UNIVERSIDAD DE GRANADA

Peer-to-Peer Evolutionary Computation: A Study of Viability

TESIS DOCTORAL

Juan Luis Jiménez Laredo

Granada, 2010

Departamento de Arquitectura y Tecnología de Computadores

Editor: Editorial de la Universidad de Granada
Autor: Juan Luis Jiménez Laredo
D.L.: GR 3194-2010
ISBN: 978-84-693-4381-4

UNIVERSIDAD DE GRANADA

Peer-to-Peer Evolutionary Computation: A Study of Viability

Memoria presentada por

Juan Luis Jiménez Laredo

Para optar al grado de

DOCTOR EUROPEO EN INFORMÁTICA

Fdo. Juan Luis Jiménez Laredo

D. Juan Julián Merelo Guervós, Profesor Titular de Universidad y **D. Pedro A. Castillo Valdivieso**, Profesor Asociado, ambos del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada

CERTIFICAN

Que la memoria titulada: *"Peer-to-Peer Evolutionary Computation: A Study of Viability"* ha sido realizada por **Juan Luis Jiménez Laredo** bajo nuestra dirección en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada para optar al grado de Doctor en Informática.

Granada, a 26 de Abril de 2010

Fdo. Pedro A. Castillo Valdivieso	Juan Julián Merelo Guervós
Director de la Tesis	Director de la Tesis

Agradecimientos

A mis directores de tesis, Juan Julián Merelo y Pedro Castillo, principales responsables de que este escribiendo estas líneas. Es a ellos a quienes se le debería de atribuir gran parte del posible mérito de este trabajo, con sus comentarios y correcciones han hecho que mejore en calidad y rigurosidad.

A todas aquellas personas que me han ayudado a amar a la ciencia y a entusiasmarme con ella, Ben Paechter por ofrecerme la oportunidad de compartir unos meses con él en Edimburgo, Gusz Eiben y Maarten van Steen por hacerme sentir que mi trabajo tenía valor, Christian Gagné por intercambiar ideas conmigo durante innumerables comidas en Laval. A Carlos Fernandes, mucho de lo bueno que pudiera tener esta tesis, se lo debo a él. A Marco Tomassini y Paulo Novais, por haberse ofrecido desinteresadamente a revisar esta memoria. A aquellos revisores que anónimamente se han tomado la molestia de estudiar mis artículos, sus aportaciones han sido imprescindibles en mi proceso de maduración como investigador, alguno tal vez aparezca citado en la bibliografía de esta memoria, a estos autores también, gracias, son los hombros en los que se basa toda mi carrera. A los dos mejores divulgadores científicos que he tenido la suerte de conocer, Carl Sagan y Carlos Cotta. A todos mis compañeros de departamento, por intercambiar opiniones y momentos de desasosiego en numerosos desayunos, comidas, reuniones y tardes de fútbol. En especial, gracias a los miembros de geneura, esta tesis ha sido desarrollada gracias a su apoyo.

A mis padres, porque son una lección viva de que la dignidad ni se compra ni se vende, además son unos cachondos mentales. A mi hermano José, porque estaba sólo en el mundo hasta que llegó él. A mi hermano Manolo, por su fortaleza de espíritu y callado sacrificio, nunca te rompas, hay gente que nació para ser grande (aunque sea criando gallinas) y nadie dijo que fuera fácil.

A mi abuelo Juan (Juan Cano) porque todavía soy capaz de sentir la intensidad de su mirada. A mi abuela Josefa (Josefa la del loco) porque aún siento las caricias de sus manos. A mi abuelo Mariano (Luisillo el de los praos), la persona más buena y sacrificada que conozco. A mi abuela Josefa (Josefa guiñapa), porque llenas los espacios con tu encanto, guapa!

A mis tíos y primos, porque la mitad están como cabras locas y me hacen sentir que pertenezco a la familia correcta, tito, a ver cuando me compro la moto.

A Lena, porque me hizo comprender que tanto en el paraíso como en el infierno hay una temperatura de veintidós grados y sopla un poco de brisa. A todas las mujeres que he amado, porque me han enseñado a esperar con paciencia a la que voy a querer más (mientras tanto aceptamos barco).

A mis amigos, innombrables y variopintos, gracias (Joder, Manu, nombraros a todos sería una pasada tio).

A mi pueblo, Zagra, por darme identidad en un mundo anónimo.

A mis ahijadas, Alba y Sheila, porque sois lindas.

A mis padres y hermanos.

Prefacio

Los algoritmos evolutivos son un conjunto de técnicas bioinspiradas aplicadas a problemas de optimización que están basados en el proceso darwiniano de selección natural. Al igual que en la evolución de las especies, aquellos individuos (*o soluciones candidatas*) que muestran ser las más aptas, son seleccionadas preferentemente para la reproducción, de este modo, los descendientes heredarán sus genes a través del paso de las generaciones. Iterativamente, la selección actúa como un filtro para los genes y solo aquellos que pertenecen a soluciones óptimas son capaces de superar la presión selectiva y recombinarse formando soluciones de más alto orden. En analogía con este proceso, la búsqueda estocástica de los algoritmos evolutivos tiene éxito en problemas de optimización a partir del refinamiento progresivo de un conjunto de soluciones candidatas.

