthonormal columns and $V_m^H f_m = 0$.

This factorization can be used to reduce the eigenproblem with the large order matrix A to a problem with a smaller order matrix H_m. If $y_i^{(m)}$ is an eigenvector of H_m associated with the eigenvalue $\lambda_i^{(m)}$ and $u_i^{(m)} = V_m y_i^{(m)}$ then $\|(Au_i^{(m)} - \lambda_i^{(m)} u_i^{(m)})\|_2 = \|(AV_m - V_m H_m)y_i^{(m)}\|_2 = |\beta_m e_m^T y_i^{(m)}|$ and $(u_i^{(m)}, \lambda_i^{(m)})$ is an approximate eigenpair for A. The number $|\beta_m e_m^T y_i^{(m)}|$ is called Ritz estimate for the Ritz eigenpair $(u_i^{(m)}, \lambda_i^{(m)})$ where $\beta_m = \|f_m\|_2$.

By using this process, the eigenproblem of size n is replaced by an eigenproblem of size m (with $m \leq n$). If the desired eigenvalues are well-separated, this technique could offer good approximation for them. Moreover, only some basic linear algebra computations are necessary to compute these approximations.

In this process, the choice of m, the size of the subspace is empirical and could be done according to the number of desired eigenvalues and/or the size of the problem to solve [91]. Meanwhile, an m too large generates too high computation cost. The restarting approach proposed by Saad allows to choose m small but to improve the quality of the projection subspace by improving the initial vector v_1. The restarting strategy in this approach, called explicitly restarted Arnoldi method (ERAM), is a critical part. Saad [91] proposed to restart the Arnoldi method with a preconditioning vector in order to force it to be in the desired invariant subspace. It concerns a polynomial preconditioning applied to the starting vector of ERAM. Nevertheless, even with an optimized preconditioning vector, ERAM may not well work in practice for many eigenproblem [92]. Moreover, the

problem of the choice of the size of the subspace remains.

Implicitly restarted Arnoldi method This variant of the Arnoldi method based on restarting technique is called implicitly restarted Arnoldi method (IRAM). That is a technique that combines the implicitly shifted QR mechanism with an Arnoldi factorization and can be viewed as a truncated form of the implicitly shifted QR-iteration [96]. This method involves an implicit application of a polynomial in A to the starting vector. IRAM allows to compute a few eigenvalues ($k \leq m$) such as those of the largest real part or the largest magnitude. An m-step Arnoldi factorization

$$AV_m = V_m H_m + f_m e_m^T, \tag{4.1}$$

is compressed to a factorization of length k with the eigen-information of interest. This is achieved using QR steps to apply $p = m - k$ shifts implicitly. The results after the shift process and equating the first k columns on both sides are

$$AV_m^+ = V_m^+ H_m^+ + f_m e_m^T Q, \tag{4.2}$$

where $V_m^+ = V_m Q$, $H_m^+ = Q^T H_m Q$, and $Q = Q_1 Q_2 \cdots Q_p$ with Q_j the orthogonal matrix in QR process associated with the shift $\mu_j^{(m)}$ and

$$AV_k^+ = V_k^+ H_k^+ + f_k^+ e_k^T, \tag{4.3}$$

with $f_k^+ = V_m^+ e_{k+1} \widehat{\beta}_k + f_m \sigma_k$ where $\widehat{\beta}_k = H_m^+(k+1, k)$ and $\sigma_k = Q(m, k)$. Using this as a starting point one can apply p additional steps of the Arnoldi process to obtain an m-step Arnoldi factorization. Each shift cycle involves the implicit application of a polynomial in A of degree p to the starting vector $v : \psi(A)v$ with $\psi(\lambda) = \Pi_{j=1}^{p}(\lambda - \mu_j^{(m)})$. The roots of this polynomial are the shifts used in the QR algorithm [96]. The resulting algorithm takes the form of the algorithm 1.

The stopping criterion in above algorithm can be computed by the expression called Ritz estimate : ***(a)-*** $|\beta_m e_m^T y_i^{(m)}|$, or by its mathematically equivalent explicit formula of the residual norm : ***(b)-*** $\|(Au_i^{(m)} - \lambda_i^{(m)} u_i^{(m)})\|_2$ (for $i = 1, \cdots, k$). Criterion (a) has a computational cost much lower than that of

Algorithm 1: Implicitly restarted Arnoldi method

Input : (A, V_m, H_m, f_m) with $AV_m = V_m H_m + f_m e_m^T$ an m-step Arnoldi factorization

For $it = 1, \ldots,$ until convergence

1. - Compute $\sigma(H_m)$ the eigenvalue of H_m and their associated eigenvectors,
   - Compute residual norm, if convergence stop,
2. Select set of $p = m - k$ shifts $(\mu_1^{(m)}, \cdots, \mu_p^{(m)})$, based upon $\sigma(H_m)$ or other information and set $q^T \leftarrow e_m^T$,
3. For $j = 1, 2, \ldots, p$
   - Factor $[Q_j, R_j] = qr(H_m - \mu_j^{(m)} I)$;
   - $H_m \leftarrow Q_j^H H_m Q_j$; $V_m \leftarrow V_m Q_j$;
   - $q^H \leftarrow q^H Q_j$;
4. Set $f_k \leftarrow v_{k+1}\widehat{\beta}_k + f_m \sigma_k$, $V_k \leftarrow V_m(1:n, 1:k)$, $H_k \leftarrow H_m(1:k, 1:k)$
5. Beginning with the k-step Arnoldi factorization $AV_k = V_k H_k + f_k e_k^T$, Apply p additional steps of the Arnoldi process to obtain a new m-step Arnoldi factorization $AV_m = V_m H_m + f_m e_m^T$

(b). However, criterion (b) may better represent the residual corresponding to the Ritz elements and is more reliable when rounding errors are present. This is because the expression of (b) contains computed Ritz elements and thus takes into account the rounding errors in their calculation. It should be noted that, as explained in [96, 83], when A is Hermitian, the relation ((a)=(b)) may be used to provide computable rigorous bounds on the accuracy of the eigenvalues of H_m as approximations to eigenvalues of A. When A is non-Hermitian the possibility of non-normality precludes such bounds and one can only say that the residual norm $\|(Au_i^{(m)} - \lambda_i^{(m)} u_i^{(m)})\|_2$ is small if $|\beta_m e_m^T y_i^{(m)}|$ is small.

Note that if $v = \sum_{j=1}^{n} \gamma_j u_j$, the implicit restarting Arnoldi method with exact shifts provides a specific selection of expansion coefficients γ_j for a new starting vector as a linear combination of the current Ritz vectors for desired eigenvectors. Implicit restarting provides a way to extract eigen-

information of interest from large Krylov subspaces while avoiding the storage and numerical difficulties. This is done by continually compressing eigen-information of interest into an k-dimensional subspace of fixed size. This means that IRAM continues an m-step Arnoldi factorization, having kept all Ritz vectors of interest.

Time and space complexities of IRAM We assume that $m << n$. Therefore, in the time complexity expression of IRAM we can disregard terms not containing n. Let it be the number of iterations excluding the $input$ step in the above algorithm. The cost of IRAM in terms of matrix-vector products for it iterations with `(a)` criterion is $[m + (it - 1) \times (m - k)]$. Indeed, in the first iteration the number of matrix-vector products is m and for each of the other restart cycles, the number of matrix-vector products is $p = (m - k)$. The cost of IRAM will be increased by $it \times k$, if `(b)` criterion is used. Noted that the cost of orthogonalization in Arnoldi process is about $O(m^2 n)$. When A is sparse and p is large, this cost of orthogonalization may dominant the computation.

For space complexity, in addition to A, the method keeps m vectors of length n and an $m \times m$ Hessenberg matrix, which gives $O(nm + m^2/2)$

4.2 Multiple implicitly restarted method

The purpose of restarting m-step Arnoldi factorization is to improve the quality of the initial Krylov subspace $\mathbb{K}_{m,v}$. This objective can be achieved by improvement of the vector v and/or the subspace size m. Indeed, the information obtained through the m-step Arnoldi factorization process is completely determined by the choice of the starting vector v and the subspace size m. The current Arnoldi (explicit/implicit) restarting techniques propose an amelioration of the initial vector v. Regarding the size of the subspace, it is known that the eigen-information of interest may not appear when m is too small [96]. Furthermore, if m is too large, the computation cost of orthogonalization process becomes excessive. The size of the subspace has to be chosen as a compromise between these factors and is chosen empirically according to the number of desired eigen-elements, the size of

the original problem, etc. Here, we present a way to increase the quality of the Krylov subspace by improving the size of the subspace. Indeed, to remedy the essential question of the choice of the size of the subspace, this paper suggests the proliferation of these subspace sizes and to select the best one. The size of the subspace is chosen dynamically in every restarting step.

This approach consists to make use of IRAM with a set of Krylov subspaces which differ only by their size which means a set of nested subspaces. Let v be an initial vector and $M = (m_1, \cdots, m_\ell)$ be a set of ℓ subspace-sizes with $m_1 < \cdots < m_\ell$. We built ℓ Arnoldi projections on the subspaces $\mathbb{K}_{m_i,v}$ (for $i = 1, \ldots, \ell$) where $\mathbb{K}_{m_1,v} \subset \mathbb{K}_{m_2,v} \subset \ldots \subset \mathbb{K}_{m_\ell,v}$. We select then the subspace size m_{best} corresponding to the Arnoldi factorization which offers the Ritz estimates for k desired eigenpairs. The steps 2 to 4 of IRAM algorithm (i.e. algorithm 1) are applied then onto this Arnoldi factorization : $AV_{m_{best}} = V_{m_{best}} H_{m_{best}} + f_{m_{best}} e^T_{m_{best}}$. That means only this factorization among the ℓ ones will be compressed to a factorization of length k with the eigen-information of interest. This is achieved using QR steps to apply $p_{best} = m_{best} - k$ shifts implicitly. The results after the shift process and equating the first k columns on both sides are the same as in equation (4.3) with $m = m_{best}$. Beginning with this resulting k-step Arnoldi factorization, we apply then $p_i = m_i - k$ additional steps of Arnoldi factorizations to obtain ℓ new projections onto the updated subspaces (for $i = 1, \cdots, \ell$). This allows again the projection onto ℓ nested subspaces with initial guess determined by the compressed k-step Arnoldi factorization issued from the QR shifted applied to $m_{best}-$step Arnoldi factorization.

We notice that this technique allows to update the restarting vector v by taking the eigen-information obtained by several subspaces into account. The interest of the approach is that the additional information obtained by multiple subspaces allows to take advantage of the appearance of the eigen-information of interest due to the larger subspace-sizes as well as that one of the orthogonality due to the smaller subspace-sizes. Moreover, for a given restart cycle MIRAMns has almost the same time complexity as IRAM with the largest subspace size. Besides, in MIRAMns as in IRAM, the appearance of spurious eigenvalues may be avoided through complete reorthogonaliza-

tion of the Arnoldi vectors using the DGKS correction [96, 30]. An algorithm of this method to compute k ($k \leq m_1$) desired Ritz elements of A is presented by the algorithm 2.

Algorithm 2: Multiple IRAM with nested subspaces

Input : $(A, V_{m_i}, H_{m_i}, f_{m_i})$ with $AV_{m_i} = V_{m_i}H_{m_i} + f_{m_i}e_{m_i}^T$ the m_i-steps Arnoldi factorization $(i = 1, 2, \ldots, \ell)$

For $it = 1, 2, \ldots$ until convergence

1. - Compute $\sigma(H_{m_i})$ and their associated eigenvectors (for $i = 1, \ldots, \ell$)
   - Compute residual norms. If convergence in one of subspaces then stop.
2. Select the best results in these subspaces and the associated best subspace size m_{best}. Set $m = m_{best}$, $H_m = H_{m_{best}}$ and $V_m = V_{m_{best}}$, $f_m = f_{m_{best}}$.
3. Select set of $p = m - k$ shifts $(\mu_1^{(m)}, \ldots, \mu_p^{(m)})$, based upon $\sigma(H_m)$ or perhaps other information and set $q^T \leftarrow e_m^T$.
4. For $j = 1, \ldots, p$
   - Factor $[Q_j, R_j] = qr(H_m - \mu_j^{(m)} I)$;
   - $H_m \leftarrow Q_j^H H_m Q_j$; $V_m \leftarrow V_m Q_j$,
   - $q \leftarrow q^H Q_j$
5. Set $f_k \leftarrow v_{k+1}\widehat{\beta}_k + f_m q(k)$, $V_k \leftarrow V_m(1:n, 1:k)$, $H_k \leftarrow H_m(1:k, 1:k)$
6. Beginning with the k-step Arnoldi factorization $AV_k = V_k H_k + f_k e_k^T$, apply $p_i = m_i - k$ additional steps of the Arnoldi process to obtain ℓ new m_i-step Arnoldi factorization $AV_{m_i} = V_{m_i}H_{m_i} + f_{m_i}e_{m_i}^T$ (for $i = 1, \ldots, \ell$).

In order to select the best results in the step (2) of the algorithm 2 we suppose that $(V_{m_i}, H_{m_i}, f_{m_i})$ is "better" than $(V_{m_j}, H_{m_j}, f_{m_j})$ if $r_k^{m_i} < r_k^{m_j}$ where $r_k^{m_i} = max(\rho_{1,m_i}, \ldots, \rho_{k,m_i})$ is defined by Ritz estimates when `(a)` stopping criterion is used. The $r_k^{m_i}$ value is defined by the residual norm of Rayleigh quotient corresponding to $(V_{m_i}, H_{m_i}, f_{m_i})$ when `(b)` stopping criterion is used.

4.3 Numerical experiments

We have implemented and tested Algorithm 1 (IRAM) and Algorithm 2 (MIRAMns) using MATLAB to compute $k = 2$ eigenvalues of greatest magnitude, except for certain case where convergence is too fast for both IRAM and MIRAMns. The stopping criterion used in IRAM is $r_k^m = \max(\rho_1^m, \ldots, \rho_k^m) < tol$ with $\rho_i^m = |\beta_m e_m^T y_i^{(m)}| / \|A\|_F$ where tol specifies the accuracy requested. The criterion used to select the best subspace size in MIRAMns is $r_k^{m_{best}} = \min(r_k^{m_1}, \ldots, r_k^{m_\ell})$ where $m_1 < \cdots < m_\ell$ are subspace sizes (with $m_1 \geq 2 \times k$). The tolerance value tol is 10^{-8} for the figures 4.1 to 4.7, 4.12, 4.14, 4.15; 10^{-14} for the figures 4.8 to 4.10, 4.13, 4.16 to 4.17 and 10^{-12} for the figures 4.18 to 4.19. Every other stopping criterion can replace the requirement to find k eigenvalues. In all experiments presented here, initial vector is $x = z_n / \|z_n\|_2$ except for the figure 4.14 that initial vector is $x = s_n / \|s_n\|_2$ and the figures 4.16, 4.18 to 4.19 that initial vector is $x = t_n / \|t_n\|_2$. The initial vectors of IRAM and MIRAMns are the same one. The efficiency of these algorithms can thus be measured in terms of the number it of restarts (iterations) or the number of matrix-vector products $M.V.P$. Our matrices are presented in the table 4.1.

Matrix	Size of matrix	nonzero elements
*af*23560.*mtx*	23560	484256
*bfw*782*a*.*mtx*	782	7514
*A*9_1000.*mtx*	1000	2998
*west*0989.*mtx*	989	3537
*AM*_1000.*mtx*	1000	2998
*sherman*3.*mtx*	5005	20033
roadNet-PA.mtx	1088092	3083796
com-Youtube.mtx	1134890	2988374
WikiTalk.mtx	2394385	5046614

TABLE 4.1: General information about the test matrices

We have used four matrices *af*23560.*mtx*, *bfw*782*a*.*mtx*, *west*0989.*mtx* and
*sherman*3.*mtx* from Matrix Market [8]. *A*9_1000 is a tridiagonal matrix of

order 1000 defined by $a_{i,i} = 3$, $a_{i,i+1} = a_{i,i-1} = 1$. All other entries are zero. The tridiagonal matrix AM_1000 is of dimension $n = 1000$. The diagonal entries are $a_{i,i} = i$, the codiagonal entries are $a_{i,i+1} = -0.1$ and $a_{i,i-1} = 0.1$. All other entries are zero. This example has been taken from [74]. Matrices *roadNet-PA.mtx, com-Youtube.mtx* and *WikiTalk.mtx* are transition matrices constructed from three social graphs by using Markov chain. These graphs could be found in [64].

4.3.1 MIRAMns versus IRAM

In all the following figures MIRAMns$(m_1, \cdots, m_\ell)$ denotes an MIRAMns with subspace sizes $(m_1, \cdots, m_\ell)$, IRAM(m) denotes an IRAM with subspace size m and $M.V.P$ denotes the number of matrix-vector products. It is important to note that the **main objective** of our experiments is to compare the performance of MIRAMns$(m_1, \cdots, m_\ell)$ and IRAM(m_ℓ). However, some of experiments have the aim of highlighting the influence of certain parameters on the convergence of these methods. For some typical cases, we present the best subspaces choosen by MIRAMns throughout the restarting cycles, so as to clarify the necessity of using the whole set of subspaces.

The table 4.2 presents the results obtained with IRAM algorithm and the table 4.3 presents the results obtained with MIRAMns and a comparison between IRAM and MIRAMns in term of number of matrix-vector products (MVP), execution time in seconds ($Ex.Time$) and number of restarts (it). $Ex.Time$ represents the total execution time : from the beginning of of the algorithm (after inputs) upto obtaining the wanted eigenpairs. We can see that in almost half of the experiments presented in table 4.2, IRAM does not converge. The results presented in the table 4.3, show that MIRAMns overcomes these problems of non-convergence.

We show graphically in Figures 4.1 to 4.19 the norm of residual as a function of iteration number to reach convergence using MIRAMns and IRAM. We see that there is no convergence for IRAM in figures 4.1 to 4.4, 4.7 to 4.10, 4.16 and 4.18 to 4.19 while MIRAMns reaches convergence. Moreover, the convergence of MIRAMns in figures 4.5, 4.12 to 4.15 and 4.17 is better

than IRAM. Specifically, Figure 4.1 shows that the curves of convergence of IRAM and MIRAMns undergo oscillations around the residual norm 10^{-6}. However, the peak to peak amplitude of the oscillations corresponding to IRAM is very large while the one corresponding to MIRAMns are quite small. This could become related to a kind of smoothing of the curve of convergence of IRAM by MIRAMns.

Figure 4.2 shows the influence of the size of the subspace on the convergence of IRAM. Indeed, we note that an augmentation of this size relative to that of Figure 4.1 smooths the curve of the convergence of IRAM. We can see also that with the chosen tolerance ($tol = 10^{-8}$), MIRAMns converges but IRAM does not converge. However, with a greater tolerance value such as $tol = 10^{-7}$, IRAM reaches convergence when MIRAMns does not reach it. But it must be remembered that in this case, the parameters of IRAM and MIRAMns no longer meet the criteria for comparison. Indeed, the subspace size of IRAM (22) is larger than $m_l = 10$.

Figure 4.5 shows the effect of increasing the size of the subspace on the convergence of IRAM and MIRAMns and highlights the speed of convergence of MIRAMns with respect to that of IRAM. The acceleration of convergence of IRAM by MIRAMns is also shown in Figures 4.7 to 4.19. However, we can see that in Figure 4.7, before $tol = 10^{-5}$, IRAM could reach convergence faster than MIRAMns. But this is just an oscillation peak stronger than the others and the residual norms do not decrease continually while those of MIRAMns decreases steadily during the iterations. Figures 4.7, 4.8 and 4.10 show further how the convergence curves of IRAM are "smoothed" by MIRAMns. Furthermore, by comparing figure 4.3 with 4.4 and figure 4.9 with figure 4.10, we notice that an increase in number of subspaces could improve the speed of convergence for MIRAMns itself as well.

To check the influence of the strategy proposed in [97], we compared the tick restarted versions of IRAM and MIRAMns. The Figures 4.6 and 4.11 show this comparison with the number of wanted eigenpairs $k = 10$ et the thick parameters $k + 2$ and $k + 5$ which means in each restart cycle, a buffer of 2 and 5 extra vectors are kept. We can notice that IRAM could sometimes perform as good as MIRAMns.

Figures 4.20 to 4.22 show the subspaces selected by MIRAMns throu-

ghout iterations. We see that m_{best} is choosen randomly from interval $[m_1, m_l]$ and for some experiments m_1 is selected as best subspace size more often that m_l (figure 5.22). This phenomenon of local optimality of small subspaces was also observed by Embree on GMRES [36].

Figures 4.23 and 4.24 show the execution time of MIRAMns versus IRAM throughout iterations for *AM*_1000 and *west*0989 matrices. We notice that the execution times of each restarting cycle is almost the same for both methods. Nevertheless, the total execution time of MIRAM is much smaller than IRAM.

Tables 4.2 and 4.3 and Figures 4.1 to 4.19 indicate that our algorithm improves performances of IRAM. We notice also that this improvement is much more significant when the matrices have clustered eigenvalues such as *af*23560 and *bfw*782 matrices used in our experiments.

Matrix	m	x	*tol*	it	*MVP*	Ex.Time	Res.Norm	Fig.
*af*23560.*mtx*	10	z_n	10^{-8}	* 500	*4002	10.27	no conv.	4.1
*af*23560.*mtx*	22	s_n	10^{-8}	* 500	*10002	31.44	no conv.	4.2, 4.3, 4.4
*af*23560.*mtx*	25	z_n	10^{-8}	6	140	0.60	2.60e-09	4.5
*af*23560.*mtx*	32	z_n	10^{-14}	6	182	0.13	2.68e-16	4.6
*bfw*782*a.mtx*	10	z_n	10^{-8}	* 500	*4002	1.00	no conv.	4.7
*bfw*782*a.mtx*	20	z_n	10^{-14}	* 500	*9002	2.18	no conv.	4.8, 4.9, 4.10
*bfw*782*a.mtx*	30	z_n	10^{-14}	12	338	0.01	1.98e-15	4.11
*A*9_1000.*mtx*	20	z_n	10^{-8}	309	5564	1.40	9.57e-09	4.12
*west*0989.*mtx*	10	z_n	10^{-8}	53	426	0.11	2.75e-09	4.13
*AM*_1000.*mtx*	20	s_n	10^{-8}	22	398	0.12	8.93e-09	4.14
*sherman*3.*mtx*	10	z_n	10^{-8}	4	34	0.04	5.40e-13	4.15
roadNet-PA.mtx	20	t_n	10^{-14}	* 500	*9002	1680.24	no conv.	4.16
com-Youtube.mtx	20	s_n	10^{-14}	30	482	113.20	3.03e-15	4.17
WikiTalk.mtx	20	z_n	10^{-12}	* 500	*9002	4487.15	no converge	4.18, 4.19

TABLE 4.2: *IRAM performances*

			IRAM			MIRAMns			
Matrix	Fig.	ℓ	m	it	MVP	$m_1, \cdots, m_\ell$	it	MVP	Ex.Time
af23560	4.1	3	10	* 500	* 4002	5, 7, 10	272	2178	7.84
	4.2	3	22	* 500	* 10002	5, 7, 10	250	2002	7.88
	4.3	3	22	* 500	* 10002	16, 19, 22	31	622	2.68
	4.4	5	22	* 500	* 10002	10, 13, 16, 19, 22	8	162	0.82
	4.5	3	25	6	140	5, 10, 25	6	140	0.63
	4.6	6	32	6	182	16, 20, 24, 28, 32	7	212	0.30
bfw782a	4.7	3	10	* 500	* 4002	5, 8, 10	70	562	0.13
	4.8	3	20	* 500	* 9002	5, 10, 20	104	1874	0.32
	4.9	3	20	* 500	* 9002	8, 18, 20	92	1658	0.38
	4.10	3	20	* 500	* 9002	8, 11, 14, 17, 20	32	578	0.16
	4.11	3	30	12	338	20, 25, 30	14	394	0.01
A9_1000	4.12	3	20	309	5564	10, 15, 20	94	1694	0.36
west0989	4.13	3	30	53	426	5, 8, 10	32	258	0.07
AM_1000	4.14	3	20	22	398	13, 17, 20	17	308	0.09
sherman3	4.15	3	10	4	34	5, 8 , 10	2	18	0.02
roadNet-PA	4.16	6	20	* 500	*9002	5, 8, 11, 14, 17, 20	157	2828	599.74
com-Youtube ($k = 4$)	4.17	6	20	30	482	5, 8, 11, 14, 17, 20	20	332	108.08
WikiTalk ($k = 2$)	4.18	6	20	* 500	*9002	4, 7, 10, 13, 16, 20	312	5618	3438.66
WikiTalk ($k = 4$)	4.19	6	20	* 500	*9002	5, 8, 11, 14, 17, 20	15	272	181.48

TABLE 4.3: Comparison of IRAM(m) and MIRAMns($m_1, \cdots, m_\ell$)

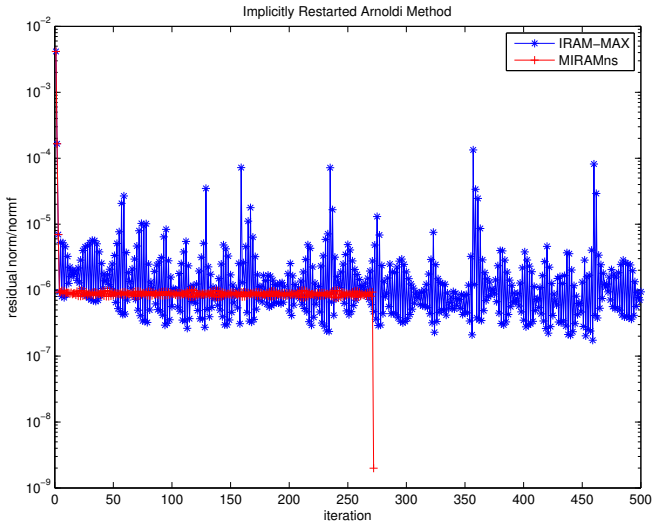


FIGURE 4.1: MIRAMns$(5, 7, 10)$ versus IRAM(10) with $af23560$ matrix

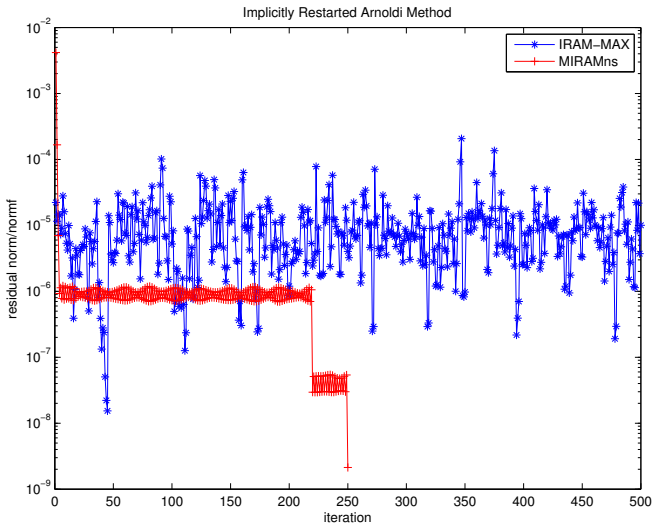


FIGURE 4.2: MIRAMns$(5, 7, 10)$ versus IRAM(22) with $af23560$ matrix

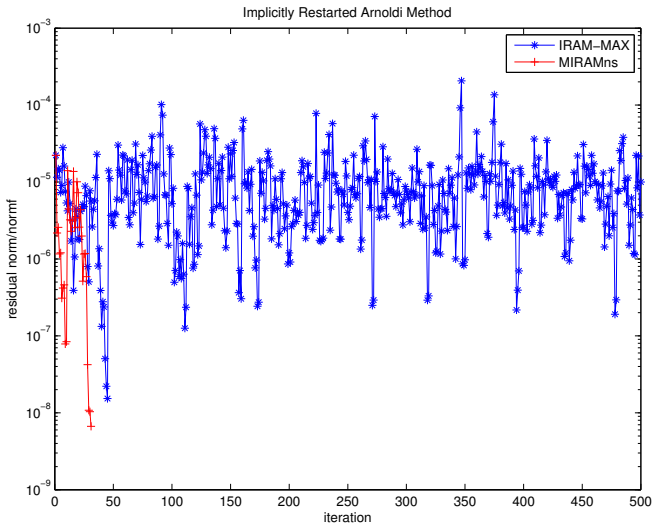


FIGURE 4.3: MIRAMns$(16, 19, 22)$ versus IRAM(22) with $af23560$ matrix

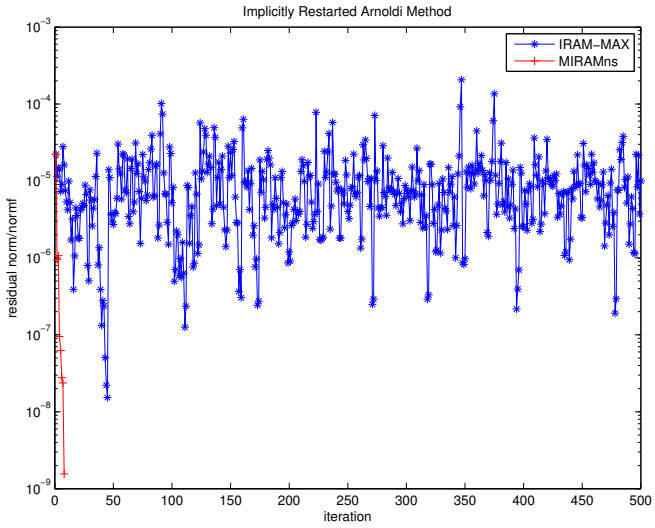


FIGURE 4.4: MIRAMns$(10, 13, 16, 19, 22)$ versus IRAM(22) with $af23560$ matrix

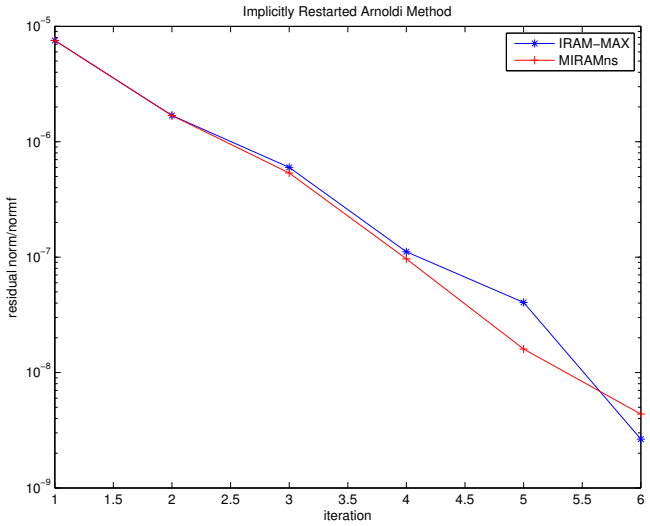


FIGURE 4.5: MIRAMns$(5, 10, 25)$ versus IRAM(25) with $af23560$ matrix

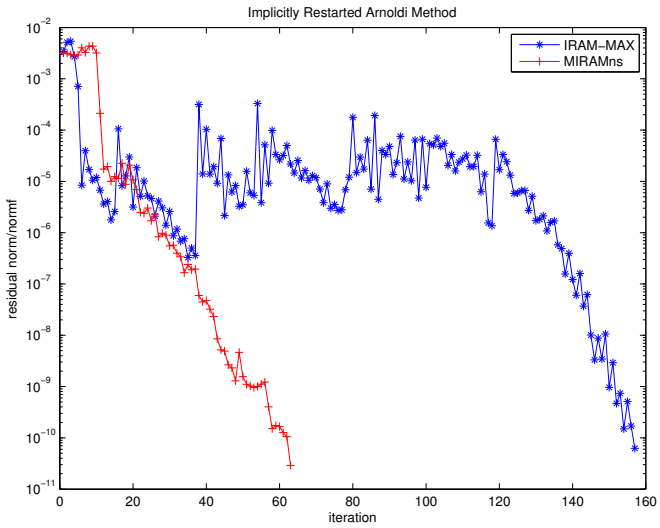


FIGURE 4.6: MIRAMns$(20, 30, 40)$ versus IRAM(40) with $af23560$ matrix, k=10 with a buffer of 2 extra vectors

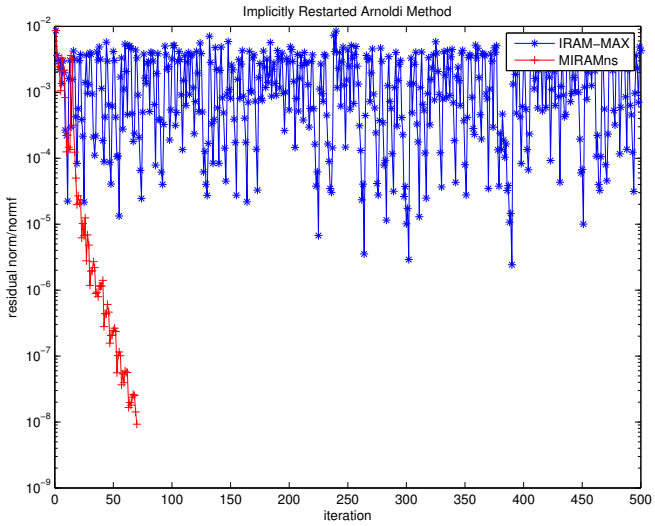


FIGURE 4.7: MIRAMns$(5, 8, 10)$ versus IRAM(10) with $bfw782a$ matrix

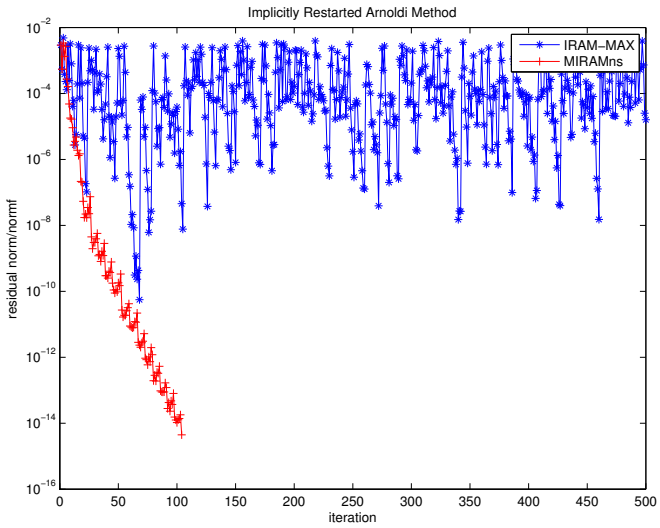


FIGURE 4.8: MIRAMns$(5, 10, 20)$ versus IRAM(20) with $bfw782a$ matrix

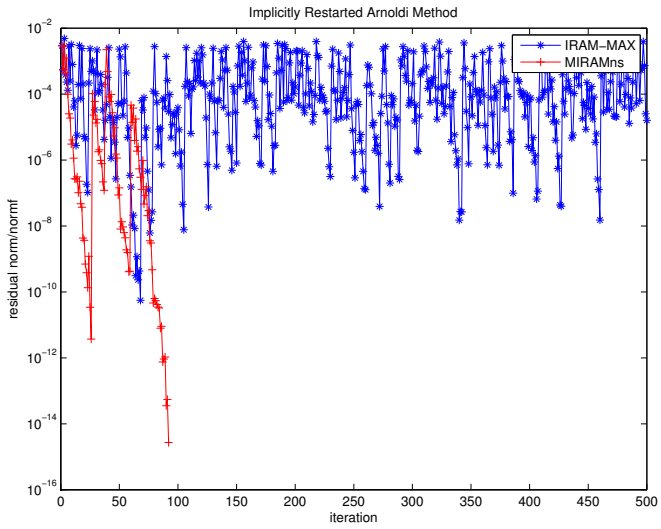


FIGURE 4.9: MIRAMns$(8, 18, 20)$ versus IRAM(20) with $bfw782a$ matrix

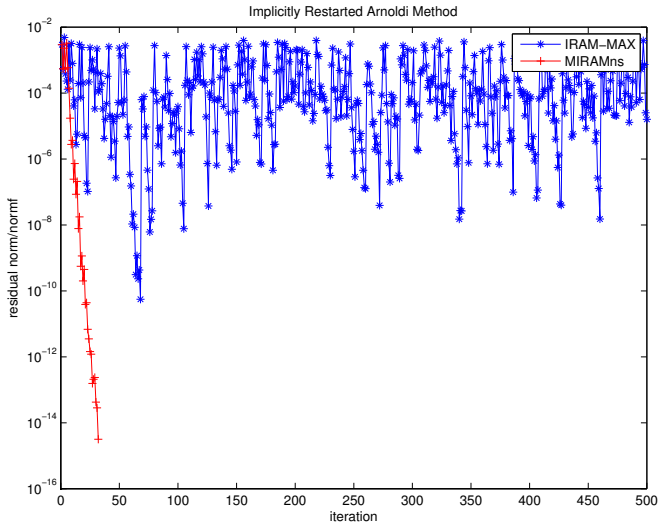


FIGURE 4.10: MIRAMns$(8, 11, 14, 17, 20)$ versus IRAM(20) with $bfw782a$ matrix

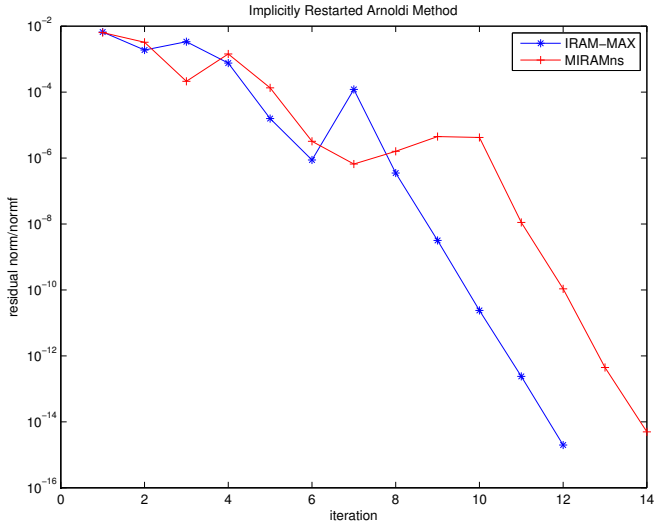


FIGURE 4.11: MIRAMns$(20, 25, 30)$ versus IRAM(30) with $bfw782a$ matrix, k=10 with a buffer of 5 vectors

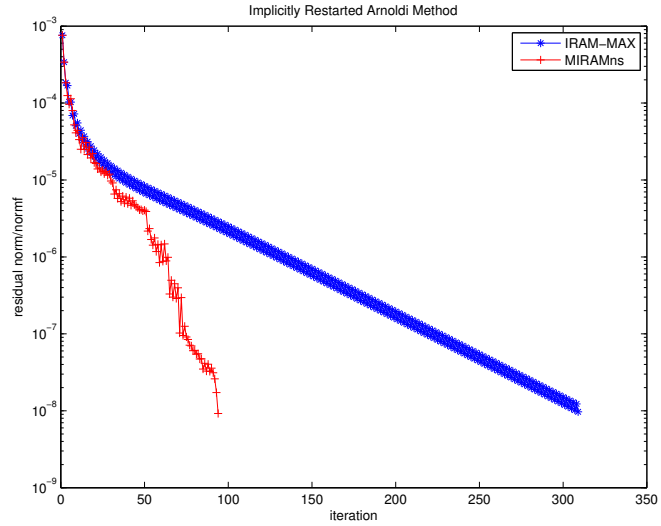


FIGURE 4.12: matrix MIRAMns$(10, 15, 20)$ versus IRAM(20) with $A9\_1000$ matrix

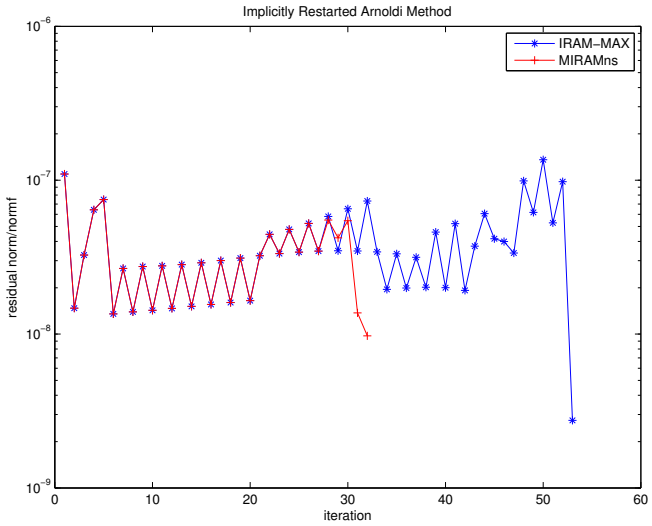


FIGURE 4.13: MIRAMns$(5, 8, 10)$ versus IRAM(10) with *west*0989 matrix

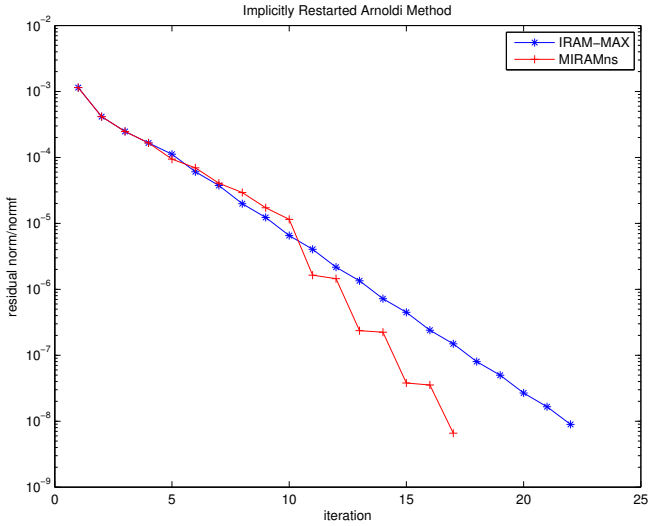


FIGURE 4.14: MIRAMns$(13, 17, 20)$ versus IRAM(20) with AM_1000 matrix

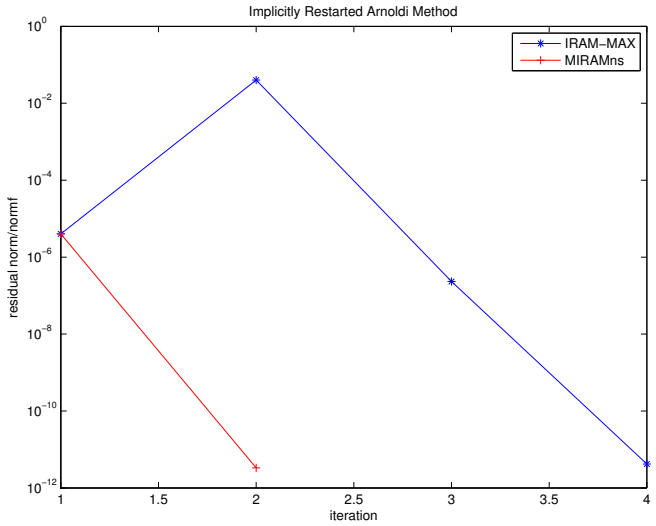


FIGURE 4.15: MIRAMns$(5, 8, 10)$ versus IRAM(10) with *sherman*3 matrix

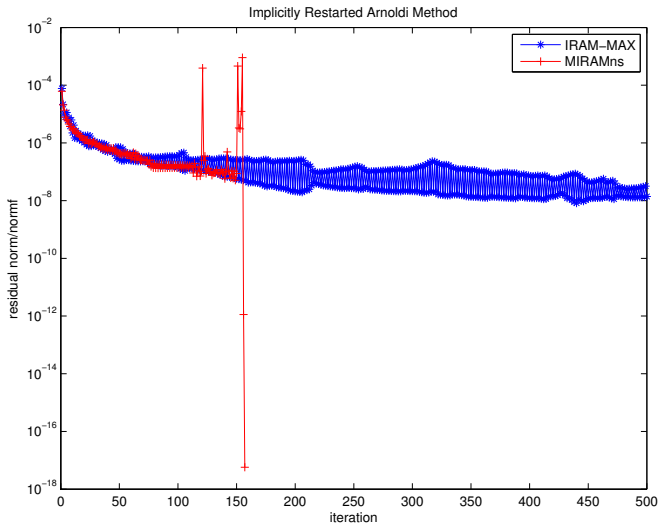


FIGURE 4.16: MIRAMns$(5, 8, 11, 14, 17, 20)$ versus IRAM(20) with roadNet-PA matrix