No obstante, en aplicaciones con un alto coste de cálculo e instancias grandes de problemas, los requisitos computacionales pueden llegar a ser tan grandes que retrasen el proceso de búsqueda de soluciones óptimas más allá de un tiempo razonable. Afortunadamente, la naturaleza de los algoritmos evolutivos es inherentemente paralela ofreciendo así una forma fácil de mejorar las propiedades de escalabilidad. La idea principal consiste en acelerar los tiempos de ejecución del algoritmo repartiendo la carga computacional de los individuos en diferentes procesadores.

En este contexto, esta tesis propone un nuevo algoritmo evolutivo paralelo (denominado modelo de Agente Evolutivo) que aprovecha las capacidades de cómputo de un entorno dinámico P2P (*del inglés Peer-to-Peer; se traduce al castellano como sistema entre pares, aunque se adopta normalmente la voz inglesa*). La motivación subyacente al algoritmo es abordar instancias grandes de problemas de optimización complejos de forma eficiente y precisa, aprovechando, para tal fin, la escalabilidad masiva de las redes de computo P2P. Por lo tanto, un correcto entendimiento de dicha plataforma es clave para el diseño eficiente del algoritmo.

Los sistemas P2P ofrecen una infraestructura paralela potente para la computación evolutiva, capaz de constituir un único computador virtual compuesto de un número de recursos potencialmente grande. Sin embargo, dicha plataforma esta desprovista de servidores centrales lo cual supone un reto a la gestión centralizada del ciclo evolutivo (tanto la selección de padres como la reproducción o la selección de los supervivientes se realiza comúnmente de forma centralizada en los algoritmos evolutivos). Para abordar dicha problemática, el modelo de Agente Evolutivo asigna a cada nodo de cómputo un individuo y adopta una estructura de población descentralizada que es definida por el protocolo P2P newscast. De esta forma, cualquier individuo del algoritmo tiene un número limitado de vecinos y el proceso de selección se restringe a la vecindad local P2P.

Además de la problemática de la descentralización de recursos, un reto a tener en cuenta para la paralelización eficiente en este tipo de redes, es que los nodos son propensos a fallos dado que los recursos computacionales son añadidos y eliminados dinámicamente, frecuentemente como consecuencia de la decisión de los usuarios que ceden libremente CPUs bajo su control. De esta forma, un algoritmo evolutivo P2P tiene que ser tolerante a fallos con respecto a la dinámica de los nodos. En este sentido, el modelo de Agente Evolutivo implementa un mecanismo de degradación grácil, siendo capaz de abordar instancias grandes de problemas de optimización a pesar de que los nodos abandonen el sistema sin otro mecanismo que el propio comportamiento emergente del modelo.

Resumiendo, un algoritmo evolutivo P2P eficiente será aquel capaz de abordar instancias grandes de problemas de optimización de forma descentralizada a pesar de que los recursos de cómputo se degraden. Por lo tanto, esta tesis se centra en el análisis de la descentralización, escalabilidad y tolerancia a fallos como problemáticas clave para poder establecer la viabilidad de este nuevo paradigma de computo evolutivo.

Con ese propósito, el modelo de Agente Evolutivo ha sido analizado empíricamente en un entorno simulado P2P donde los experimentos han sido llevados a cabo bajo diferentes escenarios usando funciones trampa como problemas de prueba. Estas funciones representan un conjunto de problemas descomponibles en funciones parciales, en los que la bondad de las soluciones es calculada sumando las distintas bondades y donde el nivel de dificultad es ajustable, lo cual posibilita observar el escalado de los tamaños de población y esfuerzos computacionales para distintos tipos y tamaños de problema.

En esta memoria de tesis se exponen de forma detallada cada uno de los aspectos previamente mencionados. A modo de resumen se describen a continuación los distintos capítulos de los que está compuesta:

Capítulo 1: Expone una introducción al resto de la memoria describiendo los principales objetivos de esta tesis. Los algoritmos evolutivos P2P son presentados como alternativa para abordar instancias grandes de problemas de optimización con requisitos de cómputo altos.

Capítulo 2: En este capítulo se revisan los modelos de algoritmos evolutivos paralos más extendidos y en particular, aquellos enfoques en la literatura relacionados con el computo evolutivo P2P. Además, se proporciona una descripción del efecto que las poblaciones estructuradas juegan en la presión selectiva de los algoritmos evolutivos.

Capítulo 3: Este capítulo se centra en la descripción detallada del protocolo newscast dentro del contexto de las plataformas P2P actuales. Además, la dinámica del protocolo es analizada en tiempo de ejecución mostrando su robustez, escalabilidad masiva y capacidad de diseminación de la información de forma eficaz.

Capítulo 4: Se presenta el modelo de Agente Evolutivo como el marco de trabajo con el que esta tesis evaluará la viabilidad del enfoque de computación evolutiva P2P. También se esbozan las primeras claves del rendimiento computacional del modelo preveyendo su capacidad de sostener ganancias lineales en problemas computacionalmente pesados sobre plataformas de alto rendimiento.

Capítulo 5: Este capítulo analiza de forma experimental el rendimiento del modelo evolutivo P2P de tal forma que la viabilidad del enfoque pueda ser extraida de los resultados. Los experimentos se centran en tres casos de prueba que respectivamente estudian la escalabilidad del algoritmo, la influencia que la estructura de la población en el rendimiento y la toleracia a fallos del modelo para distintas tasas de degradación del sistema.

Capítulo 6: Finalmente, en este capítulo se exponen las principales contribuciones de esta tesis al area de los algoritmos evolutivos paralelos así como posibles extensiones en trabajos futuros.

Preface

Evolutionary Algorithms are a set of population based stochastic search techniques able to solve optimisation problems in reasonable time. However, the execution times of EAs can be high for very demanding problems and parallelism arises as an alternative to improve the algorithm performance and to speed up times to solutions.

In that context, this thesis presents a spatially structured EA able to take full-advantage of the large amount of available resources in P2P platforms. Such an approach defines a decentralised population structure by means of a P2P protocol in which every individual has a limited number of neighbours with the mating choice locally restricted within the P2P neighbourhood. The emergent population structure behaves as a small-world topology and plays an important role in the preservation of the genetic diversity. That way, population sizes can be minimised and execution times improve.

Nevertheless, there are remaining challenges towards an efficient design of P2P EAs. Questions such as *decentralisation* (such a computation paradigm is devoid of any central server), *scalability* (since P2P

systems are large-scale networks) or *fault tolerance* (given that computational resources are added and eliminated dynamically) become of the maximum interest and have to be addressed. Therefore, this thesis focuses on analysing such issues (i.e. decentralisation, scalability and fault-tolerance) in order to conclude the viability of the Peer-to-Peer Evolutionary Computation paradigm.

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

AES:	Average Number of Evaluations to Solution
$AESQ_3$:	Evaluations to Solutions in Third Quartile
BB:	Building Block
BOA:	Bayesian Optimisation Algorithm
CA:	Cellular Automaton
CEA:	Cellular Evolutionary Algorithm
CPU:	Control Processing Unit
DHT:	Distributed Hash Table
DREAM:	Distributed Resource Evolutionary Algorithm Machine
EA:	Evolutionary Algorithm
EC:	Evolutionary Computation
EDA:	Estimation of Distribution Algorithm
EvAg:	Evolvable Agent
EP:	Evolutionary Programming
ES:	Evolution Strategy
GA:	Genetic Algorithm
GE:	Genotypic Entropy

GGA:	Generational Genetic Algorithm
GP:	Genetic Programming
MBF:	Mean Best Fitness
P2P:	Peer-to-Peer
sGA:	Sequential Genetic Algorithm
SMP	Simmetric Multi-processing
SR:	Success Rate
SSGA:	Steady State Genetic Algorithm

Chapter 1

Introduction

Evolutionary Algorithms are a set of bio-inspired techniques for optimisation based in the Darwinian process of natural selection. As in the evolution of species, those individuals (*or candidate solutions*) showing to be the fittest are preferentially selected for mating so that offsprings inherit their genes through the course of generations. Iteratively, selection acts as a filter for genes and just those belonging to optimal solutions are able to overcome the selection pressure and recombine forming higher order solutions. It is within that process where the stochastic based search of evolutionary algorithms has been shown to succeed in optimisation problems [19].

However, for very demanding applications and large problem instances computational requirements may become so high that delay finding optimal solutions in reasonable time. Fortunately, the nature of evolutionary algorithms is inherently suited to be parallelised offering a straightforward way to improve scale-up properties. The main idea is to speed-up the execution times by sharing the workload of the individuals among a pool of processors [13].

In that context, this thesis proposes a new parallel evolutionary algorithm (denominated Evolvable Agent model) that takes full advantage of a dynamic Peer-to-Peer environment. The motivation behind the algorithm is tackling large instances of hard optimisation problems in an efficient and accurate way via massive scalability of Peer-to-Peer systems. To this end, a good understanding of the underlying computing platform can leverage on an efficient design.

Peer-to-Peer systems offer a powerful parallel infrastructure for evolutionary computation able to constitute a single virtual computer composed of a potentially large number of interconnected resources [67]. However, such a computing platform is devoid of any central server which challenges the central management of the evolutionary cycle (parent selection, reproduction, survivor selection). To cope with the issue, the Evolvable Agent model designates each individual as a peer and adopt a decentralised population structure defined by the Peer-to-Peer protocol newscast [43]. Then, any given individual has a limited number of neighbours and the mating choice is restricted within the local Peer-to-Peer neighbourhood.

In addition to decentralisation, a remaining challenge that accounts for an efficient parallelisation is that peers are prone to failures and computational resources are added and eliminated dynamically, often as a consequence of a decision from an user that volunteers CPUs under his control. This way, a Peer-to-Peer evolutionary algorithm has to show resilience to the peers dynamics. In that sense, the Evolvable Agent implements a graceful degradation being able to tackle large problem instances in spite of nodes departing from the system without other mechanism than its own emergent behaviour.