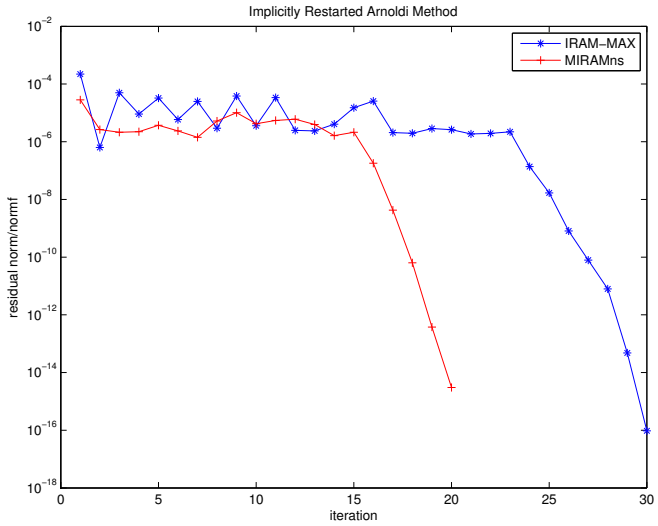


FIGURE 4.17: MIRAMns$(5, 8, 11, 14, 17, 20)$ versus IRAM(20) with com-Youtube matrix $(k = 4)$

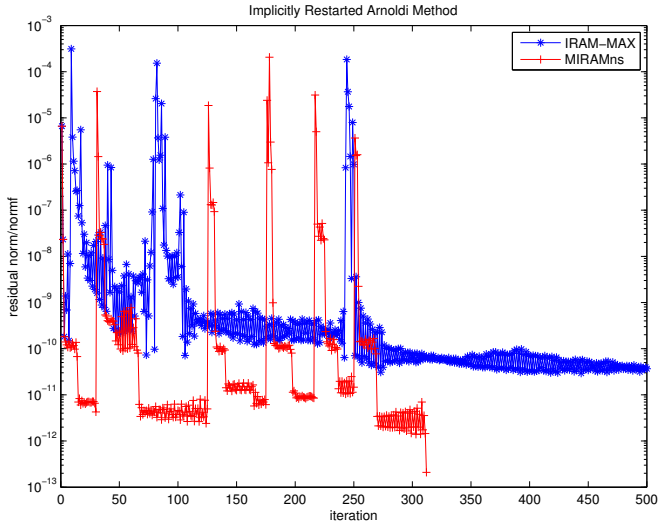


FIGURE 4.18: MIRAMns$(4, 7, 10, 13, 16, 20)$ versus IRAM(20) with *WikiTalk* matrix $(k = 2)$

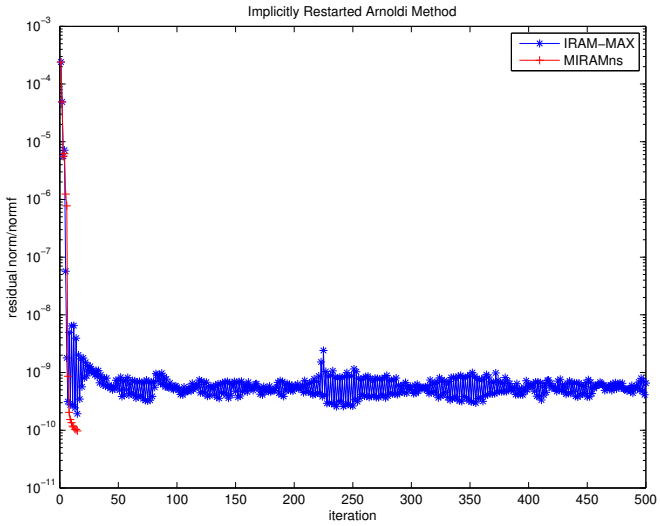


FIGURE 4.19: MIRAMns$(5, 8, 11, 14, 17, 20)$ versus IRAM(20) with *WikiTalk* matrix $(k = 4)$, $tol = 10^{-10}$

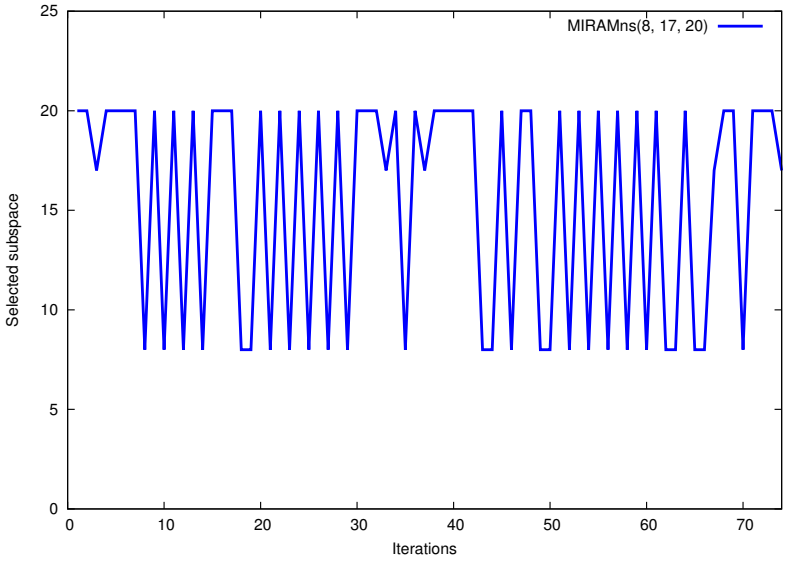


FIGURE 4.20: Evolution of m_{best} in MIRAMns$(8, 17, 20)$ among iterations with *bfw*782*a* matrix

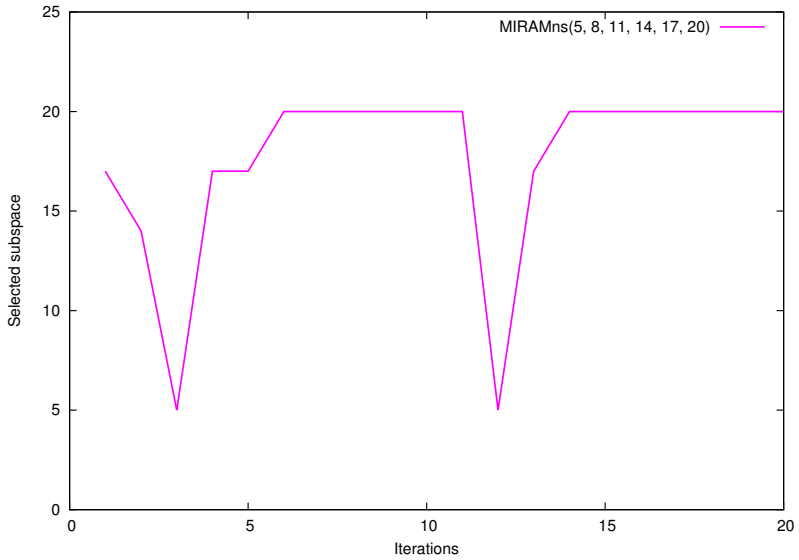


FIGURE 4.21: Evolution of m_{best} in MIRAMns$(5, 8, 11, 14, 17, 20)$ among iterations with com-Youtube matrix

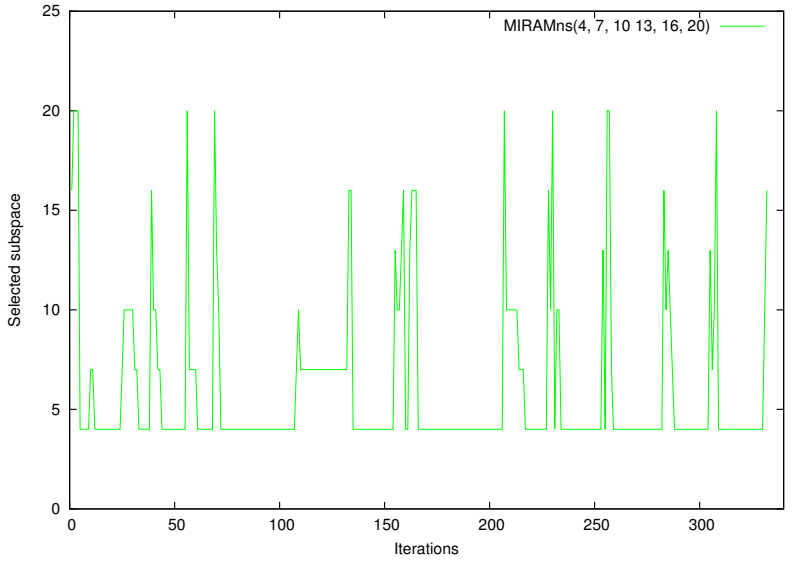


FIGURE 4.22: Evolution of m_{best} in MIRAMns$(4, 7, 10, 13, 16, 20)$ among iterations with roadNet-PA matrix

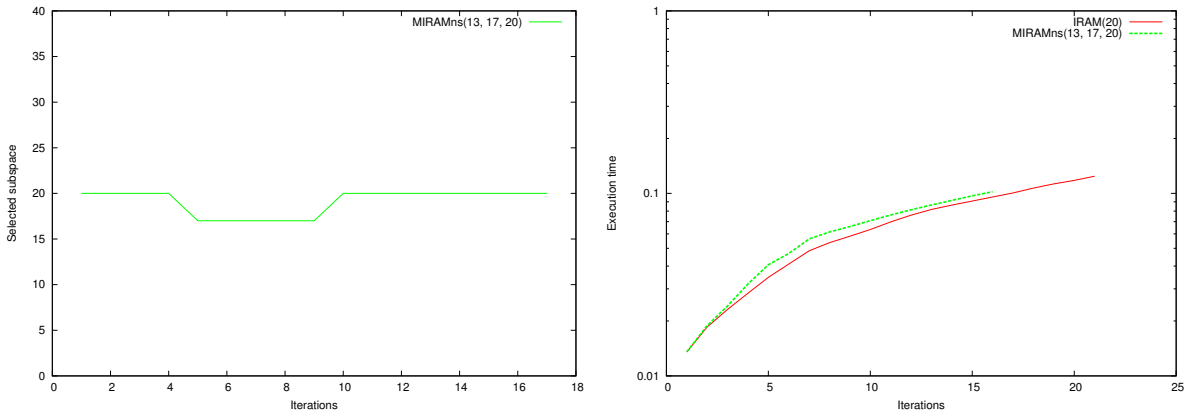


FIGURE 4.23: Execution time of MIRAMns(13,17,20) versus IRAM(20) for AM_1000 matrix, tol=10^{-8}

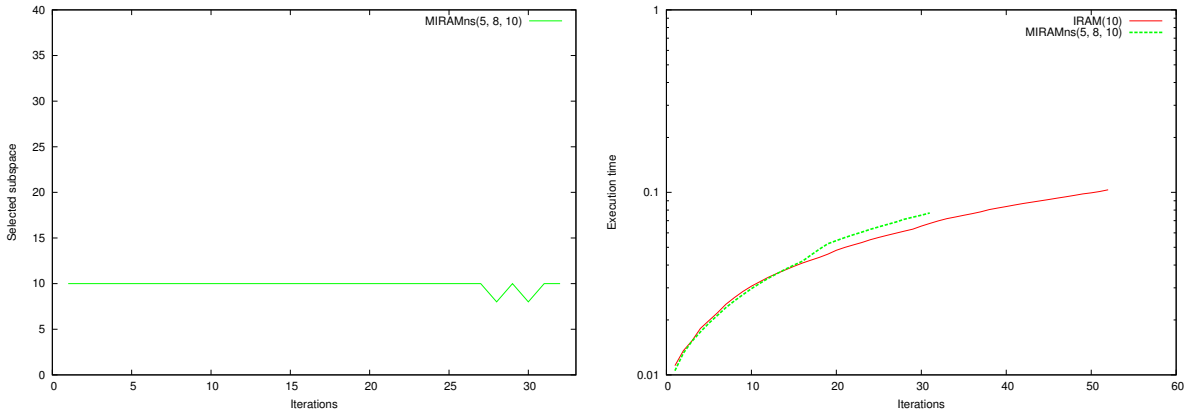


FIGURE 4.24: Execution time of MIRAMns(5,8,10) versus IRAM(10) for *west*0989 matrix

4.3.2 Conclusion

Due to the empirical choice of subspace size, the implicitly restarted Arnoldi method may not be efficient for computing a few selected eigenpairs of large sparse non-Hermitian matrices. In order to improve this choice, we have proposed to make use of this method with several Krylov subspaces. We have seen that the multiple implicitly restarted Arnoldi method with nested subspaces accelerates the convergence of IRAM with the same number of matrix-vector products in each iteration. Our numerical experiments have shown that MIRAM improves the quality of the Krylov subspaces of IRAM and has consequently better convergence properties. Moreover, the strategy presented in the paper can be applied to many other restarted projection methods. In a general context, this is equivalent to coupling some iterative methods in order to accelerate the convergence of one of them as is the case of hybrid Arnoldi-Chebyshev method described in [41, 91]. We mentioned that for MIRAM we can use the same parallel programming model as the one used in P_ARPACK [70] which implements the parallel implicitly restarted Arnoldi method. This is because we define a Krylov subspace and make use of the eigen-information of some subspaces nested in it. Another approach consists to make use of IRAM with several Krylov subspaces which differ by both their initial vector and subspace size. This approach, multiple IRAM, has the advantage to update (implicitly) the initial vector of an IRAM by taking into account the eigen-information obtained by several different (non nested) subspaces. The increase cost engendered by these different subspaces could be compensated by the implementation of the method in a large-scale distributed environment as for the multiple explicitly Arnoldi method in [35].

Chapitre 5

A parallel MIRAM algorithm for PageRank computation

5.1 Introduction

In order to simulate the epidemic spread, such as H1N1 outbreak in France, traditional models need hundreds of experiments and compute the expected outcome by averaging. In addition, these experiments should be adjusted on a daily basis during the initial outbreak.

To answer urgent requests during the beginning phase of outbreak, an eigenvalue model is proposed in [66]. In this model, a PageRank-like Infection vector is calculated, which could help health officials decide the relative importance of different agents or groups of agents in a population facing an epidemic. Concerning the computational aspect, the difficulty for computing PageRank arises from the size of network and the big damping factor. Due to similar characteristics, this problem is also encountered in other real applications. In this chapter, we study the computation of PageRank within this context.

PageRank citation ranking was initially introduced in [82] to bring order to the Web. A page has high rank if the sum of the ranks of its inlinks is high. In other words, rank is propagated through links. To use mathematical formalism, we look for a PageRank vector x, which is the dominant eigenvector of the *Google matrix*,

$$A = \alpha P + (1 - \alpha)ve^T,\ 0 \leq \alpha < 1 \tag{5.1}$$

where the matrix P is a column stochastic matrix, called *transition matrix*, representing the outlink structure of the Web, e is the vector $(1, ..., 1)^T$, α is called the damping factor, and the vector v is the teleportation vector, which ensures the uniqueness of the PageRank vector. Noticing that the virus has a small probability $(1 - \alpha)$ to jump from any individual to any other individual in a social graph. This would happen, for example, when an infected person (virus carrier) meets and passes the disease to someone outside his normal contacts . This event happens rarely so that the damping factor α is very close to 1 in application on epidemics. A difficulty in PageRank model is caused by the existence of dangling nodes [23]. These nodes will result in one or more columns of zeros in transition matrix P. Several ideas have been proposed to deal with this problem. A good reference can be found in section 8.4 from the Langville and Meyer book on PageRank [59]. We will continue our discussion about this issue within epidemic application in Section 3.2.1.

Many algorithms have been proposed for computing PageRank [14]. In this paper, we focus on Arnoldi-type algorithms. The method proposed by Golub and Greif combines Arnoldi process and singular value decomposition to compute PageRank [47]. Wu and Wei use an extrapolation procedure to provide increasingly better initial guess to Arnoldi iteration [105]. Their idea is to periodically subtract off estimates of the non-principal eigenvectors. Authors of [46] demonstrated the fast convergence of Krylov subspace methods for the PageRank linear systems. A comparison of the eigenproblem viewpoint and the linear system viewpoint over the PageRank problem can be found in [106]. The idea to use GMRES method for PageRank is further explored in [107].

In real applications, computation of PageRank has three challenging aspects. First, the matrices involved are very large and rely on a parallel sparse matrix-vector product (MVP) kernel. Suppose z is a vector of p-norm 1, Az can be written as $\alpha Pz + (1 - \alpha)v(e^T z)$ where $e^T z$ is a scalar. So the MVP of A is expressed as MVP of a sparse matrix P plus a vector. Otherwise, any direct computation using A will be bottlenecked by memory requirement for large networks. In fact, the Google matrix A becomes a dense matrix due to the part $(1 - \alpha)ve^T$. For the above reason, algorithms based on MVP

might be advantageous. Secondly, the damping factor α generally needs to take values approaching 1. For example, in the model of epidemic spread, the virus has the probability $1 - \alpha$ to jump randomly from an infected individual to any other individual through some unusual contact. Intuitively, this event rarely happens, and for disease spread, α must be very close to 1. This is an argument in favor of using Arnoldi-type methods, as opposed to the Power method. In fact, it can be proved that the second largest eigenvalue of matrix A is very close to α [50]. For big α, the second largest eigenvalue will be close to the dominant eigenvalue (that equals to 1) of A, which will slow down the convergence of the Power method. Last but not least, the network is very large and of scale-free structure. Vectors used in the computation should be stored in parallel among p processors, because they could be larger than any single processor could handle. For example, take $n = 10^9$ for a network, the corresponding PageRank vector contains 10^9 entries, which could take as much as $16 * 10^9$ bytes $\approx$ 15 GB of memory in complex double precision. This issue of storage requirement is worsened when using Krylov subspace methods. For instance, we can consider the parameters $n = 10^{11}$ and $m = 10^3$, where n is the size of the problem and m is the projection subspace size. Then, each iteration requires 10 Peta bytes memory space to maintain the orthogonal basis.

The model of parallelization used is so general that it could be employed for modern (possibly future) parallel architecture. According to our numerical results, we inspect that : the strategies proposed could accelerate the convergence of single IRAM for matrices derived from real applications.

5.2 High performance systems evolution

A parallel computer is a collection of processing elements that communicate and cooperate to solve large problems fast [6]. This definition can be applied to most nowadays computers, desktops with multiple cores or hyper threaded processors, workstations with several processors, and high performance systems dedicated to scientific problems. The need for computing resources increases jointly with the order of scientific problems. Results obtained open new research areas which lead to more complex simulation

models. The more accurate the results become the more experiments and simulations are needed. The amount of computation required increases faster than the computing capacities. To support the demand of computation capacity, high performance systems evolve at a fearsome speed. They will reach the capacity of executing an exa floating point operations per second in the near future. In a little more than ten years, high performance computers will have multiplied their capacities a thousand times. This performance is possible thanks to the evolution of processors, networks, operating systems and programming environments. In order to study the evolution of parallel computers, it is not enough to focus on the hardware. A high performance computer is always associated to multiple software layers involved in the management of the global architecture. In this chapter, we focus on a presentation of the evolution of the hardware, parallel programming models and their realization for high performance computers. In the mean time, the networking infrastructure supporting Internet evolved too in order to support massive and numerous data transfers, multimedia applications, games and more generally interactive contents. The number of computers connected to Internet has increased significantly too. The amount of potential resources connected to Internet is large enough to justify research for a new kind of high performance systems based on computers connected to each other using wide area networks (WAN). They are known as meta-computing systems. Several approaches exist in order to aggregate computing resources distributed in multiple locations. These approaches are known as grid, global computing and peer to peer systems. In the mean time, Internet evolved into a set of services which collaborate to provide more complex treatment to end users.

Recently, the evolution of processors took a new direction. Indeed, the increase of the frequency is no more a possible solution to gain performance. However, the Moore law [73], which specifies that the number of transistors composing a processor doubles every eighteen months, still applies. Processor architects introduced the concept of cores. A processor is now composed of several cores following the MIMD model. Processors composed of multiple cores are already available on high performance systems, workstations, desktop computers and laptops. The number of threads or control flows

which will be executed concurrently on next generation high performance systems will be of a few millions. The issue related to the handling of large scale distributed systems are the same in the context of meta computing systems and future high performance systems. They will need to provide mechanism to handle heterogeneity, fault tolerance, scheduling, etc. In this chapter, we present the evolution of high performance systems by means of architecture changes, parallel programming models and tools to use them.

5.2.1 Classification of high performance systems

According to the taxonomy of Flynn[43], there are four main architectural classes, which is based on the way of manipulating instructions and data streams :

SISD machines : A sequential computer which exploits no parallelism in either the instruction or data streams. Single control unit (CU) fetches single Instruction Stream (IS) from memory. The CU then generates appropriate control signals to direct single processing element (PE) to operate on single Data Stream (DS) i.e. one operation at a time. Examples of SISD architecture are the traditional uniprocessor machines like a PC (currently manufactured PCs have multiple processors).

SIMD machines : A computer which exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized. For example, an array processor or GPU.

MISD machines : Multiple instructions operate on a single data stream. Uncommon architecture which is generally used for fault tolerance. Heterogeneous systems operate on the same data stream and must agree on the result. Examples include the Space Shuttle flight control computer.

MIMD machines : Multiple autonomous processors simultaneously execute different instructions on different data. Distributed systems are generally recognized to be MIMD architectures ; either exploiting a single

shared memory space or a distributed memory space. A multi-core superscalar processor is MIMD processor.

The Flynn taxonomy is not enough to classify architectures for high performance systems. Almost all nowadays HPC systems fall in the MIMD class of machines. However, the classification of Flynn can be refined for MIMD systems based on the memory model used.

5.2.2 Shared memory systems

As shown in Figure 5.1, shared memory systems have multiple CPUs all of which share the same address space. This means that the knowledge of where data is stored is of no concern to the user as there is only one memory accessed by all CPUs on an equal basis. In shared memory systems, the communication between processors occurred through memory accesses. The memory controller is often more complex than in other systems. It is responsible for retrieving memory area accessed for reading and writing and to maintain the consistency of local caches. Shared memory systems used to depend on a single memory area which was accessible from all processors of the systems. In this kind of architecture, a bus or a network connects processors to memory. The main problem with the shared-memory system as described above is that it is not scalable to large numbers of processors. Most bus-based systems are limited to 32 or fewer processors because of contention on the bus. If the bus is replaced by a crossbar switch, systems can scale to as many as 128 processors, although the cost of the switch increases as the square of the number of processors, making this organization impractical for truly large numbers of processors. Multistage switches can be made to scale better at the cost of longer latencies to memory. The scalability of high performance systems such as the IBM SP series or the SGI Origine 2000, uses a different approach to allow scalability. Non uniform memory access (NUMA) architectures were introduced to bring more scalability to systems providing shared memory. NUMA systems distribute the memory to each processor. In such systems, the network of interconnexion

is still used for all memory operations (read/write). However, the round trip time to retrieve a memory area is not fixed and varies depending on the distance between the two processors involved.