Summarising, an efficient Peer-to-Peer Evolutionary Algorithm is that one able to tackle large problem instances in a decentralised way in spite of nodes departing from the system. Therefore, this thesis focuses on analysing the issues of decentralisation, scalability and fault-tolerance in order to conclude the viability of the Peer-to-Peer Evolutionary Computation paradigm.

To that aim, the Evolvable Agent model has been empirically analysed in a simulated Peer-to-Peer environment where experiments are conducted under different scenarios using trap-functions as benchmark. These functions represent a set of decomposable problems based on unitation in which the level of difficulty can be tuned and the total fitness is additively calculated by summing partial fitness. That way, it is possible to observe how population sizes and computational efforts scale with increasing problem size and difficulty.

1.1 Motivation of the thesis

The main motivation behind this thesis is tackling those large problem instances in which, due to memory or computational constraints, sequential approaches are unsuitable. In concrete, this thesis tries to cope with the challenge of massive parallelisation of EAs derived from the new synergies in the computer architecture and EC areas.

With respect to the advances in computer architecture, parallel computing platforms cover nowadays a wide range of systems going from special purpose architectures to interconnected off-the-shelf computers. The case of this last has received much attention from the scientific/technical community which has promoted its development. That way, there is a set of technologies taking advantage of deallocated resources that are controlled at a network level with the aim of reducing costs associated to the management of hardware infrastructures. Within such technologies, GRID [24], Cloud [72] and P2P Computing [66] are probably the best-known and widespread. Despite having differences between, all of them share the concepts of massive scalability and virtualisation of a decentralised and heterogeneous underlying infrastructure. In fact, there is no a clear line dividing their respective application fields and the convergence between them has been reported in the literature as something unavoidable, e.g. Foster in [25]. Therefore, our distributed approach to EC aims to go an step further from a mere P2P parallelisation to become a prove of concept for a whole set of technologies based on the decentralised management of computing resources.

On the other hand, the massive use of such computing platforms is justified from the perspective of EC. As established by Goldberg's population sizing theory in [35], there is an increasing necessity of computing resources in EC when tackling problem instances becoming large. In other words, the population sizing theory states that there is an optimal criterion for tuning the population size of an EA, so that, a small problem instance requires a smaller population size than a larger instance of a more difficult problem. As the instance size increases, the computational requirements scale, becoming specially demanding for the case of very large problem instances.

In that sense, studies of scalability on the population size, such e.g. Thierens in [70] for GAs or Pelikan et al. in [57] for BOAs, show that the population size should roughly scale with an order $O(l^{\alpha})$, where l is the chromosome length and α is a constant that depends on the algorithm and the problem complexity. In the same way, Fernandes and Rosa show in [21] that the number of generations g required to find optimal solutions also scale following an order $O(l^{\beta})$. Such factors influence the scalability of an EA as highlighted in Algorithm 1.

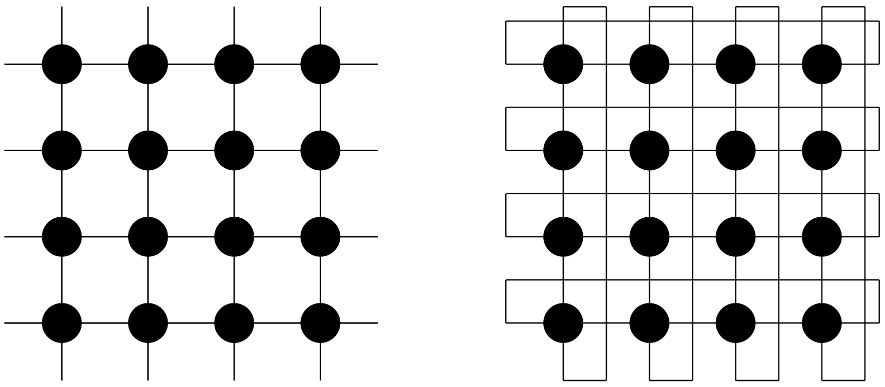
Algorithm 1 Time consuming keys in the evolutionary loop

```
for $i = 0$ to $g$ generations do
    for $j = 1$ to $n$ individuals do
        $evaluate_{n_{ij}}(l)$
    end for
end for
```

Therefore, the overall scalability order of an EA might be represented as $O(l^{\alpha+\beta+\zeta})$ by simply assuming a problem in which the evaluation function scales with a polynomial order $O(l^{\zeta})$ which is rather reasonable considering realistic problems.

In this context, parallel EAs aim to outperform the scalability orders of sequential approaches by improving the α-component which is responsible of the population size and the β-component that stands for the speed of the algorithm convergence. In particular, fine grained spatially structured EAs (that are described in detail in Section 2.1.3) might be able to reduce the scalability order from $O(l^{\alpha+\beta+\zeta})$ to $O(l^{\beta'+\zeta})$ whenever we assume that every individual is placed on a different processor and that the algorithm design is able to improve the speed of convergence from β to β'. In addition to an accurate calibration of the evolutionary operators, such an improvement is usually owns to the population structure that plays a key role on the preservation of the genetic diversity and, therefore, on the algorithmic performance.

Within this line, Giacobini et al. study in [30, 31] the impact of different regular lattices, as those depicted in Figure 1.1, on the selection pressure of an EA. However, spatially structured EAs are not only subject to regular population structures and there are some interesting results considering complex networks as population structure (see Figure 1.2). In this sense, previous authors analyse in [33] the influence of random and small-world structured populations on the selection pressure and empirically demonstrate in [32] that complex network population structures are competitive against panmictic EAs. Such results show the suitability of the spatially structured approach for decentralised systems as P2P system since they are inherently organised as complex networks. Therefore, our approach to P2P EC in this thesis is designed as a spatially structured EA in which the population structure is defined by a P2P overlay network.



(a) Grid topology. (b) Toroidal grid.

Fig. 1.1 — Different grid topologies.

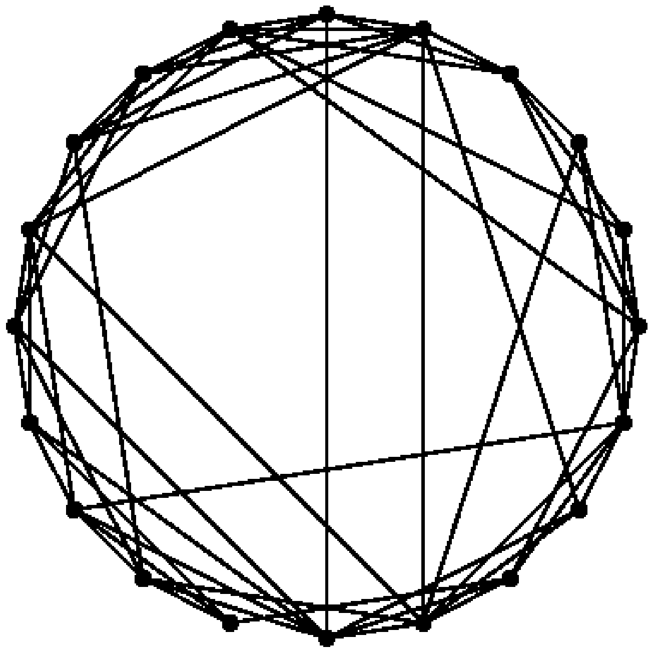


Fig. 1.2 — Complex graph topology.

1.2 Structure of the thesis

Current chapter of introduction has a descriptive orientation on the main goals and motivations behind this thesis. To that aim, P2P EAs are presented as an alternative for tackling large problem instances with high computing requirements whose viability will be studied in the following chapters.

Chapter 2 reviews the most extended models of parallel and distributed EAs and, in particular, those approaches in the literature related to P2P EAs. Besides, it provides with some insights into the role that the population structure plays on the environmental selection pressure of EAs.

In chapter 3, newscast is put on the context of the current P2P platforms. Furthermore, the protocol dynamics are assessed on the issues of decentralisation, massive-scalability and fault-tolerance.

Chapter 4 presents the Evolvable Agent Model as the framework to evaluate the viability of the P2P EA approach. In addition, it provides some keys on the computational performance of the model and how linear speed-ups can be hold for very demanding problems in high performance computing platforms.

In chapter 5, the experimental analysis of the model is proposed in a simulated P2P environment so that the viability of the P2P EA can be drawn from the *algorithmic performance* of the approach. Experiments are conducted for three different test-cases which focus on the algorithm scalability, the influence of the population structure in the algorithm performance and the fault-tolerance of the model for different degradation rates of the underlying computing platform.

Finally, chapter 6 exposes the main contributions of this thesis to the P2P EC area as well as possible extensions in future works.

Chapter 2

Parallel and Distributed Evolutionary Algorithms

As saw in chapter 1, computational efforts in EAs can be high for demanding problems with parallelisation arising as a way of improving the algorithm performance. In that context, a good knowledge of adequate parallel models turns into a key for providing efficient designs.

This chapter introduces a general overview on the most extended models of parallel and distributed EAs in Section 2.1. Rather than presenting an exhaustive analysis, the aim is to establish a global framework to help understanding the following chapters in the concrete computer architecture area of P2P Computing. This way, Section 2.2 provides some keys on how complex systems, such as P2P, can host distributed EAs by structuring the population with the shape of a complex network. Finally, Section 2.3 complements the view with the most relevant approaches to P2P EAs in the literature.

As a final consideration, we have adopted the following criterion for the alternate use of the concepts *parallel* and *distributed*. By *parallel* we will refer to the property of an EA to be subdivided in sub-tasks that can be executed with a certain degree of independence. On the other hand, *distributed* refers to the parallel quality of several sub-tasks for being executed in a loosely-coupled fashion (e.g. in distributed memory systems in which every processor access its own private memory) [56, 26].