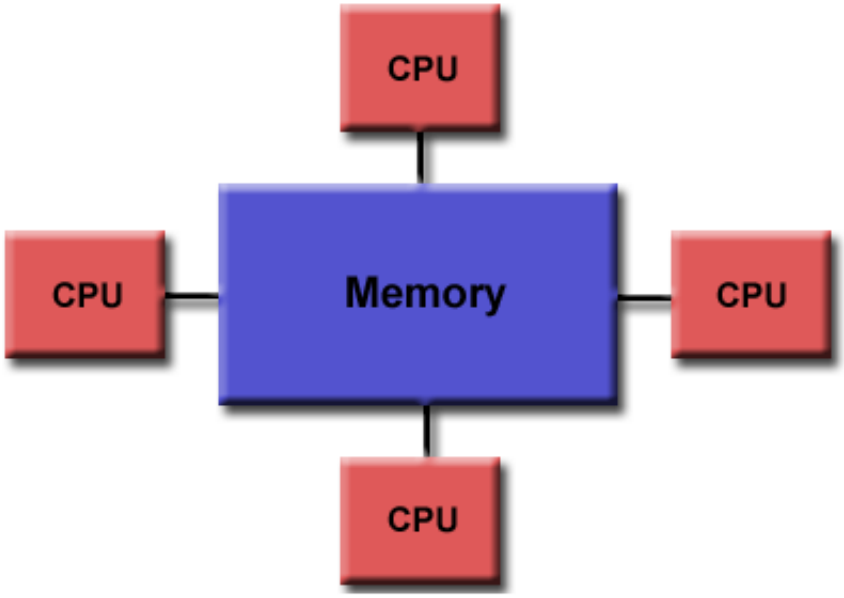


FIGURE 5.1: A shared memory architecture

5.2.3 Distributed memory systems

In distributed memory systems (see Figure 5.2) each CPU has its own associated memory. The CPUs are connected by some network and may exchange data between their respective memories when required. In contrast to shared memory machines the user must be aware of the location of the data in the local memories and will have to move or distribute these data explicitly when needed. Distributed-memory MIMD systems exhibit a large variety in the topology of their connecting network. The details of this topology are largely hidden from the user which is quite helpful with respect to portability of applications. The main difference between a NUMA shared and a distributed memory systems lies in the integration with the network of interconnection. In shared memory systems, the network interacts with the memory controller. In distributed systems, the network of interconnection is connected to processors instead. This approach is often preferred over the integration with the memory controller. Example of ar-

chitectures which are based on this memory model include the CRAY T3E, Fujistsu AP3000 and networks of workstations. The aspect which evolves the most in these kind of systems is the network of interconnection. Many topology have been evaluated. This kind of architecture is really similar to clusters where nodes are workstations built upon widely available processors and networking solutions. A cluster is constituted by a collection of nodes connected to each other by a network of interconnection. In clusters each node provides persistent storage, one or several CPUs, local memory and runs its own operating system. In order to improve performance, enhanced networking solution can be used instead of Ethernet. The principal programming problem for distributed-memory systems is management of communication between processors. Usually this means consolidation of messages between the same pair of processors and overlapping communication and computation so that long latencies are hidden. In addition, data placement is important so that as few data references as possible require communication.

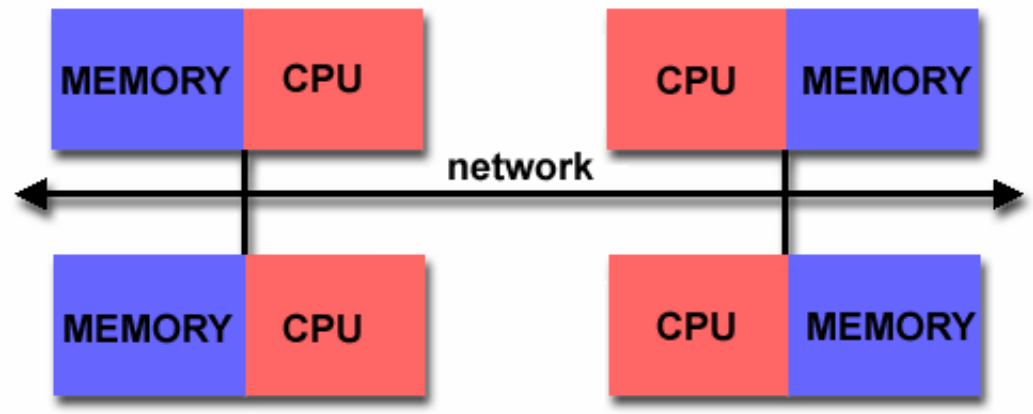


FIGURE 5.2: A distributed memory architecture

5.2.4 Hybrid systems

This is the architecture where the two previous paradigms are combined (see Figure 5.3). Some distributed-memory machines allow a processor to directly access a datum in a remote memory. On these distributed shared-memory (DSM) systems, the latency associated with a load varies with the distance to the remote memory. Cache coherency on DSM systems is a com-

plex problem that is usually handled by a sophisticated network interface unit. Given that DSM systems have longer access times to remote memory, data placement is an important programming consideration. For very large parallel systems, a hybrid architecture called an SMP cluster is common. An SMP cluster looks like a distributed-memory system in which each of the individual components is a symmetric multiprocessor rather than a single processor node. This design permits high parallel efficiency within a multiprocessor node, while permitting systems to scale to hundreds or even thousands of processors. Programming for SMP clusters provides all the challenges of both shared- and distributed-memory systems. In addition, it requires careful thought about how to partition the parallelism within and between computational nodes.

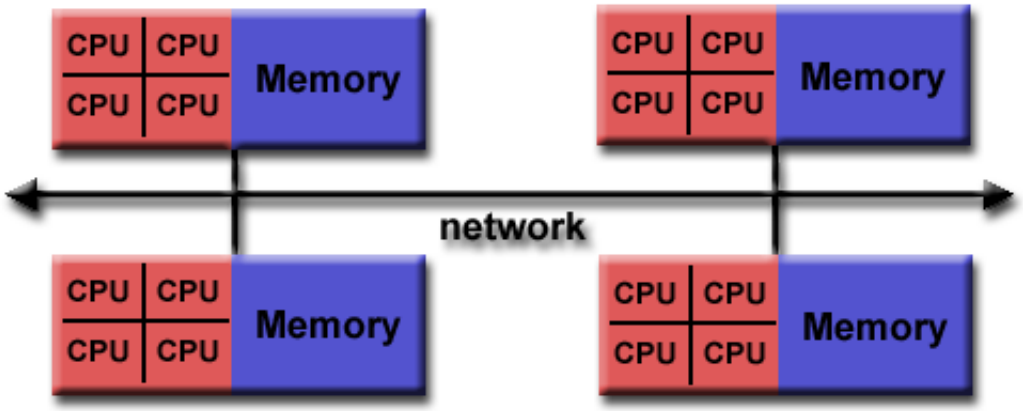


FIGURE 5.3: A distributed memory architecture

5.3 Programming models

An important part in forming a parallel programs is which programming model to use. This decision will affect the choice of programming language system and library for implementation of the application.

5.3.1 Data parallelism

Data parallelism consists in defining a single control flow which is common to multiple data. In this model the programmer is only able to explicit

the data distribution. Other aspects of the parallelism are expressed implicitly. It corresponds to the SIMD execution model. The same instructions are executed synchronously by all processing elements. Data parallel languages allow the definition of a virtual topology of processing elements. This topology is mapped on the read processing units available on the system. Processing elements can communicate only with nearest neighbors. The underlying topology was most often a two dimensional grid. This topology is well adapted to operations on regular data sets such as matrices and images. Connexion Machines, like the CM-5, were especially design to support data parallel algorithms and applications. Vectorization is a adaptation of the data parallel model. Nowadays vector instructions are available in most general purpose processors. A data parallel application first defines a virtual topology of processors such as a 2D grid. This topology is then mapped on real processors during the execution of the program. The program defines a single flow of execution which is executed by all virtual processors concurrently. Instruction of the program consists in traditional instructions and in communication operations with neighbours processors in the virtual topology. Many linear algebra applications can be efficiently implemented using this model [34]. A more detailed description of concepts, tools and languages associated with this model can be found in [85].

5.3.2 Task parallelism

Task parallelism consists in defining an application as a collection of tasks. Control flow units collaborate using either explicit or implicit communications/synchronizations. In this model, each control flow unit manages a private memory area. It collaborates with other control flow units to create the global action of the program. Many implementations of this model exist. At the opposite of the data parallel model, the task parallel model allows the programmer to explicitly control every aspects of the parallelism. She/he is responsible for explicitly managing the concurrency, synchronisation, distribution and the communications. Using the task parallelism model it is possible to emulate data parallelism. This approach is named SPMD for Single Program Multiple Data.

5.3.3 Multi-level parallelism using notion of graph

The coming of post-petscale and exascale supercomputers offers the perspective to accelerate the solution of engineering problems and to tackle highly complex models. However, these future systems challenge computer scientists to built such machines. Many issues must be faced such as fault-tolerance, energy consumption and the programming of these complex systems composed of hundred of millions of cores. In [86], authors have proposed a multi-level programming paradigm composed of three levels. At the low level, a data parallel paradigm is used to program many-cores processors for its focus on data mapping and movements. they have implemented and evaluated the SpMV with various sparse matrix formats on GPU to illustrate this point. At the intermediate level, a message passing paradigm is used in order to optimize inter-sockets and inter-nodes communications. At the high level, a graph description paradigm is used to program and manage the parallelism between nodes.

5.4 Parallel MIRAM algorithm

For PageRank computation in real applications, IRAM should not be used naively. Due to the very large problem scale, the subspace size m must be small to maintain the orthogonal basis W_m in memory. It is known that the eigen-information of interest may not appear when m is too small [96]. In addition, high damping factor results in clustered eigenvalues around the dominant one [50], which will slow down the convergence even further.

In IRAM, only the initial vector is used to improve the quality of the subspace during iterations. The authors of [40] investigate the influence of the size of subspace. The idea is to make use of Arnoldi method to compute the Ritz elements of a large matrix A in a set of l nested Krylov subspaces. If the accuracy of the Ritz elements calculated is not satisfactory in any of these subspaces, the algorithm will select the one that contains the "best" current Ritz elements. Then a QR shifted algorithm will be applied to the $m_{best} \times m_{best}$ matrix which represents A in this $m_{best}-$size projection subspace. The leading $k \times k$ submatrix issued from QR algorithm concen-

trates the information corresponding to the desired eigenvalues. Arnoldi projections are then completed on nested Krylov subspaces starting with this submatrix. This method can be considered as an IRAM with the largest subspace size, which uses eigen-information of some of its nested subspaces in order to update its restarting vector. In this paper, we focus on the parallelization of the method and present a parallel “multiple IRAM" algorithm (MIRAM).

Recall that the MIRAM procedure is described in Algorithm 2 and the overview of the algorithm is shown in Fig. 5.4.

It is important to notice that the communication of the eigen-information of interest of each IRAM process to other IRAM processes can be avoided. The idea is to run a single Arnoldi process proceeding across all processors and to save the information whenever m reaches l different values. In other words, the steps 1 to 4 in Algorithm 2 are duplicated across all processors. Furthermore, since f_m and W_m are distributed, the implicit QR iterations in steps 5 to 9 could be done locally on different processors as well.

Concerning the choice of parameter k, Stathopoulos et al. proposed in [97] a technique, called thick restart, where k_0 eigenpairs are needed, k ($k > k_0$) pairs are retained after each restart, and $r = m - k$ additional vectors are built. Some results of using thick restarting approach for the choice of parameter k are given in section 5.

Concerning the time and space complexities of MIRAM versus that of IRAM. We assume that $m \ll n$ and let nrc be the number of restarting cycles excluding the initialization. The cost of IRAM in terms of matrix-vector products for nrc restarting cycles is $m + r \times (nrc - 1)$. Indeed, in the first cycle, the number of matrix-vector products is m and for each of the restarting cycles, the number of matrix-vector products is $r = m - k$. Note that the cost of orthogonalization in a restarting cycle is $O(2 \times r \times n^2)$. When A is sparse and r is large, this cost of orthogonalization may be dominant in the computation. The space complexity of IRAM is $n^2 + O(m \times n)$.

Recall that m_ℓ is the maximum of the $m_1, \ldots, m_\ell$ subspace sizes. The cost of MIRAM in terms of matrix-vector products is $m_\ell + r_\ell \times (nrc - 1)$. Still the cost of orthogonalization in Arnoldi process is $2 \times r_\ell \times n^2$. As a result, this cost of orthogonalization may be dominant in the computation when A is

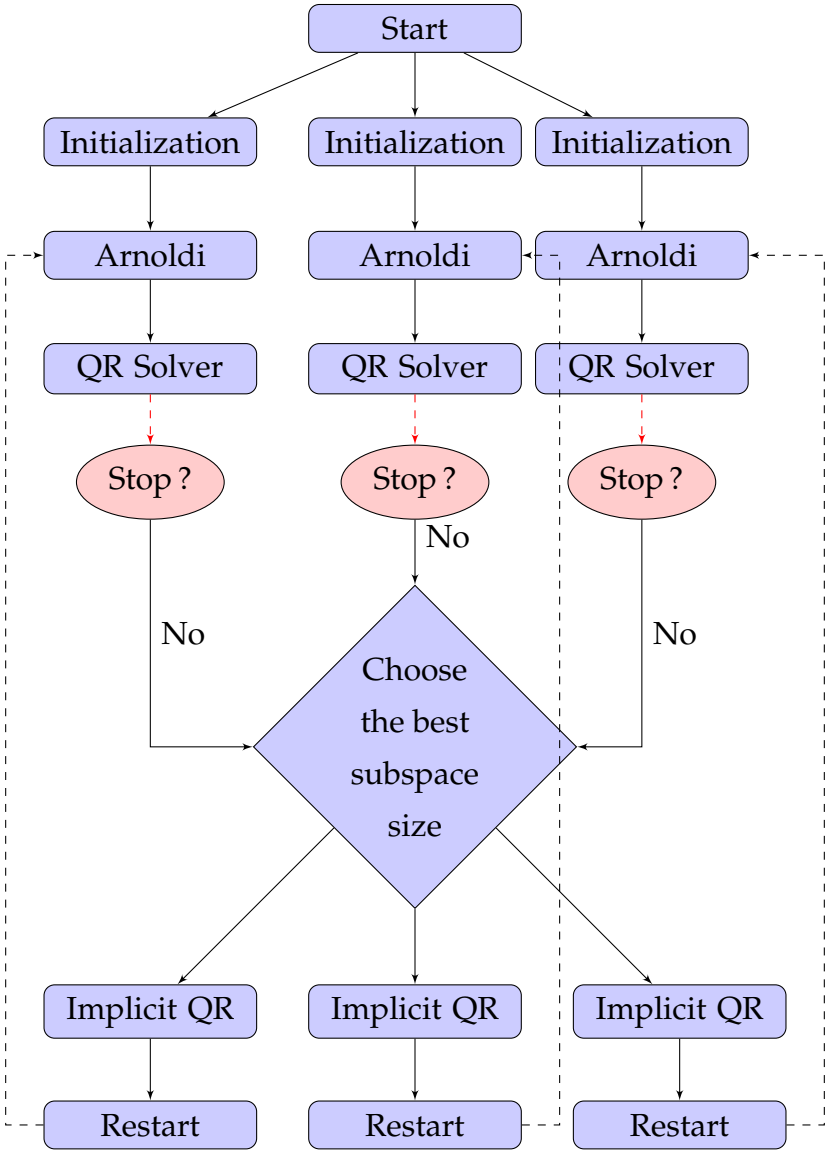


FIGURE 5.4: The overview of MIRAM

sparse and r_ℓ is large. The space complexity of MIRAM is n^2+$O(n \times m_\ell)$.

We denote by CI and CM the time complexities of *one restarting cycle* of IRAM(m_ℓ) and MIRAM($m_1, \cdots, m_\ell$) respectively. Ignoring terms not including n and the cost of stopping criterion, these complexities can be given by $CI = \alpha + 2 \times n \times m_\ell^2$ and $CM = \alpha + 2 \times n \times \left[k \times (m_1 + \ldots + m_\ell) + m_{best}^2 - k \times m_{best}\right]$, where α is a common part in both algorithms. In the worst case for MIRAM, where $m_{best} = m_\ell$, $CM - CI =$

$2 \times n \times k \times (m_1 + \ldots + m_{\ell-1})$, which is positive. In the best case for MIRAM, where $m_{best} = m_1$, $C_M - C_I = 2 \times n \times \left[k \times (m_2 + \ldots + m_\ell) + m_1^2 - m_\ell^2\right]$, which could be positive or negative. Depending on the values of k and m_i, one restarting cycle of MIRAM could be less expensive than that of IRAM. To conclude, if MIRAM$(m_1, \cdots, m_\ell)$ uses m_{best} most of the time, it will cause more computations than IRAM(m_ℓ). This is confirmed by our experiments in chapter 8.

5.5 Scalable sparse MVP for scale-free networks using hypergraph partitioning

The name "scale-free networks" comes from a project to map the struccuture of the World Wide Web in 1998, which has revealed a surprising fact that a few highly connected pages are essentially holding the World Wide Web together. Counting how many Web pages have a certain number of links showed that the degree distribution followed a power-law. Following researches observed many real networks that display similar phenomenon, among which are social networks.

When mining information from a network, eigenpairs of the various matrices that represent the network are used. Sparse Matrix vector product is the bottleneck of many existing eigensolvers for scale-free networks. This is especially true for any Krylov subspace method. There are a couple of approaches to improve the sparse MVP performance.

One way consists in balancing the workload : first, each processor should have at most $\lceil n/p \rceil$ columns ; second, each processor should have roughly equal number of nonzero elements. We could use a simple heuristic method. Suppose there are p processors. We begin by sorting the columns according to their number of nonzero elements. Then from dense to sparse we attribute the column j to processor $i(i = 1, ..., p)$. After that, the rest sorted columns should be attributed one by one to the processor with the least number of nonzero elements each time. Another constraint is when a processor has $\lceil n/p \rceil$ columns, it should not be considered for attribution any more.

However, there are a couple of issues associated with this approach. First

of all, the columns in each processor are usually not contiguous after redistribution, which will generate complex communication pattern while doing sparse MVP. A possible remedy is by reordering the nodes to make the columns in the same processor contiguous. The procedure above is equivalent to symmetrically permuting rows and columns of A. In other words, we construct a new matrix $B = T^T AT$, where T is the product of successive permutation matrices : $T = (T_1 \times T_2 \times ...)$. Then

$$Bu = \mu x \Rightarrow T^T ATx = \mu x \Rightarrow A(Tx) = \mu(Tx), \tag{5.2}$$

so that A and B have the same eigenvalues, and if x is an eigenvector of B, then $x' = Tx$ is an eigenvector of A. In consequence, the computation by MIRAM could be applied on the distributed matrix B instead.

Secondly, this workload-centric approach may result in extensive communication volume. Parallel sparse MVP for scale-free networks does not scale well due to the high communication overhead caused by hubs (the most connected nodes). While sparse, the nonzero structure of their adjacency matrices are quite different from that of a PDE matrix. Rather, the existence of hubs necessitates an all-to-all communication either before or after the reduction operation in MVP, which makes the parallel communication requirements more similar to those of a dense matrix.

To clarify this observation, we use a simple example given in Fig. 5.5 with 3 processors. The columns 0, 3 and 6 of P correspond to three hubs in the network since they contain the most nonzero elements. From left to right, the columns 0 to 2 are distributed to processor 0; the columns 3 to 5 are distributed to processor 1, and the columns 6 to 8 are distributed to processor 2. The vector w is also partitioned into three segments (marked by three colors) and distributed among these processors.

Before the reduction of MVP, the processor that owns the column j needs only the corresponding element $w[j]$ in the vector w, which is also local to this processor. After the reduction operation to get its “partial sums" (in row-wise), each processor sends its partial sums to the processor that owns the vector segment for the corresponding rows. For example, after local reduction, the processor p_1 will get 9 partial sums, numbered from 0 to 8. The partial sums 0 to 2 will be sent to the processor 0 since it owns the

red segment (the rows 0 to 2) of the vector w. Due to the existence of hubs, each processor will have 9 partial sums in the example. As a result, using 1D column-wise partitioning, every processor might be required to send messages to all other processors. This results in an all-to-all communication after the local reduction. Similarly, if we use 1D row-wise partitioning, an all-to-all communication before the reduction will be needed because of the existence of hubs.

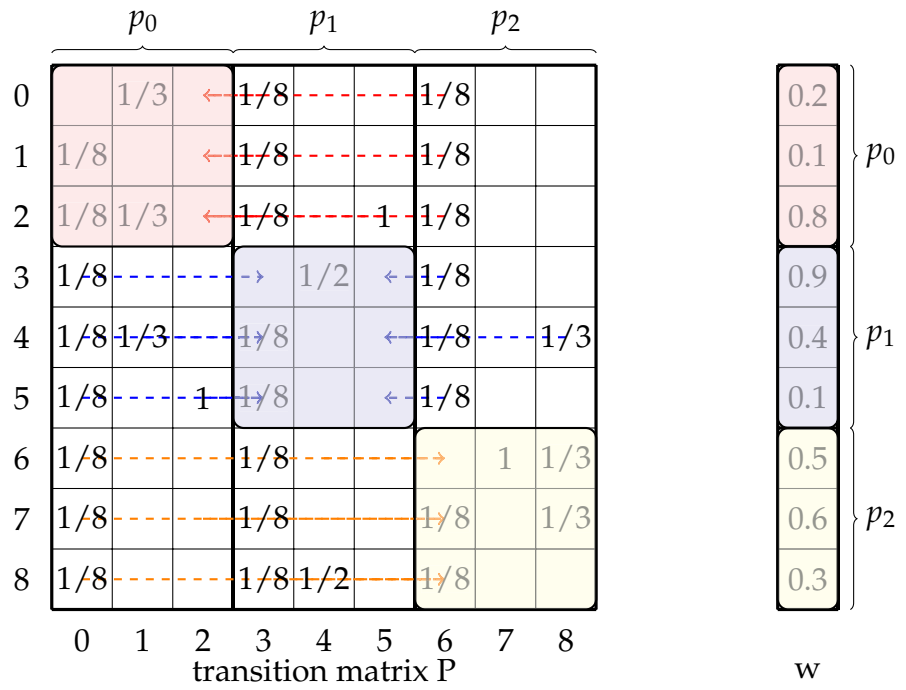


FIGURE 5.5: An 1-D column-wise partitioning on 3 processors and its matrix vector multiplication.