2.1 Parallel and Distributed Evolutionary Algorithms models

Parallel and Distributed EAs models face two different aspects, the algorithmic and the computational performance. The first is related to the changes that the algorithm structure suffers when deployed on several processors while the latter corresponds to the computational speedup that can be expected. In fact, parallel EAs are studied as a way of preserving genetic diversity while improving the execution time of the algorithm.

The following sections present different models of parallelisation showing that not all of them fit well with P2P systems due to issues such as decentralisation, massive scalability or fault-tolerance. The parallel models are presented in two main groups according with the classification of Alba and Tomassini in [4]:

- Global parallel EAs (Section 2.1.1) in which there is a single population that follows the panmictic scheme of reproduction of a sequential EA. In such an approach, parallelism lies in the parallel evaluation of individuals and sometimes the parallel application of the genetic operators.
- Spatially structured EAs in which the parallelism is present at population level. The purpose is to balance the algorithm workload by spatially structuring the population of individuals among the available processors. Sections 2.1.2 and 2.1.3 present different approaches to spatially structured EAs depending on the parallelisation grain.

2.1.1 Global parallel evolutionary algorithms

This approach takes advantage of the parallelism at evaluation level in the case of very demanding *fitness* evaluation functions. Global parallelisation consists in the parallel evaluation of the individuals, usually following the *master-slave model* depicted in Figure 2.1 in which the algorithm runs on the master node and the individuals are sent for evaluation to the slaves. Additionally, the master is responsible for collecting the results and applying the genetic operators. Therefore, scalability is often limited by the master performance and the incoming bandwidth leading to sublinear speedups. For instance, Hauser and Männer in [38] report an speedup of 5 using 6 processors. Abramson et al. in [1] get a linear speedup up to 16 processors in a 128 processor architecture. Finally, Cantú-Paz points in [13] that speedup degrades quickly using such a model as a consequence of the master acting as a bottleneck in distributed memory computers. Therefore, we can conclude that global parallelisation models do not fit with the P2P systems requirements on massive scalability.

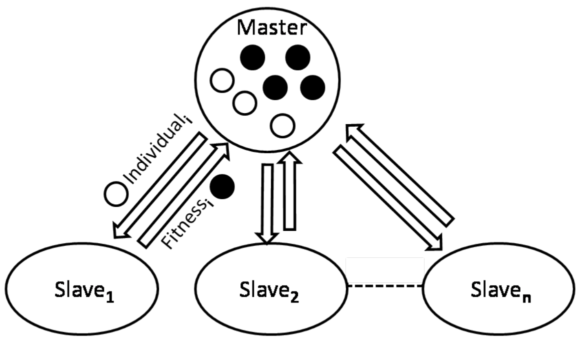


Fig. 2.1 — Global parallel evolutionary algorithms. Master-Slave model.

2.1.2 Spatially structured coarse-grained approaches

One of the most usual and widely studied coarse-grained approaches is the *Island model* depicted in Figure 2.2. As described by Cantú-Paz in [12], the idea behind this model is that the global panmictic population is split in several sub-populations or demes called islands. The communication between islands is defined by a given topology, through which they exchange individuals (migrants) with a certain rate and frequency. The migration follows a selection policy in the source island and a replacement policy in the target one.

Practitioners generally use a fixed population size P in studies of scalability, a maximum number of islands N and a population size per island of P/n where $n = 1, \dots, N$. That is the case described by Hidalgo and Fernández in [39] where the authors show how the algorithmic results are highly sensitive to the calibration of the multiple parameters, and in particular, to the number of islands. Specifically, the best algorithmic result is obtained using a single island, that is, a sequential EA. In addition, the authors conclude that using a small number of individuals per island is usually a bad choice. Such results are in consonance with those of Cantú-Paz and Goldberg in [14] suggesting that for difficult problems (large problem instances with high-order BBs) the best choice is using a single deme with a large population. Therefore, the island model turns into an unsuitable alternative for P2P systems, given that, distributed EAs over P2P systems have to focus on the efficient load-balance of a population among a large amount of resources in order to tackle large problem instances.