In the second approach, the problem of load distribution and balancing in parallel MVP is formulated as a graph partitioning one. The idea is to find the subsets of nodes in the origin graph such that the number of edges between any two partitions are minimized. The nodes correspond to different rows/columns in the matrix A, and the edges between two partitions represent the communication requirements between two processors in parallel MVP. There are some popular graph partitioning packages, such as Chaco, Metis and Scotch [51][56][84]. They also offer MPI-based libraries for parallel graph partitioning. These packages are based on row-wise partitioning where each processor holds a block of rows of the matrix. From

a matrix theoretical view, they simply try to minimize the total number of off-block-diagonal nonzeros without considering the relative spatial locations of such nonzeros. In other words, the graph models treat all off-block-diagonal nonzeros in an identical manner by assuming that each of them will incur a distinct communication of a single word [24]. However, before the reduction, the off-block-diagonal nonzeros in the same column generate only one message to get the corresponding vector component. After the reduction, the off-block-diagonal nonzeros in the same row reduce to one partial sum and incur only one message as well.

Recently, hypergraph-based partitioning [24] has drawn much attention from the PageRank community. We will continue our discussion firstly with a retrospect of some basic definitions of the hypergraph theory.

Definition 1. *A hypergraph* $\mathbb{H} = (V, N)$ *is defined as a set of vertices* V *and a set of nets (hyperedges)* N *among these vertices. Every net* $n_j \in N$ *is a subset of vertices, i.e.,* $n_j \subseteq V$.

Definition 2. *A k-way partition* Π $(k > 1)$ *of the set* V *is defined as* $\Pi = \{V_1, \cdots, V_k\}$, *where* V_i *are subset of* V *s.t.* $V_i \bigcap V_j = \emptyset$ *for all* $1 \leq i < j \leq k$.

Definition 3. *The* $k-1$ *metric is defined as*

$$f(\mathbb{H}) = \sum_{i=1}^{n^*} (\pi_i - 1)\omega_i \tag{5.3}$$

where n^* *is the number of nets,* π_i *is the number of subsets that the net* n_i *spans (i.e. has a vertex in) and* ω_i *is the number of constituent vertices of the net* n_i.

The hypergraph partitioning problem consists in finding a k-way partition $\Pi = \{V_1, \cdots, V_k\}$ such that the $k-1$ metric is optimized, and the number of vertices in each subset V_i is balanced. For 1D sparse matrix decomposition scheme, a matrix A is represented as a hypergraph $\mathbb{H}_{\mathbb{R}} = (V_R, N_C)$. Vertex and net sets V_R and N_C correspond to the rows and columns of matrix A, respectively. The distribution of the rows of matrix A to p processors for parallel sparse MVP corresponds to a p-way partition of the above hypergraph. For 2D sparse matrix decomposition scheme, the objective is to distribute matrix nonzeros to processors instead. Here, each nonzero is represented by a vertex. Every column/row is modelled by a net. Its constituent

vertices are the nonzeros of the column/row. In consequence, minimizing communication before and after the reduction of MVP could be accurately modelled by a hypergraph partitioning problem. Using these two schemes, Bradley et al. has observed a reduction of communication by a factor of three compared to conventional graph partitioners [20]. In our implementation, we use the "Zoltan" package [17] as hypergraph partitioning tool. The drawback is that it takes longer to run than graph algorithms.

5.6 Parallel implementation

Today, the building block of the high-end computing system consists of multiple multi-core chips sharing memory in a single node. We use a hybrid programming model with message passing and shared memory (MPI and OpenMP). This model assumes that the system has a number of nodes with local memories that communicate with each other by means of message transfer. In the meantime, each node is composed of a number of processors sharing a local memory. There is thus a hierarchical two-level parallelization in our implementation. The first one applies the 1D row-wise hypergraph partitioning for minimizing the communication in sparse MVP. Each MPI process works on one group of rows and exchanges data before the reduction operation. Parallelism in the first level is limited to the number of computing nodes available in the system. For the second-level parallelism, MVP kernel uses OpenMP parallel regions for local multiplication and reduction within a node. Our code is developed based on the Trilinos framework [52], where about fifty C++ packages are included.

From the developer's view, parallel MIRAM consists of three components. The first is a network loader to store the entire network in memory on a distributed memory parallel computer, the second is the Zoltan package that preprocesses the parallel matrix for load balancing and communication minimization and the third is the eigensolver described in Fig. 5.4.

5.6.1 Network loader

The networks are initially stored as edge set in a coordinate format file. We parse the file and derive the corresponding transition matrix P and store it in matlab coordinate format. An example of 5 nodes is given in Fig. 5.6. The two columns on the left of the Table 5.1 are the endpoints of the edges, while the three columns on the right are the triplet (row_index, col_index, value).

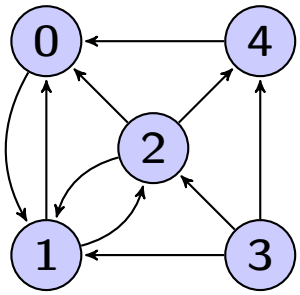


FIGURE 5.6: Small social network of 5 individuals

TABLE 5.1: Example of network coordinate format and matlab coordinate format for Fig.5.6

network coordinate format		matlab coordinate format		
0	1	2	1	1
1	0	1	2	0.5
1	2	3	2	0.5
2	0	1	3	0.333
2	1	2	3	0.333
2	4	5	3	0.333
3	1	2	4	0.333
3	2	3	4	0.333
3	4	5	4	0.333
4	0	1	5	1

After the conversion, we use *MatlabFileToCrsMatrix* function (in Trilinos' EpetraExt package) to load the matrix. Epetra provides construction routines as well as services function for data objects in distributed memory parallel machines. A class called *Epetra_Map* describes the mapping of every

vector and matrix over MPI ranks. Vectors have a single 1D map while sparse matrices may have 1D or 2D maps. 1D row-wise/column-wise distribution of sparse matrices is specified by row/column map. The 2D distribution can be specified by giving both row map and column map to the constructor of matrices. In our implementation, we store the transition matrix P as an *Epetra_CrsMatrix* using row map.

5.6.2 Hypergraph partitioner

The main focus is to improve the scalability of sparse matrix vector multiplication over scale-free networks. To do so, we use the *Isorropia* package, interface to the Zoltan toolkit. It performs the partitioning mainly through three steps :

1. Create a *Isorropia : :Partitioner* instance.
2. Create a *Isorropia : :Redistributor* object.
3. Use the *Isorropia : :Redistributor* to redistribute one or more objects to the new partitioning.

Weights can be defined by *Isorropia : :CostDescriber* class for graphs and hypergraphs. *Isorropia* currently supports partitioning/redistributing of several Epetra objects, including *Epetra_CrsGraph* and *Epetra_CrsMatrix*, etc. *Isorropia* has a number of parameters that control the partitioning methods [**?**]. These parameters are placed in a *Teuchos : :ParameterList* object, which is passed as an argument to the following *Isorropia*'s function :
Epetra_CrsMatrix Isorropia : :Epetra : :createBalancedCopy(const Epetra_CrsMatrix & input_matrix, const Teuchos : :ParameterList & paramlist)*.
We implement the hypergraph partitioner by calling
paramlist.set("PARTITIONING METHOD", "HYPERGRAPH").

5.6.3 Parallel MIRAM

MIRAM consists of four main tasks. First, the projection phase manipulates the n-sized data sets for sparse MVP. The second phase including implicitly shifted QR iterations acts on m-sized data sets. The third phase constructing the r additional steps of Arnoldi factorization manipulates on

n-sized data sets as well. At last, the convergence test deals with n-sized data sets to calculate $\|f_m\|$. Because phase one and three constitute the most expensive part of the algorithm, we propose to distribute them among processors and to run phases two and four redundantly on all processors.

To conduct sparse MVP, Epetra uses two additional maps to specify the distribution of the input (*domain map*) and the output vectors (*range map*). Both the domain and range maps are one-to-one : that is, each global index in the map is uniquely owned by only one process. There are four steps for sparse MVP implemented in Epetra :

1. Import : Send w_i to the processes that own a nonzero a_{ij} for some i.
2. Local reduction : $y_i := y_i + a_{ij} * w_j$.
3. Export : Send partial y values to the owner processes.
4. Reduction : Add up partial y contributions received.

The communication steps 1 and 3 are point-to-point in Epetra and are implemented as the *Epetra_Import* and *Epetra_Export* classes respectively. In our MIRAM code, we use the following function :

int Epetra_CrsMatrix : :Multiply(bool TransA, const Epetra_Vector & x, Epetra_Vector & y)

of *Epetra_CrsMatrix* class to perform MVP on the matrix P. The Importer and the Exporter classes will be automatically constructed based on its maps.

Chapitre 6

Parallel Social Graph Generator for PageRank Computation

6.1 Introduction

In social science, real data about individuals' relationships are generally hard to collect. This is especially true for large population such as Chinese (more than 1.3 billion people). Furthermore, this kind of data is often not accurate since it depends sometimes on individuals' subjective judgments. For example, it is hard to decide if one person is friend of another person. Because they could have different answers to this question. Lack of realistic data, scientists tend to use synthesized or network-based social graphs to study various social problems including epidemiology. Moreover, studies in other complex networks, such as the Internet [38], biological networks [45], and various infrastructure networks [26], motivate the efficient generator of massive random networks.

The recent advances in the research of complex networks have significantly increased the interest in various random graph models [12]. Among them, the first and well-studied model is the Erdös-Rényi model [37]. However, this model does not exhibit the characteristics observed in many real-world complex systems. As a result, many other models emerges, such as small-world [103], Barabàsi-Albert [10] and exponential random graph [88], etc.

The scale-free network is a network whose degree distribution follows a

power law asymptotically. In mathematical formalism, the fraction $P(k)$ of nodes in the network that has k edges to other nodes follows

$$P(k) \approx k^{-\gamma}$$

where γ is a parameter whose value is typically $2 < \gamma < 3$. Preferential attachment [10] and the fitness model [11] have been proposed as mechanisms to explain conjectured power law degree distributions in real networks. A few examples of networks claimed to be scale-free include : Social networks, many kinds of computer networks, some financial networks such as interbank payment networks, protein-protein interaction networks, semantic networks and airline networks.

A small-world network is a type of mathematical graph in which most nodes have a very few neighours, but most nodes can be reached from every other by a small number of hops or steps. In mathematical formalism, a small-world network is defined to be a network where the typical distance L between two randomly chosen nodes (the number of steps required) grows proportionally to the logarithm of the number of nodes N in the network,

$$L \propto logN$$

Small-world properties are found in many real-world phenomena, including websites with navigation menus, food chains, electric power grids, metabolite processing networks, networks of brain neurons, voter networks, telephone call graphs, and social influence networks.

Demand for large random networks necessitates efficient algorithms to generate such networks. Recently, some efficient sequential as well as parallel algorithms have been developed to generate scale-free graphs [12, 4].

However, although efficient algorithms proposed are able to generate networks with billions of nodes quickly, generating networks with both scale-free and small world structure has not been well discussed. In this chapter, we propose an efficient and highly scalable parallel graph generation algorithms that can produce massive social graphs. The method employed is so general that can be used to identify and mine certain knowledge or data of interest by other algorithms. The synthetic graphs generated possess the most common properties of real complex networks such as power-law degree distribution and small-worldness.

6.2 Sequential algorithm for Barabàsi-Albert model

The common way to generate a scale-free network is to use the Barabàsi-Albert model (BA model). Starting with a small clique of s nodes, in each step, a new node is added to the network and connected to k ($k \leq s$) randomly chosen existing nodes based on the degrees of the nodes in the current network. The idea is to connect a new node to an existing node that is chosen with probability directly proportional to its current degree. In other words, the probability p that node t is connected to node i ($i < t$) is given by $p_i = \frac{d_i}{\sum_j d_j}$, where d_j represents the degree of node j. Barabàsi and Albert showed this preferential attachment method of selecting nodes results in a power-law degree distribution [10]. The algorithm is described as follows.

Algorithm 3: Sequential BA algorithm

Input : a small clique of s nodes.

For $j = 1, \ldots, k$

1. Calculate the p_i for every node i in the current network.
2. Choose an existing node h according to its corresponding p_h value.
3. Add an edge (i,h) to the network and update the degree of node h.

6.3 Parallel algorithm for graph generator

Scale-free graphs can be easily generated using preferential attachment. Starting with a small clique, a scale-free network is constructed by repeatedly adding a new node and attach it to one of the existing nodes with probability proportional to its current degree. We parallelize this method by distributing the vertices among processors, and all their adjacent edges are stored on the same processor to which the vertex is assigned. We additionally create the small world structure by grouping sets of processors. Each processor belongs to one or several groups. Each group corresponds to a community in social networks.

The proposed algorithm is based on the work [108] and is described as follows.

Algorithm 4: Parallel BA algorithm

Input : a small clique of s nodes distributed among all processors.
a set of groups $\{G_0^{(p)}, G_1^{(p)}, \cdots, G_{n-1}^{(p)}\}$ for each processor p.
For each newly created node on processor p, we need to add k edges.

1. For each of these k edges, select randomly a processor from $\{G_0^{(p)}, G_1^{(p)}, \cdots, G_{n-1}^{(p)}\}$.
2. Send message <n_{pq}> to each selected processor q, where n_{pq} is the number of endpoints needed by processor p in processor q.
3. Receive message<n_{qp}> from processor q, where n_{qp} is the number of endpoints needed by processor q in processor p.
4. Choose n_{qp} endpoints using sequential BA algorithm (shown in Algorithm 3) for each selected processor q and send them to processor q.
5. Receive message containing endpoints from each selected processor q and add them to edges corresponding to q.

We assume that the processor p is a member of groups $G_0^{(p)}, G_1^{(p)}, \cdots, G_{n-1}^{(p)}$. An edge is attached in two phases. In the first phase, k edges are added per newly created local node (a node that resides on p) as in the conventional Barabàsi-Albert model. However, each edge, e, associates a local node with some processor q, instead of connecting two nodes as in the serial model. The particular node that is to be the eventual endpoint of e is determined remotely by the processor q. The processor q is selected randomly from groups $G_0^{(p)}, G_1^{(p)}, \cdots, G_{n-1}^{(p)}$. Let A denote a local edge list maintained by the processor p. First, we initialize A by associating the first s edges with the processors in groups $G_0^{(p)}, G_1^{(p)}, \cdots, G_{n-1}^{(p)}$. For an edge e_j , where $j \geq s$, we select an existing edge from A with a uniform probability and then assign its associated processor to e_j (thus realizing preferential attachment). This process is repeated until the predetermined number of local nodes and edges are created on p. At the end of the first phase, p sends a message to each processor q to notify the number of

TABLE 6.1: An example of graph construction on processor p_0.

u	0	0	1	1	2	2	3	3	4	4
v	p_1	p_2	p_0	p_1	p_1	p_2	p_0	p_1	p_0	p_2

u	0	0	1	1	2	2	3	3	4	4
v	8	p_2	p_0	7	5	p_2	p_0	8	p_0	p_2

occurrences of q in A. In the second phase, p determines the endpoints for the edges on remote processors and connects the endpoints calculated by remote processors to its local vertices. The processor p first receives messages from other processors, which contain the numbers of occurrences of p in their respective local edge list. That is, the message received from a processor q represents the number of incomplete edges one of whose endpoints resides on the processor q. These edges are to be connected to the local nodes on p, selected by using the standard preferential attachment technique. Once the list of the nodes for the attachment is determined, it is divided up among the processors. Here, each processor is assigned as many nodes as requested. The selected nodes are then sent to the corresponding processors. Having sent the endpoints for the remote edges, then p receives the lists of endpoints from other processors for its own incomplete edges. Using the remote nodes received, p completes its local partition of the graph. This is done by simply substituting each occurrence of processor q in A with the next endpoint in the list sent by q. The resulting collection of edges defines the portion of the graph stored on p.

The algorithm is illustrated by an example shown in Table 6.1. In this example, we generate a graph with 5 nodes per processor and 2 edges per node. It is assumed that there are three groups, $G_0 = \{p_0, p_2\}$, $G_1 = \{p_1, p_2\}$, and $G_2 = \{p_0, p_1\}$ and processor p_0 belongs to groups G_0 and G_2. The nodes are assumed to be evenly distributed among the processors so that nodes 0-4 are on p_0, nodes 5-9 on p_1, and so on. In the first phase of the algorithm, p_0 selects processors and associates them with the local nodes as shown in Table 6.1 where the edge list on p_0 is depicted. Note that the first four processors in the list are the ones in the factions that p_0 belongs to G_0 and G_2. The rest of the processors in the list are selected using the standard prefe-

rential attachment technique. At the end of phase 1, p_0 needs four endpoints from p_1 (and three endpoints from each of p_0 and p_2). In processor p_1, endpoints are determined via preferential attachment and sent to p_0 in the second phase. In this example, we assume that vertices 8, 7, 5, and 8 are sent to p_0. Once receiving the list, p_0 simply replaces the entries marked with p_1 with the endpoints in the list. This is shown in Table 6.1.

6.4 In-memory matrix representation for PageRank computation

To transform a generated parallel graph into a distributed Google matrix, we predefine a data mapping (partitioning) for the parallel matrix. Partitioning (load balancing) is the problem of assigning data and computation to processes. Its goal is to balance the load (data, computation) while also reducing interprocess communication during computation. A node corresponds to a row and an edge corresponds to a nonzero element in the parallel matrix. Two endpoints of an edge correspond to row and column indices of the nonzero element. Initially, the rows are stored locally in the processor where the corresponding nodes reside. Then, a 2D hypergraph partitioning as presented in the previous chapter is performed on the parallel matrix to obtain a better load balancing scheme.

6.5 Conclusion

A parallel algorithm that can generate scale-free and small world networks are discussed in this chapter. The network generated is distributed among processors and could be further explored to obtain information of interest.

Chapitre 7

Numerical results

7.1 Introduction

7.2 Data of experiments

In social science, lack of realistic data, scientists tend to use synthesized or network-based social graphs to study various social problems including epidemiology. Many studies show that web graphs display similar underlying structure as social graphs such as power law distribution of degrees and small-world phenomenon. In the following section, we present our results on seven networks. Their statistics are presented in Table 7.1. n is the number of nodes, nnz is the number of links. The number of links in the table is bigger than that in initial datasets because we add links for dangling nodes. *ba* is collected at the Oregon router views [1]. *com-Youtube*, *stanford*, *Wiki-Talk* and *soc-LiveJournal1* are obtained from Stanford Large Network Dataset Collection [2]. *twitter* is collected from 467 million Twitter posts from 20 million users covering a 7-month period from June 1, 2009 to December 31, 2009 [58]. This dataset is more realistic to represent a social network. *yahoo* contains URLs and hyperlinks for over 1.4 billion public web pages indexed by the Yahoo ! AltaVista search engine in 2002.

The statistics for the above datasets are presented in Table 7.1. n is the number of nodes, nnz is the number of links. The number of links in the table is bigger than that in initial datasets because we add links for dangling nodes.

TABLE 7.1: Statistics for datasets

Name	n	nnz	Storage
ba	7010	13985	117 KB
stanford	281,903	2,321,669	30 MB
com-Youtube	1,134,890	2,988,374	38.7MB
Wiki-Talk	2,394,385	5,21,410	66.5MB
soc-LiveJournal1	4,847,571	68,993,773	1.1GB
twitter	41,652,230	1,469,914,131	25 GB
yahoo	1,413,511,394	8,050,112,173	78 GB

7.3 Architecture and machines

Grid5000 platform.

We run our experiments on the nation wide cluster of clusters Grid5000. Grid5000 is a scientific instrument for the study of large scale parallel and distributed systems. It provides a highly reconfigurable, controllable and monitorable experimental platform to its users. The infrastructure of Grid'5000 is geographically distributed on different sites hosting the instrument, initially 5 sites in France (since 2005). Porto Alegre, Brazil is now officially becoming the first site abroad. We conduct our experiments mainly on five clusters as follows (some hardware details are given in Table 7.2) :
- Cluster "Taurus" : 16 nodes×2 cpus per node×6 cores per cpu=192 cores.
- Cluster "Graphene" : 144 nodes×1 cpus per node×4 cores per cpu=576 cores.
- Cluster "Paradent" : 64 nodes×2 cpus per node×4 cores per cpu=512 cores.
- Cluster "Granduc" : 22 nodes×2 cpus per node×4 cores per cpu=176 cores.
- Cluster "Griffon" : 120 nodes×2 cpus per node×4 cores per cpu=960 cores.

7.4 Metrics of performance used

The following metrics are used : execution time, residual norm, strong scalability, time evolution of epidemic spread, vaccination strategy based

TABLE 7.2: Hardware details of Clusters

Name of Cluster	CPU	Network	Memory
Taurus	Intel Xeon	Gigabit Ethernet	32 GB
Graphene	Intel Xeon X3440	Gigabit Ethernet	16 GB
Paradent	Intel Xeon L5420	Gigabit Ethernet	32 GB
Granduc	Intel Xeon L5335	Gigabit Ethernet	16 GB
Griffon	Intel Xeon L5420	Gigabit Ethernet	16 GB

PageRank.

7.5 Results

7.5.1 Comparison with other methods

In the first place, we compare the behaviour of residual norm $\| Ax - x \|$ of single IRAM with that of power method. The idea of Power method is to write the initial vector x_0 as a linear combination of $\sum_{j=1}^{n} \alpha_j v_j$, where v_j are eigenvectors of A. Without loss of generality, suppose λ_1 is the dominant eigenvalue, we have :

$$\begin{aligned} x_k &= Ax_{k-1} = A^2 x_{k-2} = ... = A^k x_0 \\ &= A^k \sum_{j=1}^{n} \alpha_j v_j = \sum_{j=1}^{n} \alpha_j A^k v_j = \sum_{j=1}^{n} \lambda_j^k \alpha_j v_j \\ &= \lambda_1^k (\alpha_1 v_1 + \sum_{j=2}^{n} (\lambda_j/\lambda_1)^k \alpha_j v_j) \end{aligned} \tag{7.1}$$

For $j > 1$, $|\lambda_j/\lambda_1| < 1$, so that $(\lambda_j/\lambda_1)^k \to 0$, leaving only the PageRank eigenvector v_1. We choose the damping factor to be 0.85 and the tolerance value to be $1E-7$ for both methods. The initial vector is taken as the vector $e = (1, 1, ..., 1)^T$. Within IRAM, the size of Krylov subspace is 4 and the number of shifts used is 3. The result is presented in Fig. 7.1. This figure shows that residual in IRAM decreases solidly even in the first iteration while power method has big residual norms during the initial iterations.

Noticed that a power iteration is computationally much cheaper than an IRAM iteration. To avoid confusion, in Table 7.3, we compare the number

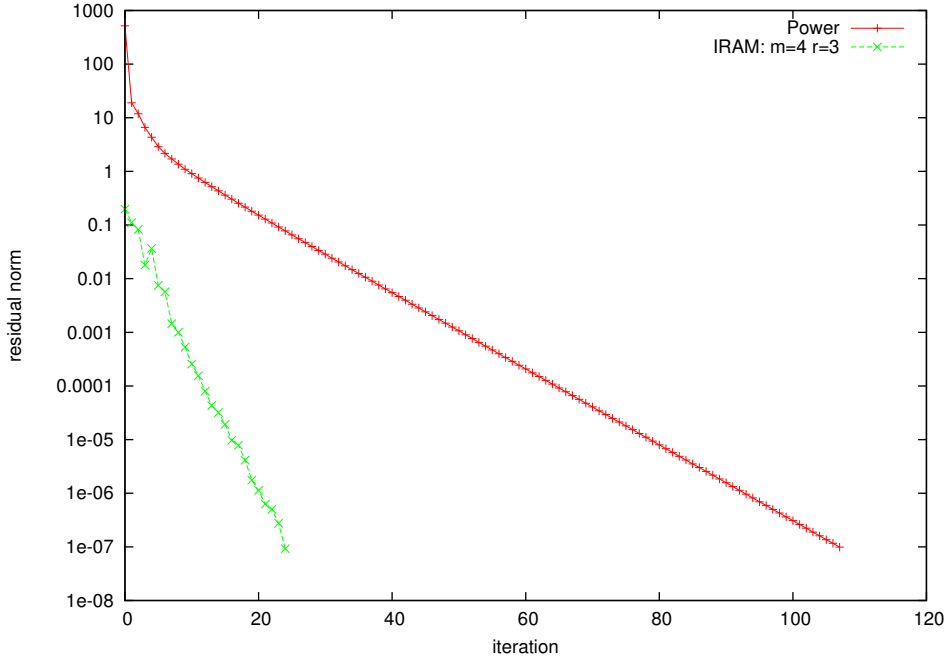


FIGURE 7.1: Convergence behavior for the $281,903 \times 281,903$ *stanford* matrix, $\alpha = 0.85$

TABLE 7.3: Number of matrix vector products for the $281,903 \times 281,903$ *stanford* network

α	Power	m=4	m=8	m=16
0.85	122	79	71	61
0.90	184	124	99	91
0.95	367	238	148	136
0.99	1775	394	358	316

of matrix vector products used in both methods, where α is the damping factor, m is the size of Krylov subspace. We choose the number of shifts to be $m - 1$. The tolerance value used is $1E - 8$ and the initial vector is taken as the vector e. This table shows that the number of matrix vector products used in IRAM are less than that of power method.

From the two tests above, we could conclude that IRAM has a faster convergence than the power method for the test matrix. Also, some experiments using explicitly restarted Arnoldi method on this network have been given in [47]. Still, we find a faster convergence than the power method.

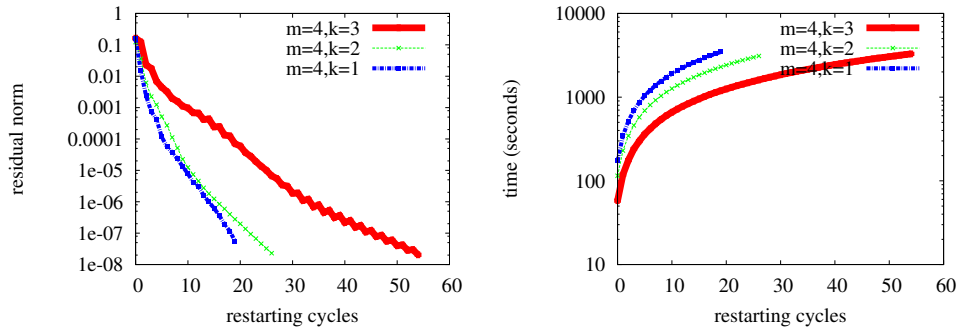
7.5.2 Experiments on damping factor α

We study the influence of damping factor on convergence rate. For *stanford* network, this dependency is quantified in Table 7.3. It is found that with bigger damping factor α, more iterations are needed to reach the accuracy for both methods. However, IRAM has a much better performance than Power method for bigger α. As explained in [50], bigger α engenders a closer-to-1 second largest eigenvalue. This fact also favors Arnoldi-type methods, as opposed to the power method.

7.5.3 Thick restart for the choice of parameter k

We first check the strategy proposed by Stathopoulos et al. in [97] for the choice of parameter k. The damping factor α is fixed to 0.85.

In the test on *twitter* network, we set the m to be 4 and change the value of k to 1, 2 and 3. We run our experiments on cluster "Taurus" using 16 nodes with 2 MPI processes per node (without OpenMP multithreading). The results are presented in Fig. 7.2. While $(k = 1)$ uses the fewest restarting



(a) Number of restarting cycles

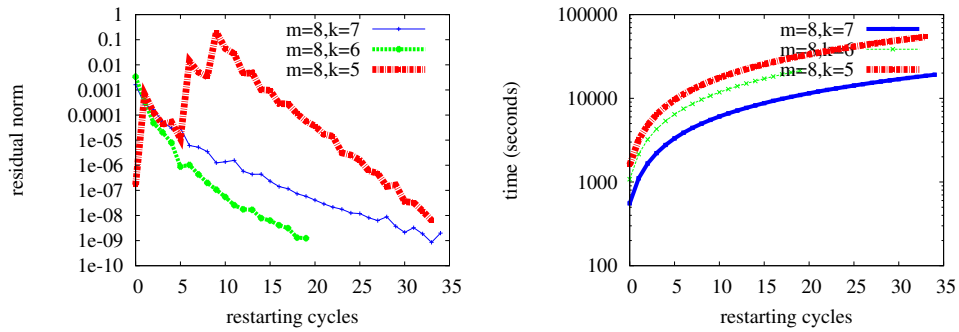
(b) Execution time

FIGURE 7.2: Convergence experiments for different number of shifts on *twitter* network, where $\alpha = 0.85$ and $tol = 1E-7$.

cycles, $(k = 2)$ allows the fastest convergence in terms of execution time. In consequence, keeping a buffer of 1 extra vector accelerates the convergence rate for the dominant eigenpair.

Similar experiments are conducted for *yahoo* network using 144 nodes of "Graphene" cluster with one MPI process per core. The results are presented

in Fig. 7.3. Parameter configurations $(k = 7)$ and $(k = 6)$ have almost the



(a) Number of restarting cycles

(b) Execution time

FIGURE 7.3: Convergence experiments for different number of shifts on *yahoo* network, where $\alpha = 0.85$ and $tol = 1E - 8$.

same convergence rate, while $(k = 5)$ converges much slower.

To sum up, retaining more eigenvectors in IRAM $(k > 1)$ is generally beneficial to the convergence of dominant eigenpair.

7.5.4 Strong scalability tests

In this experiment, we run each MPI process on one 4-core CPU with 4 OpenMP threads. So each core has only one OpenMP thread running on it.

Firstly, we test the scalability of sparse MVP on *com-Youtube* network. Fig. 7.4 shows the computation time as a function of number of processors. The first curve in the top-down order corresponds to an equal-partitioned scheme with $\lceil n/p \rceil$ rows per processor. The curve below shows the strong scalability result of hypergraph partitioning on matrix A. Equal-partitioned scheme leads to slower computation due to more significant communications overhead. The result shows that the hypergraph partitioning strategy is useful to handle matrices of this particular structure. And our implementation has obtained up to $11\times$ acceleration with many cores.

In the second place, we conduct scalability tests for our parallel MIRAM implementation. The experiments are conducted for *com-Youtube* and *soc-LiveJournal1* matrices. Still, we see that the hypergraph-based implementation outperforms the equal-partitioned version. With 160 processors, we have obtained an acceleration up to $27\times$.

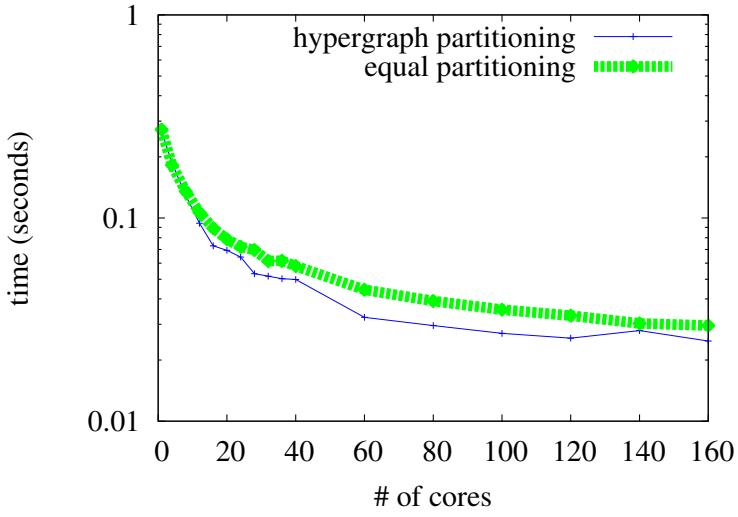
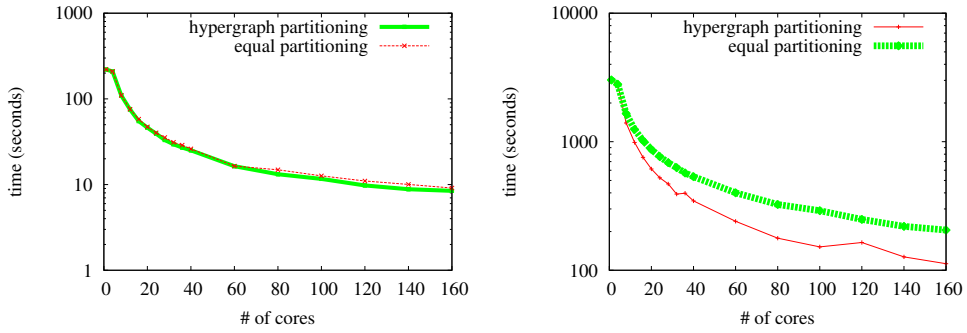


FIGURE 7.4: Scalability experiment of sparse MVP for *com-Youtube* network, where $\alpha = 0.85$, on "Griffon cluster"



(a) MIRAM(4,8) for *com-Youtube*, on "Griffon cluster"

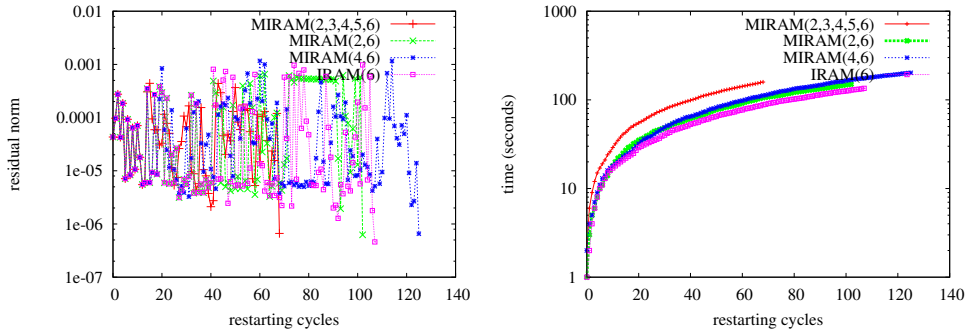
(b) MIRAM(4,8,16) for *soc-LiveJournal1*, on "Granduc cluster"

FIGURE 7.5: Scalability experiment of MIRAM, where $\alpha = 0.85$, $k = 2$ and $tol = 1E-12$.

Parallel efficiency has tendency to decrease as the number of nodes increase. This is because the communication overhead is important in grid systems. As shown in Fig. 7.3(b), with 144 grid nodes, we could expect an execution time around 8 hours for a very large network such as *yahoo*, comparable to a country/continental wide realistic scenario.

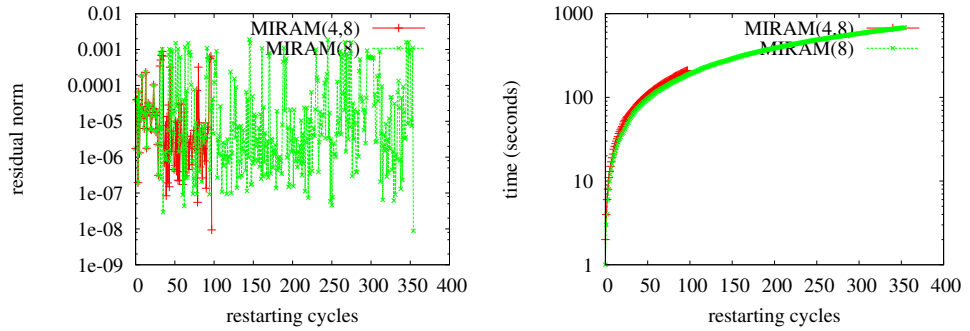
7.5.5 MIRAM vesus IRAM

Concerning the use of the parallelism of the system, we use 30 nodes from “Griffon" cluster. We run one MPI process on each node with 8 OpenMP threads (one OpenMP thread per core). Totally, 240 cores are used for each test.



(a) Number of restarting cycles (b) Execution time

FIGURE 7.6: MIRAM versus IRAM for *com-Youtube*, where $\alpha = 0.99$, $k = 1$ and $tol = 1E-6$.



(a) Number of restarting cycles (b) Execution time

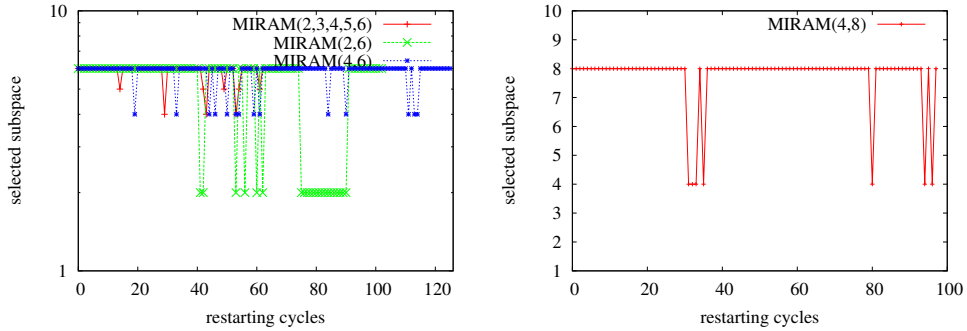
FIGURE 7.7: MIRAM versus IRAM for *com-Youtube*, where $\alpha = 0.85$, $k = 1$ and $tol = 1E-8$.

In Fig. 7.6(a), MIRAM(2,3,4,5,6) and MIRAM(2,6) use fewer restarting cycles than IRAM(6). The result shows that the convergence of MIRAM can be better than that of IRAM. Nevertheless, MIRAM(4,6) using the most restarting cycles indicates that an unfortunate parameter setting for MIRAM could result in slower convergence. Moreover, it is not the number of sub-

space spaces who counts. In fact, MIRAM(2,6) uses the fewest restarting cycles in this test.

Fig. 7.6(b) shows that IRAM(6) has the fastest convergence in terms of execution time. As analysed in section 3.2, one restarting cycle of MIRAM (when m_ℓ is chosen) is more expensive than that of IRAM. Indeed, from Fig. 7.8(a), we see that m_ℓ is used most of the time for all three MIRAMs. That is the reason why MIRAM spends less restarting cycles but uses more execution time.

The good news is that MIRAM can significantly reduce the number of restarting cycles, which could compensate for its additional computation cost. This is demonstrated by the result shown in Fig. 7.7 and Fig. 7.8(b).



(a) $\alpha = 0.99$ and $tol = 1E-6$ (b) $\alpha = 0.85$ and $tol = 1E-8$

FIGURE 7.8: Evolution of m_{best} in MIRAM along restarting cycles for *com-Youtube*, where $k = 1$.

Due to the limitation on subspace size for large scale applications, IRAM may not be efficient for computing the dominant eigenvector for such large sparse non-Hermitian matrices. Making use of several nested Krylov subspaces could help to improve the convergence as shown in our experiments. Furthermore, the number of MVP in MIRAM is decided by the largest subspace size because other subspaces are nested within this one. As a result, MIRAM$(m_1, \cdots, m_\ell)$ accelerates the convergence of IRAM(m_ℓ) with the same number of MVP in each restarting cycle.

7.5.6 Vaccination strategies based on PageRank

In this experiment, we use a small network *ba* to simulate the real world epidemic spread with distribution of vaccination. We consider people who receive vaccination being permanently immunized against viruses. For larger network, parallelization will be needed due to the memory and computation requirement but the implementation of such parallel simulator is not the objective of the test.

We assume a universal infection rate ν, a jumping rate $1 - \alpha$ (damping factor) and a curing rate δ for every individual. Before each simulation, we randomly choose a set of infected individuals. Then the propagation of virus proceeds by time step. During each time step, an infected individual infects each of its neighbours with probability ν. And this infected individual also passes the disease to another random chosen non-neighbour by probability $1 - \alpha$. Additionally, every infected individual is cured with probability δ. The result is the average over 10 runs and it is presented in Fig. 7.9. Here,

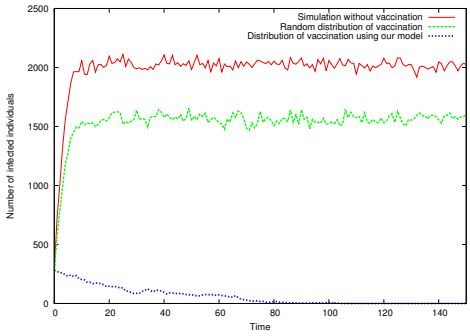


FIGURE 7.9: Time series of infection in an 7010-node power-law social network *ba*, with $\alpha = 0.85$, $\nu = 0.2$ and $\delta = 0.24$

we compare three cases. First of all, without distribution of vaccination, we try to give the worst case for time evolution of infection. Secondly, with random distribution of vaccination, we begin the simulation by distributing vaccination to a random chosen group of individuals. Then, we simulate time evolution of infection. Thirdly, with distribution of vaccination using the PageRank-like vector, we calculate the infection vector for the underlying social network and then distribute vaccination to individuals with big ranking in the vector.

The figure verifies the absence of epidemic threshold in scale-free networks [89]. Without interventions, the epidemic will always enter an endemic state. The second curve, in top-down order from the figure, shows that random distribution of vaccination could not prevent the virus from entering the endemic state. However, distributing vaccination to individuals with big ranking in the PageRank-like vector makes the epidemic die out quickly. This simple experiment confirms the important implication of infection vector for the control of epidemic spread.

7.6 Conclusion

Discuss these results in the context of models and numerical methods proposed.

Modeling of epidemic spread benefits a lot from network research to understand infection evolution in a population. PageRank-like model could give insight for understanding the impact of social network structure on propagation of virus and could possibly help identifying individuals most likely to spread the disease. Besides, parallelization makes the model computationally adavantageous over Monte Carlo simulation. Numerical results obtained are quite promising.

We demonstrate that PageRank can be computed using numerical methods based on sparse MVP and propose to use a multiple implicitly restarted Arnoldi method. The proposed parallel MIRAM implementation takes into account the scale-free structure of the underlying networks and is scalable to handle memory and computation issues arising from very large networks such as *twitter* and *yahoo* network. From our tests, we have obtained a speedup of 27$\times$ compared to sequential solver. Additionally, it is found in Experiment 7.5.3 that the shifts used in a single IRAM process could help to accelerate the convergence of method even under constraints caused by storage.

For future work, we intend to expand the proposed epidemic model by including various indicators of epidemic spread, such as characteristics of individuals as well as that of viruses, spreading timestamps, etc. Moreover, we intend to investigate the behaviour of auto-tuning strategy based on

IRAM within the context of PageRank.

Chapitre 8

Conclusion and future work

The present thesis addresses the computational modelling problem of complex systems and health systems. The objective is to investigate the computational approach for modelling very large scale complex phenomenon such as epidemic spread. There are several contributions of this thesis : a computational epidemic model combining PageRank model and models based on pretopology, use of multiple implicitly restarted Arnoldi method and proposition of its parallel implementation for PageRank computation, and proposition of a parallel social graph generator allowing to test the above methods over very large scale synthetic data.

Modeling of epidemic spread benefits a lot from network research to understand infection evolution in a population. PageRank-like model could shed light on understanding the impact of social network structure on propagation of virus and could help identifying individuals most likely to spread the disease. Besides, parallelism makes the model computationally adavantageous over traditional approaches.

8.1 Pretopology as a tool for modelling social networks

A social network is composed from different types of links between individuals, which implies that one graph is not sufficient to cover the complexity of the real world. So we propose a new mathematical formalism for modelling social networks, dealing with families of graphs. This formalism

enables to study topological properties of a network by means of a generalization of mathematical topology, called pretopology [13].

8.1.1 Basics on pretopoloy

Let's consider a non empty set E. We define a function $a(.)$ from the power set $\mathcal{P}(E)$ of E into itself such as :
(P_1) $a(\emptyset) = \emptyset$
(P_2) $\forall A, A \subset E, A \subset a(A)$
$a(.)$ is called pseudo-closure on E. Then, the couple $(E, a(.))$ is called a "pretopological space". As in topology, we can define the interior function $i(.)$ by putting : $\forall A \subset E, i(A) = (a(A^c))^c$ where A^c denotes the complementary of A in E. Thus, related to usual concepts of topology, we only keep two first properties of the topological closure mapping.

Different pretopological spaces

A basic pretopological space $(E, a(.))$ is such as :
(P_1) $a(\emptyset) = \emptyset$
(P_2) $\forall A, A \subset E, A \subset a(A)$

$\mathcal{V}$ type space

Let us consider the following axiom :
(P_3) $\forall A, A \subset E, \forall B, B \subset E, A \subset B \Rightarrow a(A) \subset a(B)$

Definition 1. *if $a(.)$ fulfils P_1, P_2 and P_3, we say that $(E, a(.))$ is a $\mathcal{V}$ type space.*

In this case, the concept of neighbourhood becomes a quite interesting one. In pretopology, this concept is defined in the same way as in topology.