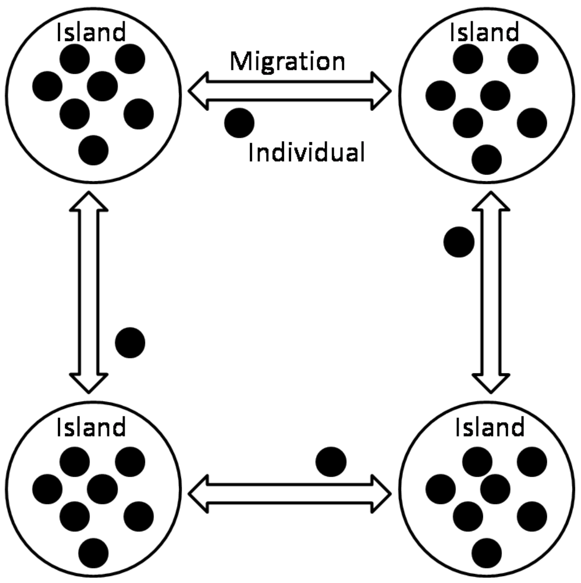
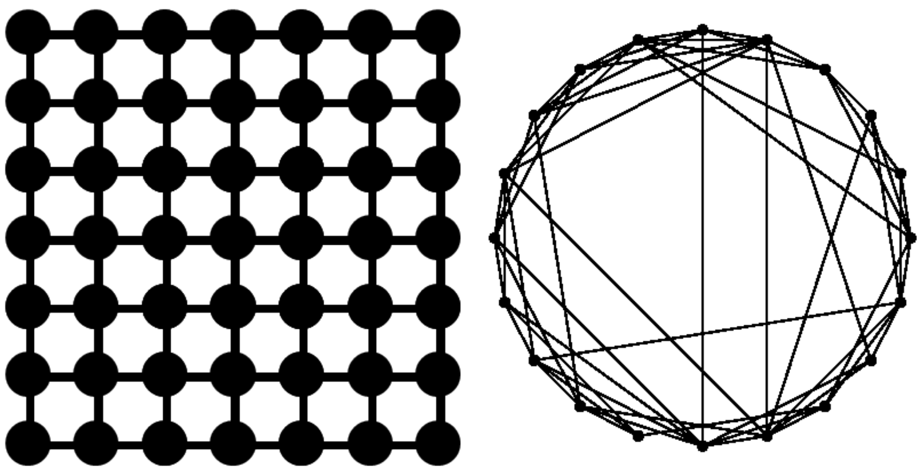


Fig. 2.2 — Spatially structured coarse-grained approach. Island model.

2.1.3 Spatially structured fine-grained approaches

In *fine-grained approaches* every individual in the population is placed on its own processor and evolves within a defined neighbourhood. One of the most common fine-grained approaches is the Cellular Evolutionary Algorithm (CEA) [3]. Individuals in CEAs are usually disposed in n-dimensional grid lattices as depicted in Figure 2.3 (a) where the mate choice is restricted to those individuals in the neighbourhood, thus, unlike in panmictic EAs, selection is decentralised.

Most of the efforts in CEAs have focused on analysing the algorithmic effects of using different neighbourhood policies. For example, Giacobini studies in [30] the impact of regular lattices on the selection pressure and of different graph structures such a toroid in [31]. Dorronsoro et al. study in [17] the effect of different grid shapes and asynchrony in CEAs. However, fine-grained approaches are not only subject to regular population structures and there are some interesting results considering complex networks as population structure (see Figure 2.3 (b)). Giacobini et al. analyse in [33] the influence of random and small-world structured populations on the selection pressure and empirically demonstrate in [32] that complex network population structures are competitive against panmictic EAs. Such results show the suitability of the fine-grained approach for decentralised systems as P2P system since they are inherently organised as complex networks.



(a) Regular lattice neighbourhood. (b) Complex network neighbourhood.

Fig. 2.3 — Spatially structured fine-grained approaches.

2.2 Population structure as a complex network

To help understand the role of the population structure in a P2P EA, this section introduces the structural design of a simple and easy understandable complex network proposed by Watts and Strogatz in [74]. As described by the authors, the procedure for building a small-world topology can start from a ring lattice with n vertices and k edges per vertex. With a given probability p, each edge is rewired at random. Since the procedure does not allow duplicate edges, no edge is generated whenever it matches an existing one. This way for a rewiring factor of $p = 0$ the ring lattice is kept while for $p = 1$ a random graph is generated. It has been shown that already for small values of p, the average distance between two nodes decreases rapidly.

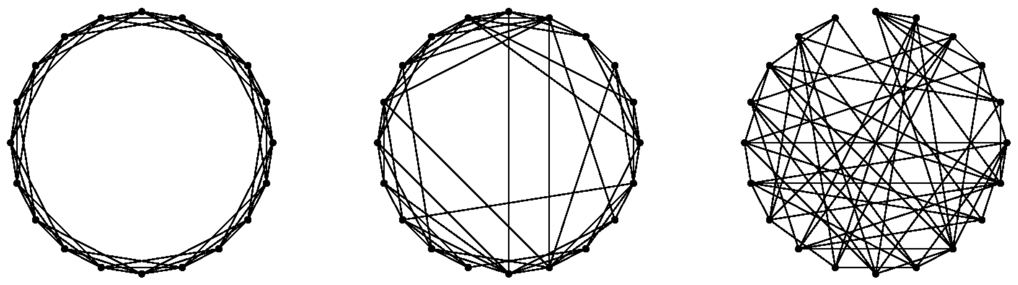


Fig. 2.4 — Watts-Strogatz graphs with $n = 20$ and $k = 6$. From left to right, the original ring lattice for $p = 0$, a small-world graph for $p = 0.2$ and a random graph for $p = 1$.

Figure 2.4 shows three stages of evolution of the Watts-Strogatz model in which the small-world graph preserves the high clustering coefficient of regular lattices and the small average path length of random graphs. Despite having a larger average path length than panmictic graphs, the inhomogeneity in such kind of topologies was shown by Giacobini et al. in [33] to induce qualitatively similar selection pressures on EAs compared to panmictic population structures.