Definition 2. *Let $(E, a(.))$ be a $\mathcal{V}$ type space. Any subset V of E is said a neighbourbood of x, $x \in E$ if and only if $x \in i(V)$.*

However, in pretopology, the family $\mathcal{V}(x)$ of neighbourhood of any x does not fulfils the same properties. In fact, generally speaking, the only

thing we can say is that $\mathcal{V}(x)$ is a prefilter of subsets of E, i.e. :
$\forall x \in E, \emptyset \notin \mathcal{V}(x)$
$\forall x \in E, \forall V \in \mathcal{V}(x), \forall W \subset E, V \subset W \Rightarrow W \in \mathcal{V}(x)$
In particular, generally speaking, we cannot say :
$\forall x \in E, \forall V \in \mathcal{V}(x), \forall W \subset E, V \cap W \in \mathcal{V}(x)$

$\mathcal{V}_D$ type space

Let us consider the following axiom :
$(P_4)\ \forall A, A \subset E, \forall B, B \subset E, a(A \cup B) = a(A) \cup a(B)$

Definition 3. *if $a(.)$ fulfils P_1, P_2 and P_4, we say that $(E, a(.))$ is a $\mathcal{V}_D$ type space.*

Obviously, if $(E, a(.))$ is a $\mathcal{V}_D$ type space, it also is a $\mathcal{V}$ type space. And the family of neighbourhoods of any x in E is a filter, i.e. $\mathcal{V}(x)$ is a prefilter and satisfies the following property :
$\forall V \in \mathcal{V}(x), \forall W \in \mathcal{V}(x), V \cap W \in \mathcal{V}(x)$.

$\mathcal{V}_s$ type space

Let us consider the following axiom :
$(P_5)\ \forall A, A \subset E, a(A) = \bigcup_{x \in A} a(\{x\})$

Definition 4. *if $a(.)$ fulfils P_1, P_2 and P_5, we say that $(E, a(.))$ is a $\mathcal{V}_s$ type space.*

Clearly, if $(E, a(.))$ is a $\mathcal{V}_s$ type space, it also is a $\mathcal{V}_D$ type space and then a $\mathcal{V}$ type space. Moreover, the family of neighbourhoods of x satisfies the following property : $\bigcap_{V \in \mathcal{V}(x)} \in \mathcal{V}(x)$.
This property is interesting from a computational point of view as it implies that it is sufficient to compute pseudoclosure of singletons of E to get pseudoclosure of any subset of E.

In fact, pretopology can be viewed as a generalization of graph theory as well as generalization of topology. Let us consider a finite set E, endowed with a family of binary relationships $(\mathcal{R}_i)_{i \in \{1, \cdots, p\}}$. We suppose these relationships are reflexive ones. Let us define : $\forall A, A \in \mathcal{P}(E), a(A) = \{x \in$

$E|\forall i = 1, \cdots, p, \Gamma_i(x) \cap A \neq \emptyset\}$ where $\Gamma_i(x) = \{y \in E|xR_iy\}$. Then it is obvious that $a(.)$ verifies properties of a $\mathcal{V}$ type pretopological space. The pretopological space $(E, a(.))$ can be analyzed regarding to "topological" properties : closed subset, open subset, properties related to connectivity and so on.

We thus get a structure on E, via $a(.)$, which enables characterizing some properties by taking into account the whole family of relationships defined on E.

Also, it must be noticed that there exists a great variety for defining a pseudo-closure $a(.)$ on E from the family $(\mathcal{R})_{i\in\{1,\cdots,p\}}$.

8.1.2 Basics on random correspondences

Up to now, we have given a mathematical model for modeling phenomenons accuring in a finite set E endowed with a family $(\mathcal{R})_{i\in\{1,\cdots,p\}}$ of reflexive binary relationships. However, what if each of these relationships model relations between agents accuring more or less at random ? In that case, pretopology must be completed by using concepts on random correspondances to lead what is called stochastic pretopology. In this subsection, we give basic concepts related to random correspondances in the specific context of $\mathbb{R}^n$.

Definition I

Let us consider a measurable space $(\Omega, \mathcal{A})$ and a correspondence Γ into $\mathbb{R}^n$. *Gamma* is assumed a non empty compact valued correspondence. We also suppose that Ω is locally compact and $\mathcal{A}$ is defined as follows :
Starting from $\mathcal{B}$ the $\sigma-$algebra of borelians of Ω, we complete it to obtain $\mathcal{B}_p$ (p being the probability on $\mathcal{B}$) and we consider :
$\mathcal{A} = \{A, A \subset \Omega|A \cap K \in \mathcal{B}_p, \forall K \in \mathcal{K}(\mathbb{R}^n)\}$
where $\mathcal{K}(\mathbb{R}^n)$ is the family of compacts of Ω.

Definition 5. *Let us consider $(\Omega, \mathcal{A})$ a measurable space, $\Gamma -\!\!\!\nearrow\!\!\!\searrow \mathbb{R}^n$. We say that Γ is measurable in the sense I if and only if for all F, closed subset of $\mathbb{R}^n$,*
$A = \{\omega \in \Omega : \Gamma(\omega) \cap F \neq \emptyset\} \in \mathcal{A}$.

This definition can also be rewritten as follows : for any O open subset of $\mathbb{R}^n$,
$B = \{\omega \in \Omega : \Gamma(\omega) \subset O\} \in \mathcal{A}$.

Definition II

In this subsection, $G(\Gamma)$ denotes the graph of the correspondence Γ, i.e. $G(\Gamma) = \{(\omega, x) \in \Omega \times \mathscr{R}(\mathbb{R}^n) | x \in \Gamma(\omega)\}$ and $\mathcal{B}_n$ denotes the $\sigma-$algebra of borelians of $\mathbb{R}^n$.

Definition 6. *Let us consider $(\Omega, \mathcal{A})$ a measurable space, $\Gamma -\!\!\!\nearrow\!\!\!\searrow \mathbb{R}^n$. We say that Γ is measurable in the sense II if and only if $G(\Gamma) \in \mathcal{A} \otimes \mathcal{B}_n$.*

Definition III

As correspondences are compact valued, a third proposition can be proposed. Let us consider the following families :
$\mathcal{U}^w = \{K.K \in \mathcal{K}(\mathbb{R}^n) | K \cap U \neq \emptyset, \forall U \in \mathcal{O}(\mathbb{R}^n)\}$
$\mathcal{U}^s = \{K.K \in \mathcal{K}(\mathbb{R}^n) | K \subset U, \forall U \in \mathcal{O}(\mathbb{R}^n)\}$
where $\mathcal{O}(\mathbb{R}^n)$ denotes the family of open subsets of $\mathbb{R}^n$.

These two families define a topology $\mathcal{T}$ on $\mathcal{K}(\mathbb{R}^n)$ which is equivalent to the topology generated by the Hausdorff metric. Thus $\mathcal{K}(\mathbb{R}^n)$ also is a separable metric space. Let us consider Σ_n the $\sigma-$algebra of borelians of $\mathcal{K}(\mathbb{R}^n)$. Γ can be considered not as a correspondence from Ω into $\mathbb{R}^n$ but as a function from Ω into $\mathcal{K}(\mathbb{R}^n)$. it is possible to consider for Γ, the usual definition of measurability for functions.

Definition 7. *Let us consider $(\Omega, \mathcal{A})$ a measurable space, $\Gamma \mapsto \mathbb{R}^n$. We say that Γ is measurable in the sense III if and only if $\forall A \in \Sigma_n, \Gamma^{-1}(A) \in \mathcal{A}$, where $\Gamma^{-1}(A) = \{\omega \in \Omega | \Gamma(\omega) \in A\}$.*

We get the following result :

Theorem 1. *Let us consider $(\Omega, \mathcal{A})$ a complete measurable space, Ω being locally compact, $\Gamma \mapsto \mathbb{R}^n$. The three definitions are equivalent ones.*

Proof. First, let us prove the equivalence of definition I and II. For that, we use the following result.

Let $(\Omega, \mathcal{A}, p)$ *a complete measurable space, let* E *a complete metric separable space and* Γ *a correspondence defined upon* Ω*, valued in the family of closed subsets of* E*, then*

$G(\Gamma) \in \mathcal{A} \otimes \sigma(E) \Leftrightarrow \{\omega \in \Omega | \Gamma(\omega) \cap F \neq \emptyset\} \in \mathcal{A}$

where $\sigma(E)$ denotes the $\sigma-$algiebra of borelians of E. As $\mathbb{R}^n$ and Γ verify properties of this result, definitions I and II are equivalent.

Now, let us suppose Γ measurable according to definition III and let us consider, for any closed subset F of $\mathbb{R}^n$, the set $A = \{\omega | \Gamma(\omega) \cap F \neq \emptyset\}$. $A = \{\omega \in \Omega | \Gamma(\omega) \subset F^c\}^c$, where F^c denotes the complementary of F in $\mathbb{R}^n$. F^c is an open subset of $\mathbb{R}^n$ and $A = \Gamma^{-1}((F^c)^s)$. As Γ is measurable according to definition III, $\Gamma^{-1}((F^c)^s) \in \mathcal{A}$ and Γ is measurable according to definition I. To prove that definition I implies definition III, it is sufficient using the following result :

$(\Omega, \mathcal{A})$ *is a measurable space,* Ω *is locally compact, if* f *is a function from* $(\Omega, \mathcal{A})$ *in* E*,* E *is a separable metric space endowed with its borelians, then* f *measurable is equivalent to* f $p-$*measurable.*

This result is applied to Γ as a function from $(\Omega, \mathcal{A})$ into $\mathcal{K}(\mathbb{R}^n)$. This leads to result. □

By combining results provided by pretopology and by random correspondance theory, we obtain the model of stochastic pretopology which leads to a formal definition of stochastic networks as in the following subsection.

8.1.3 Basics on stochastic networks

The pretopology does not take into account exogenous random factors. Since we are in the context of set-approach (adherence is a function defined by sets), we propose to use the random sets to build a stochastic model, named stochastic pretopology. The objective is to be able to explore the dynamic of complex social networks.

Given a finite population E, with n individuals, given a probability space $(\Omega, \mathcal{A}, p)$, we consider the following operator $\mathcal{R}(.)$ defined as :

$\mathcal{R}(.) : (\Omega, \mathcal{A}, p) \mapsto \mathcal{R}(E)$

where $\mathcal{R}(E)$ denotes the set of all binary relationships (networks) on E. By

definition, $\mathcal{R}(E)$ is a family of subsets of $E \times E$. So, we assume that $\mathcal{R}(E)$ is random set, i.e. a measurable correspondence from $(\Omega, \mathcal{A}, p)$ into $E \times E$.

Definition 8. *$\mathcal{R}(E)$ is called a stochastic network operator.*

Definition 9. *We define a network as a family of $\{\mathcal{R}\}_{i=1,\cdots,p}$ of binary relationships on E.*

Without loss of generality, we can assume that relationships $\mathcal{R}_i$ are reflexive ones.

Pretopology on a stochastic network

Basic concepts of pretopology and of random correspondences are respectively presented in the precedent paragraph. In this section, we define a pretopological structure on a stochastic network and list its basic properties.

Let E be a finite set. Let $\mathcal{R}_i$ a stochastic network defined on a probability space $(\Omega, \mathcal{A}, p)$. let us consider, for any subset of A of E, the function $a(.,.)$ defined by :

$a(.,.) : (\Omega, \mathcal{A}, p), \mathcal{P}(E) \mapsto \mathcal{P}(E)$ such that

$a(\omega, A) = \{x \in E | \forall i, \Gamma_i(\omega, x) \cap A \neq \emptyset\}$

where $\Gamma_i(\omega, x) = \{y \in E | x\mathcal{R}_i(\omega)y\}$. Then :

Theorem 2. *$(E, a(.,.))$ with $a(.,.)$ as previously defined is a stochastic pretopological space.*

Proof. Let $x \in E$, we put $\phi_i(\omega, x) = 1$ if $\Gamma_i(\omega, x) \cap A \neq \emptyset$ and $\phi_i(\omega, x) = 0$ otherwise. So, $\{x \in E | \Gamma_i(\omega, x) \cap A \neq \emptyset\}$ is a random correspondence for and $A \subset E$. As for any $A \subset E$, $a(\omega, A)$ is a finite union of $a_i(\omega, A)$, the correspondence $\omega \to a(\omega, A)$ is a random correspondence. So, $(E, a(.,.))$ is a stochastic pretopology. □

According to definition of $a(.,.)$ we have $x \in a(\omega, A) \Leftrightarrow \forall i, \Gamma_i(\omega, x) \cap A \neq \emptyset$. In other words, $x \in a(\omega, A) \Leftrightarrow \forall i, \exists y \in A, x\mathcal{R}_i(\omega)y$. So $x \in a(\omega, A)$ means that, for any kind of relationship, there exists a link between x and at least one element of A. $\| a(\omega, A) \|$ then is a good indicator of influence of A in the network in the sense that the greater it is, the greater is the number

of elements outside A linked to, at least, one element of A, whatever the nature of the link. By definition (see Appendix A), $\forall A \subset E, \forall \omega \in \Omega, i(\omega, A) = (a(\omega, A^c))^c$. Then, $i(\omega, A)$ is the subset of elements of A for which it is possible to find out at least one relationship such that all children of x are in A for which we can find out at least one relationship for which their children are in A. This leads us to the following definition :

Definition 10. *We call pseudoclosure ratio, the quantity*

$$pcr(\omega, A) = \frac{\parallel a(\omega, A) \parallel}{\parallel A \parallel} \tag{8.1}$$

We call interior ratio the quantity

$$ir(\omega, A) = \frac{\parallel i(\omega, A) \parallel}{\parallel A \parallel} \tag{8.2}$$

Then

Theorem 3. $\forall A \subset E, \omega \rightarrow pcr(\omega, A)$ *is a random variable.* $\forall A \subset E, \omega \rightarrow ir(\omega, A)$ *is a random variable.*

Proof. It is sufficient to note that $\parallel a(\omega, A) \parallel = \sum_{x \in E} 1_{a(\omega, A)}(x)$ and that, $a(., A)$ is a random correspondence, $1_{a(\omega, A)}(x)$ is a random variable.

Obviously, there is a strong link between stochastic networks and random graphs. In fact, in cases where there is only one relationship in the network, we are faced to a random graph. The difference between our approach and usual approaches is that pretopology and stochastic pretopology provide us with a topological analysis of the network based on concepts fully adapted to discrete spaces. Another advantage is to be able to compute statistics, to perform statistical analysis on indicators such as $pcr(.,.)$ and $ir(.,.)$ and to use new concepts of connectivity defined in the framework of pretopology.

As shown in the Fig.8.1, the four zones correspond to :
1. $pcr(\omega, A)$ is close to 1 and $ir(\omega, A)$ is close to 1. The set A and its complementary suffers little from each other. This characterizes a subset of "isolated" network.
2. $pcr(\omega, A)$ is close to 1 and $ir(\omega, A)$ is close to 0. The set A suffers little from its complementary while its complementary suffers a lot from A.

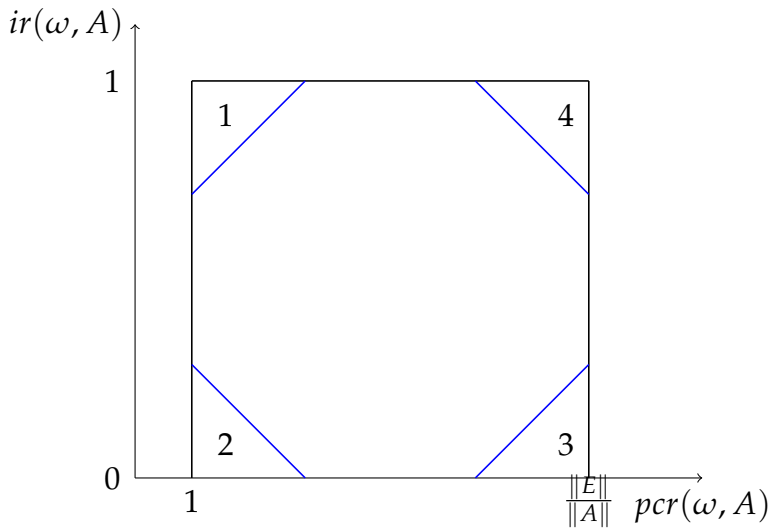


FIGURE 8.1: Pseudoclusure ratio and interior ratio relationships

3. $pcr(\omega, A)$ is close to its maximum and $ir(\omega, A)$ is close to 0. The set A generates a lot of influence and it suffers a lot from its complementary as well.
4. $pcr(\omega, A)$ is close to 1 and $ir(\omega, A)$ is close to 1. The set A generates little influence while it suffers a lot from its complementary.

Noticed that these parameters are among the list of parameters in pretopology, which generalize that computed in the theory of random graphs. In addition, stochastic pretopology can be generalized to apply to hypergraphs, whose vertex is a set of nodes. The parameters, which are initially calculated over the nodes of a graph, are now calculated over these sets of nodes. This generalization can be, for example, used to study the community structure in a society. As a result, its application to the propagation of infectious disease is obvious.

From a computational point of view, modelling based on stochastic pretopology can be implemented by a group of stochastic networks. The propagation of infectious disease on each network can be modelled as a Markov chain and can be computed within the framework of the actual thesis.

8.2 Asynchronous MIRAM

In this section, we focus on the possible improvement of computational approach to compute the rightmost eigenpairs on stochastic networks. It is known that PageRank can be computed using numerical methods based on sparse MVP and we propose to use a parallel "multiple IRAM" algorithm (MIRAM). From the Experiment 5.4, we see that MIRAM is promising especially for big damping factors. The parallel MIRAM implementation takes into account the scale-free structure of underlying networks and is scalable to handle memory and computation issues arising from very large networks such as *twitter* and *yahoo* network. From our tests, we have obtained a speedup of 27$\times$ compared to sequential solver. Additionally, it is found in Experiment 7.5.3 that thick restart could help accelerate the convergence of the method even under constraints caused by storage.

MIRAM (with nested or non nested subspaces) has a great potential for large coarse grain parallelism among its Arnoldi factorizations. In fact, the restarting vector can be made different among processors. In this case, the whole orthogonal basis of the chosen subspace should also be sent to processors. Consequently, the computation in different subspaces of MIRAM will be asynchronous. This coarse grain parallelism is fault tolerant since any loss of an IRAM process during MIRAM execution does not interfere with its termination. All these properties show that MIRAM is well suitable for large scale distributed computational environments.

Here we present a possible implementation approach of the asynchronous MIRAM. The idea is to add a central server (can be a group of nodes) that accumulates eigen-information from each MIRAM instance. The sever will then respond to MIRAM instance for available m_{best}. This idea is des-

cribed in Algorithm 5.

Algorithm 5: Asynchronous MIRAM

Data: $(A, V_{m_i}, H_{m_i}, f_{m_i})$ with $AV_{m_i} = V_{m_i}H_{m_i} + f_{m_i}e_{m_i}^T$ the m_i-steps Arnoldi factorization $(i = 1, 2, \ldots, \ell)$. Coarse grain parallelism is used to distribute different subspace m_i. Fine grain parallelism is used to distribute the origin matrix A and different orthogonal basis V_{m_i} and f_{m_i}. Noticing that every triplet $(H_{m_i}, V_{m_i}, f_{m_i})$ should be sent to the server at the beginning.

for *each subspace size* m_i **do**

1. (server)
   - Compute $\sigma(H_{m_i})$ and their associated eigenvectors (for $i = 1, \ldots, \ell$)
   - Compute residual norms. If convergence in one of subspaces then stop.
   - Select the best results in these subspaces and the associated best subspace size m_{best}. Set $m = m_{best}$, $H_m = H_{m_{best}}$ and $V_m = V_{m_{best}}$, $f_m = f_{m_{best}}$.
   - Select set of $p = m - k$ shifts $(\mu_1^{(m)}, \ldots, \mu_p^{(m)})$, based upon $\sigma(H_m)$ or perhaps other information and set $q^T \leftarrow e_m^T$.
2. (MIRAM instance)

   if *Available* m_{best} *from server* **then**

   for $j = 1, \ldots, p$ **do**
   - Factor $[Q_j, R_j] = qr(H_m - \mu_j^{(m)} I)$;
   - $H_m \leftarrow Q_j^H H_m Q_j$; $V_m \leftarrow V_m Q_j$,
   - $q \leftarrow q^H Q_j$

   else

   done
3. Set $f_k \leftarrow v_{k+1}\widehat{\beta}_k + f_m q(k)$, $V_k \leftarrow V_m(1:n, 1:k)$, $H_k \leftarrow H_m(1:k, 1:k)$
4. Beginning with the k-step Arnoldi factorization $AV_k = V_k H_k + f_k e_k^T$, apply $p_i = m_i - k$ additional steps of the Arnoldi process to obtain ℓ new m_i-step Arnoldi factorization $AV_{m_i} = V_{m_i}H_{m_i} + f_{m_i}e_{m_i}^T$ (for $i = 1, \ldots, \ell$).

The analysis of intra and inter Arnoldi factorizations parallelism in the asynchronous version of MIRAM can be the subject of a future work. Moreover, we intend to expand the proposed epidemic model by including various indicators of epidemic spread, such as characteristics of individuals as well as that of viruses, spreading timestamps, etc.

Bibliographie

[1] BA Data Sets. http://topology.eecs.umich.edu/data.html.

[2] SNAP Data Sets. http://snap.stanford.edu/data/index.html.

[3] Modeling Community Containment for Pandemic Influenza : A Letter Report. Technical report, 2006.

[4] Maksudul Alam, Maleq Khan, and Madhav V. Marathe. Distributed-memory parallel algorithms for generating massive scale-free networks using preferential attachment model. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 91 :1–91 :12, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2378-9. doi : 10.1145/2503210.2503291. URL http://doi.acm.org/10.1145/2503210.2503291.

[5] R. Albert, H. Jeong, and A. Barabási. Error and attack tolerance of complex networks. *Nature*, 406 :378–382, 2000.

[6] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1989. ISBN 0-8053-0177-1.

[7] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Q. Appl. Math*, 9(17) :17–29, 1951.

[8] A. Bai, D. Day, J. Demmel, and J. Dongarra. A Test Matrix Collection for Non-Hermitian Eigenvalue Problems. Technical report, Knoxville, TN, USA, 1997.

[9] A. H. Baker, E. R. Jessup, and Tz. V. Kolev. A Simple Strategy for Varying the Restart Parameter in GMRES(m). *J. Comput. Appl. Math.*,

230(2) :751–761, August 2009. ISSN 0377-0427. doi : 10.1016/j.cam.2009.01.009. URL http://dx.doi.org/10.1016/j.cam.2009.01.009.

[10] A. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439) :509–512, October 1999.

[11] A. Barabási and R. Albert. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74 :47–97, 2002.

[12] Vladimir Batagelj and Ulrik Brandes. Efficient generation of large random networks. *Phys. Rev. E*, 71 :036113, Mar 2005. doi : 10.1103/PhysRevE.71.036113. URL http://link.aps.org/doi/10.1103/PhysRevE.71.036113.

[13] Z. Belmandt. *Manuel de prétopologie*. Hermès, 1993.

[14] Pavel Berkhin. A survey on pagerank computing. *Internet Mathematics*, 2 :73–120, 2005.

[15] K. Bisset, J. Chen, X. Feng, VS. Anil Kumar, and M. Marathe. EpiFast : A fast algorithm for large scale realistic epidemic simulations on distributed memory systems. In *Proceedings of 23rd ACM International Conference on Supercomputing (ICS'09)*, pages 430–439, 2009.

[16] Keith Bisset. Urgent computing for interaction based socio-technical simulations. Invited presentation to Argonne National Laboratory, April .

[17] Erik G. Boman, Ümit V. Çatalyürek, Cédric Chevalier, and Karen D. Devine. The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing : Partitioning, ordering and coloring. *Scientific Programming*, 20(2) :129–150, 2012.

[18] Stefan Bornholdt and Thimo Rohlf. Topological evolution of dynamical networks : Global criticality from local dynamics. *Phys. Rev. Lett.*, 84 :6114–6117, Jun 2000. doi : 10.1103/PhysRevLett.84.6114. URL http://link.aps.org/doi/10.1103/PhysRevLett.84.6114.

[19] Mic Bowman, Saumya K. Debray, and Larry L. Peterson. Immunization of networks with community structure. *New Journal of Physics*, 11 (123018), 2009.

[20] Jeremy T. Bradley, Douglas de Jager, William J. Knottenbelt, and Aleksandar Trifunovic. Hypergraph Partitioning for Faster Parallel PageRank Computation. In *EPEW'05, Proceedings of the 2nd European Performance Evaluation Workshop*, volume 3670 of *Lecture Notes in Computer Science*, pages 155–171, September 2005. URL http://pubs.doc.ic.ac.uk/hypergraph-fast-pagerank/.

[21] Andrei Z. Broder, Ronny Lempel, Farzin Maghoul, and Jan Pedersen. Efficient pagerank approximation via graph aggregation. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, WWW Alt. '04, pages 484–485, New York, NY, USA, 2004. ACM. ISBN 1-58113-912-8. doi : 10.1145/1013367.1013537. URL http://doi.acm.org/10.1145/1013367.1013537.

[22] Helene Broutin, Eric Elguero, Francois Simondon, and Jean-Francois Guegan. Spatial dynamics of pertussis in a small region of Senegal. *Proceedings of The Royal Society B : Biological Sciences*, 271 :2091–2098, 2004. doi : 10.1098/rspb.2004.2847.

[23] Kurt Bryan and Tanya Leise. The $25,000,000,000 Eigenvector : The linear Algebra behind Google. *SIAM Rev.*, 48(3) :569–581, March 2006. ISSN 0036-1445. doi : 10.1137/050623280. URL http://dx.doi.org/10.1137/050623280.

[24] Umit Catalyurek and Cevdet Aykanat. Hypergraph-Partitioning-Based Decomposition for Parallel Sparse-Matrix Vector Multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10(7) :673–693, July 1999. ISSN 1045-9219. doi : 10.1109/71.780863. URL http://dx.doi.org/10.1109/71.780863.

[25] Dennis L. Chao, M. Elizabeth Halloran, Valerie J. Obenchain, and Ira M. Longini, Jr. FluTE, a Publicly Available Stochastic Influenza Epidemic Simulation Model. *PLoS Comput Biol*, 6(1) :e1000656, 01 2010. doi : 10.1371/journal.pcbi.1000656.

[26] D. P. Chassin and C. Posse. Evaluating North American electric grid reliability using the Barabási Albert network model. *Physica A*, 355 : 667–677, 2005.

[27] Fan R. K. Chung, Paul Horn, and Alexander Tsiatas. Distributing Antidote Using PageRank Vectors. *Internet Mathematics*, 6(2) :237–254, 2009.

[28] R. Cohen, S. Havlin, and D. Ben-Avraham. Efficient Immunization Strategies for Computer Networks and Populations. *Phys. Rev. Lett.*, 91(247901), 2003.

[29] Reuven Cohen, Keren Erez, Daniel ben Avraham, and Shlomo Havlin. Resilience of the Internet to Random Breakdowns. *Phys. Rev. Lett.*, 85 :4626–4628, Nov 2000. doi : 10.1103/PhysRevLett.85.4626. URL http://link.aps.org/doi/10.1103/PhysRevLett.85.4626.

[30] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization. *Mathematics of Computation*, 30(136) :772–795, 1976. ISSN 00255718. doi : 10.2307/2005398. URL http://dx.doi.org/10.2307/2005398.

[31] K. Dookhitram, R. Boojhawon, and M. Bhuruth. A New Method for Accelerating Arnoldi Algorithms for Large Scale Eigenproblems. *Math. Comput. Simul.*, 80(2) :387–401, October 2009. ISSN 0378-4754. doi : 10.1016/j.matcom.2009.07.009. URL http://dx.doi.org/10.1016/j.matcom.2009.07.009.

[32] I. S. Duff and J. A. Scott. Computing Selected Eigenvalues of Sparse Unsymmetric Matrices Using Subspace Iteration. *ACM Trans. Math. Softw.*, 19(2) :137–159, June 1993. ISSN 0098-3500. doi : 10.1145/152613.152614. URL http://doi.acm.org/10.1145/152613.152614.

[33] Nadav Eiron, Kevin S. McCurley, and John A. Tomlin. Ranking the web frontier. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 309–318, New York, NY, USA, 2004. ACM.

ISBN 1-58113-844-X. doi : 10.1145/988672.988714. URL http://doi.acm.org/10.1145/988672.988714.

[34] Nahid Emad. Parallel numerical linear algebra. chapter Mapping Strategies in Data Parallel Programming Models ; the Projection Methods, pages 57–70. Nova Science Publishers, Inc., Commack, NY, USA, 2001. ISBN 1-59033-114-1. URL http://dl.acm.org/citation.cfm?id=644383.644387.

[35] Nahid Emad, Serge Petiton, and Guy Edjlali. Multiple explicitly restarted Arnoldi method for solving large eigenproblems. *SIAM Journal on scientific computing SJSC, Volume 27, Number*, 27 :253–277, 2005.

[36] M. Embree. The Tortoise and the Hare Restart GMRES. *SIAM Review*, 45(2) :259–266, 2003.

[37] P. Erdös and A. Rényi. On the evolution of random graphs. Publ. Math. Inst. Hung. Acad. Sci., 1960.

[38] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On Power-law Relationships of the Internet Topology. *SIGCOMM Comput. Commun. Rev.*, 29(4) :251–262, August 1999. ISSN 0146-4833. doi : 10.1145/316194.316229. URL http://doi.acm.org/10.1145/316194.316229.

[39] S.A.S. Fazeli. *Stratégies de redémarrage des méthodes itératives d'algèbre linéaire pour le calcul global*. 2005. URL http://books.google.be/books?id=FH10twAACAAJ.

[40] Seyed Abolfazl Shahzadeh Fazeli, Nahid Emad, and Zifan Liu. A key to choose subspace size in implicitly restarted Arnoldi method. Springer Journal of Numerical Algorithms, accepted.

[41] F.Chatelain. *Eigenvalues of matrices*. Wiley, 1993.

[42] Jacques Ferber. *Multi-Agent Systems : An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1999. ISBN 0201360489.

[43] Michael J. Flynn. Some computer organizations and their effectiveness. *IEEE Trans. Comput.*, 21(9) :948–960, September 1972. ISSN 0018-9340. doi : 10.1109/TC.1972.5009071. URL http://dx.doi.org/10.1109/TC.1972.5009071.

[44] Sebastian Funk and Vincent A. A. Jansen. Interacting epidemics on overlay networks. *Phys. Rev. E*, 81 :036118, Mar 2010. doi : 10.1103/PhysRevE.81.036118. URL http://link.aps.org/doi/10.1103/PhysRevE.81.036118.

[45] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12) :7821–7826, June 2002. ISSN 1091-6490. doi : 10.1073/pnas.122653799. URL http://dx.doi.org/10.1073/pnas.122653799.

[46] David Gleich, Leonid Zhukov, and Pavel Berkhin. Fast parallel PageRank : a linear system approach. Technical Report L-2004-038, Yahoo ! Research Labs, 2004.

[47] G.H. Golub and C. Greif. An Arnoldi-type algorithm for computing PageRank. *BIT Numerical Mathematics*, 46(4) :759–771, 2006.

[48] Thilo Gross and Bernd Blasius. Adaptive coevolutionary networks : a review. *J. R. Soc. Interface*, 5(20) :259–271, 2008. doi : 10.1098/rsif.2007.1229.

[49] Thilo Gross, Carlos J. Dommar D'Lima, and Bernd Blasius. Epidemic dynamics on an adaptive network. *Phys. Rev. Lett.*, 96 :208701, May 2006. doi : 10.1103/PhysRevLett.96.208701. URL http://link.aps.org/doi/10.1103/PhysRevLett.96.208701.

[50] Taher H. Haveliwala, Sepandar D. Kamvar, and Ar D. Kamvar. The Second Eigenvalue of the Google Matrix. Technical Report 2003-20, Stanford InfoLab, 2003.

[51] Bruce Hendrickson and Robert Leland. The Chaco User's Guide : Version 2.0. Technical Report SAND94–2692, Sandia National Lab, 1994.

[52] Michael Heroux, Roscoe Bartlett, Vicki Howle Robert Hoekstra, Jonathan Hu, Tamara Kolda, Richard Lehoucq, Kevin Long, Roger Pawlowski, Eric Phipps, Andrew Salinger, Heidi Thornquist, Ray Tuminaro, James Willenbring, and Alan Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[53] Petter Holme and Gourab Ghoshal. Dynamics of networking agents competing for high centrality and low degree. *Phys. Rev. Lett.*, 96 : 098701, Mar 2006. doi : 10.1103/PhysRevLett.96.098701. URL http://link.aps.org/doi/10.1103/PhysRevLett.96.098701.

[54] Ilse C. F. Ipsen and Teresa M. Selee. PageRank Computation, with Special Attention to Dangling Nodes. *SIAM J. Matrix Anal. Appl.*, 29(4) : 1281–1296, December 2007. ISSN 0895-4798. doi : 10.1137/060664331. URL http://dx.doi.org/10.1137/060664331.

[55] Sepandar Kamvar, Taher Haveliwala, Chris Manning, and Gene Golub. Extrapolation methods for accelerating pagerank computations. In *Twelfth International World Wide Web Conference (WWW 2003)*, 2003. URL http://ilpubs.stanford.edu:8090/865/.

[56] George Karypis and Vipin Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.*, 20(1) :359–392, December 1998. ISSN 1064-8275. doi : 10.1137/S1064827595287997. URL http://dx.doi.org/10.1137/S1064827595287997.

[57] Matt J Keeling and Ken T.D Eames. Networks and epidemic models. *J. R. Soc. Interface*, 2 :295–307, 2005.

[58] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media ? In *WWW '10 : Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi : http://doi.acm.org/10.1145/1772690.1772751.

[59] Amy N. Langville and Carl D. Meyer. *Google's PageRank and Beyond : The Science of Search Engine Rankings*. Princeton University Press, Princeton, NJ, USA, 2006. ISBN 0691122024.

[60] Chris P. Lee, Gene H. Golub, and Stefanos A. Zenios. A Fast Two-Stage Algorithm for Computing PageRank and Its Extensions. Technical report, Stanford University, 2004. URL http://www-sccm.stanford.edu/pub/sccm/sccm03-15_2.pdf.

[61] R. B. Lehoucq. Analysis and Implementation of an Implicitly Restarted Iteration. *PhD thesis, Rice University, Houston, Texas,*, 1995.

[62] R. B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM J. Matrix Anal. Appl.*, 17(4) :789–821, October 1996. ISSN 0895-4798. doi : 10.1137/S0895479895281484. URL http://dx.doi.org/10.1137/S0895479895281484.

[63] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users Guide : Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods.* SIAM, 1997.

[64] Jure Leskovec. Stanford Large Network Dataset Collection. URL http://snap.stanford.edu/data/.

[65] Jennifer Lindquist, Junling Ma, P. Driessche, and FrederickH. Willeboordse. Effective degree network disease models. *Journal of Mathematical Biology*, 62(2) :143–164, 2011. ISSN 0303-6812. doi : 10.1007/s00285-010-0331-2. URL http://dx.doi.org/10.1007/s00285-010-0331-2.

[66] Zifan Liu, Nahid Emad, Soufian Ben Amor, and Michel Lamure. Towards modeling of epidemic spread : eigenvalue computation. Preprint for publication. URL http://www.prism.uvsq.fr/rapports/bin/abstract.php?id=586.

[67] N. Madar, T. Kalisky, R. Cohen, D. Ben-Avraham, and S. Havlin. Immunization and epidemic dynamics in complex networks. *Eur. Phys. J. B.*, 38(2) :269–276, 2004.

[68] Madhav Marathe and Anil Kumar S. Vullikanti. Computational Epidemiology. *Commun. ACM*, 56(7) :88–96, July 2013. ISSN 0001-0782. doi : 10.1145/2483852.2483871. URL http://doi.acm.org/10.1145/2483852.2483871.

[69] Vincent Marceau, Pierre-André Noël, Laurent Hébert-Dufresne, Antoine Allard, and Louis J. Dubé. Adaptive networks : Coevolution of disease and topology. *Phys. Rev. E*, 82 :036116, Sep 2010. doi : 10.1103/PhysRevE.82.036116. URL http://link.aps.org/doi/10.1103/PhysRevE.82.036116.

[70] K. Maschhoff and D. Sorensen. P_ARPACK : An efficient portable large scale eigenvalue package for distributed memory parallel architectures. In *Applied Parallel Computing Industrial Computation and Optimization*, pages 478–486. 1996. doi : 10.1007/3-540-62095-8_51. URL http://dx.doi.org/10.1007/3-540-62095-8_51.

[71] Joel C. Miller and James M. Hyman. Effective vaccination strategies for realistic social networks. *Physica A : Statistical Mechanics and its Applications*, 386(2) :780–785, 2007.

[72] C. Moore and M. E. J. Newman. Epidemics and percolation in small-world networks. *Phys. Rev. E.*, 61 :5678–5682, 2000.

[73] G. E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8) :114–117, April 1965. ISSN 0018-9219. doi : 10.1109/jproc.1998.658762. URL http://dx.doi.org/10.1109/jproc.1998.658762.

[74] Ronald B. Morgan. On restarting the Arnoldi method for large non-symmetric eigenvalue problems. *Mathematics of Computation*, 65 : 1213–1230, 1996.

[75] Kentaro Moriya and Takashi Nodera. The DEFLATED-GMRES(m, k) method with switching the restart frequency dynamically. 7(7–8) : 569–584, October/December 2000. ISSN 1070-5325 (print), 1099-1506 (electronic). URL http://www3.interscience.wiley.com/cgi-bin/

abstract/73505474/START;http://www3.interscience.wiley.com/cgi-bin/fulltext?ID=73505474&PLACEBO=IE.pdf.

[76] M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Lett.*, 89 (208701), 2002.

[77] M. E. J. Newman. Spread of epidemic disease on networks. *Phys. Rev. E.*, 66(016128), 2002.

[78] M. E. J. Newman. The spread of epidemic disease on networks. *Phys. Rev. E*, 66(016128), 2002.

[79] M. E. J. Newman. The structure and function of complex networks. *SIAM Rev.*, 45 :167–256, 2003.

[80] M. E. J. Newman. *Random graphs as models of networks*, pages 35–68. Wiley-VCH Verlag GmbH and Co. KGaA, 2005. ISBN 9783527602759. doi : 10.1002/3527602755.ch2. URL http://dx.doi.org/10.1002/3527602755.ch2.

[81] M. E. J. Newman and D. J. Watts. Renormalization group analysis of the small-world network model. *Phys. Lett. A.*, 263 :341–346, 1999.

[82] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The Pagerank Citation Ranking : Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, 1999.

[83] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998. ISBN 0-89871-402-8.

[84] François Pellegrini. Scotch and libScotch 5.1 User's Guide, August 2008. URL http://hal.archives-ouvertes.fr/hal-00410327. 127 pages User's manual.

[85] Guy-RenÃ© Perrin and Alain Darte, editors. *The Data Parallel Programming Model : Foundations, HPF Realization, and Scientific Applications*, volume 1132 of *Lecture Notes in Computer Science*, 1996. Springer. ISBN 3-540-61736-1. URL http://dblp.uni-trier.de/db/conf/ac/data1996.html.

[86] S. Petiton, M. Sato, N. Emad, C. Calvin, M. Tsuji, and M. Dandouna. Multi level programming Paradigm for Extreme Computing. In *Supercomputing in Nuclear Applications and Monte Carlo*, October 27-31 2013.

[87] Anand S. Rao and Michael P. Georgeff. Bdi agents : From theory to practice. In *IN PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS (ICMAS-95*, pages 312–319, 1995.

[88] Garry Robins, Pip Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph (p*) models for social networks. *Social Networks*, 29(2) :173–191, May 2007. ISSN 03788733. doi : 10.1016/j.socnet.2006.08.002. URL http://dx.doi.org/10.1016/j.socnet.2006.08.002.

[89] Pastor-Satorras Romualdo and Vespignani Alessandro. Epidemic Spreading in Scale-Free Networks. *Phys. Rev. Lett.*, 86 :3200–3203, Apr 2001. doi : 10.1103/PhysRevLett.86.3200. URL http://link.aps.org/doi/10.1103/PhysRevLett.86.3200.

[90] Y. Saad. Variations on Arnoldi's Method for Computing Eigenelements of Large Unsymmetric Matrices. *Linear Algebra Applications*, (34) :269–295, 1980.

[91] Youcef Saad. Chebyshev Acceleration Techniques for Solving Nonsymmetric Eigenvalue Problems. 42(166) :567–588, April 1984. ISSN 0025-5718 (print), 1088-6842 (electronic).

[92] Youcef Saad. Numerical solution of large nonsymmetric eigenvalue problems. *COMPUT. PHYS. COMM*, 53 :71–90, 1989.

[93] S.A. Shahzadeh Fazeli, Nahid Emad, and Zifan Liu. A key to choose subspace size in implicitly restarted arnoldi method. *Numerical Algorithms*, pages 1–20, 2015. ISSN 1017-1398. doi : 10.1007/s11075-014-9954-5. URL http://dx.doi.org/10.1007/s11075-014-9954-5.

[94] Yoav Shoham. Agent-oriented programming. *Artif. Intell.*, 60(1) :51–92, March 1993. ISSN 0004-3702. doi : 10.1016/0004-3702(93)90034-9. URL http://dx.doi.org/10.1016/0004-3702(93)90034-9.

[95] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1) :357–385, jan 1992. ISSN 0895-4798. doi : 10.1137/0613025. URL http://dx.doi.org/10.1137/0613025.

[96] Danny C. Sorensen. Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations. Technical report, 1996.

[97] Andreas Stathopoulos, Yousef Saad, and Kesheng Wu. Dynamic Thick Restarting of the Davidson, and the Implicitly Restarted Arnoldi Methods. *SIAM J. Sci. Comput*, 19 :227–245, 1996.

[98] Michael Taylor, Timothy J. Taylor, and Istvan Z. Kiss. Epidemic threshold and control in a dynamic network. *Phys. Rev. E*, 85 :016103, Jan 2012. doi : 10.1103/PhysRevE.85.016103. URL http://link.aps.org/doi/10.1103/PhysRevE.85.016103.

[99] Alexei Vazquez. Spreading dynamics on heterogeneous populations : Multitype network approach. *Phys. Rev. E*, 74 :066114, Dec 2006. doi : 10.1103/PhysRevE.74.066114. URL http://link.aps.org/doi/10.1103/PhysRevE.74.066114.

[100] Bing Wang, Lang Cao, Hideyuki Suzuki, and Kazuyuki Aihara. Epidemic spread in adaptive networks with multitype agents. *Journal of Physics A : Mathematical and Theoretical*, 44(3) :035101, 2011. URL http://stacks.iop.org/1751-8121/44/i=3/a=035101.

[101] Yang Wang, Deepayan Chakrabarti, Chenxi Wang, and Christos Faloutsos. Epidemic Spreading in Real Networks : An Eigenvalue Viewpoint. In *In SRDS*, pages 25–34, 2003.

[102] D. J. Watts. *Small worlds : the dynamics of networks between order and randomness*. Princeton University Press, Princeton, 1999.

[103] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393 :440–442, 1998.

[104] S. Hoya White, A. Martin del Rey, and G. Rodríguez Sánchez. Modeling epidemics using cellular automata. *Applied Mathematics and Computation*, 186(1) :193 – 202, 2007. ISSN 0096-3003. doi : http://dx.doi.org/10.1016/j.amc.2006.06.126. URL http://www.sciencedirect.com/science/article/pii/S0096300306009295.

[105] Gang Wu and Yimin Wei. An Arnoldi-Extrapolation algorithm for computing PageRank. *Journal of Computational and Applied Mathematics*, 234(11) :3196 – 3212, 2010. ISSN 0377-0427. doi : http://dx.doi.org/10.1016/j.cam.2010.02.009. URL http://www.sciencedirect.com/science/article/pii/S0377042710000804. Numerical Linear Algebra, Internet and Large Scale Applications.

[106] Gang Wu and Yimin Wei. Arnoldi Versus GMRES for Computing PageRank : A Theoretical Contribution to Google's PageRank Problem. *ACM Trans. Inf. Syst.*, 28(3) :11 :1–11 :28, July 2010. ISSN 1046-8188. doi : 10.1145/1777432.1777434. URL http://doi.acm.org/10.1145/1777432.1777434.

[107] Gang Wu, Yan-Chun Wang, and Xiao-Qing Jin. A Preconditioned and Shifted GMRES Algorithm for the PageRank Problem with Multiple Damping Factors. *SIAM J. Scientific Computing*, 34(5), 2012.

[108] Andy Yoo and Keith W. Henderson. Parallel generation of massive scale-free graphs. *CoRR*, abs/1003.3684, 2010. URL http://arxiv.org/abs/1003.3684.