The influence in the environmental selection pressure of such population structures can be represented by their takeover time curves. Goldberg and Deb define in [34] the takeover time as the time that it takes for a single, best individual to take over the entire population with no mechanism other than selection. Hence, takeover time is the proportion of best individuals formulated as a function of time.

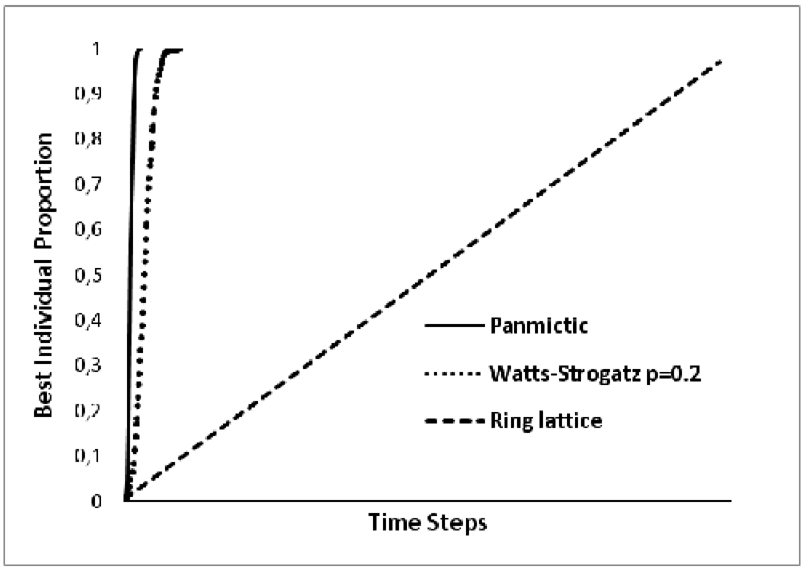


Fig. 2.5 — Takeover time curves in a panmictic population structure, Watts-Strogatz population structure with $p = 0.2$ and the original ring lattice with $k = 2$. All results averaged from 50 independent runs, for a population size of $n = 1600$ and binary tournament.

Figure 2.5 shows that the takeover time curve in the Watts-Strogatz graph is similar to a panmictic graph meaning that the induced selection pressures using both topologies are roughly equivalent. As in Watts-Strogatz small-world topologies, this thesis will show that P2P topologies can induce similar selection pressures to the panmictic one, allowing in addition a better scalability behaviour at the lower edge cardinality of P2P systems.

2.3 A review on Peer-to-Peer Evolutionary Algorithms

A first insight from the sections above is that P2P EAs models should be closer to fine-grained approaches in order to be scalable. However, studying scalability in distributed EAs over P2P networks is not straightforward and is usually approached in two complementary ways, using real environments [23] or using simulations [8]; either way presents its own advantages and drawbacks.

On the one hand, experiments in real environments present some challenges that, so far, pose a whole set of practical problems beyond the state of the art. The main reason is the difficulty to gather a large amount of reliable resources. Whenever the study of scalability is reduced to a few peers, no conclusions about massive scalability can be drawn. However, if the amount of peers is large enough, other questions, such as fault tolerance, arise [54]. On the other hand, using simulations simplifies the analysis and allows focusing on the structural design since restrictions like the harnessing of computing power or the peers' failures disappear. The drawback in this case is that simulations imply a certain number of assumptions about the real environment; hence, they have to be well stated (e.g. representing a pessimistic scenario as in [8]).

Some frameworks have been proposed in this context, some of the most relevant works that have tackled scalability are detailed bellow:

- *DREAM* [7], is one of the pioneering frameworks for P2P Evolutionary Computation (EC), the proposal focuses on the distributed processing of EAs and uses the P2P engine DRM (Distributed Resource Machine) which is an implementation of the newscast protocol [43]. However, the island-based parallelisation of *DREAM* was shown in [47] to be insufficient for tackling large-scale decentralised scenarios. There are some other frameworks based on *DREAM* as *G2DGA* [10] which uses G2P2P instead of DRM and focuses on Genetic Algorithms among all the EAs paradigms.

- Folino and Spezzano propose in [23] the *P-CAGE* environment for EC in P2P systems which is a hybrid model combining islands with cellular EAs. Every peer holds an island and every island a cellular EA. Despite results outperforming canonical EAs (either in execution time or convergence speed), the scalability analysis is limited to ten peers and the algorithm yields the best performance with five peers which points to poor scalability.

- Lee proposes in [48] a parallel system for EC using the P2P framework JADE. The optimisation is performed by three kinds of agents: state agents for controlling whether the peer is active or not, mobile agents for performing the evolutionary computation and the synchronising agent, a centralised agent for synchronising all the active peers. The algorithm's execution time speeds-up linearly but the scalability analysis is limited to eight nodes from which no conclusions about true scalability can be extracted.

In addition to the previous approaches for real environments, some other works in the literature face the design of P2P EAs by means of simulations, focusing on the viability of the approaches rather than dealing with the harnessing of computing power.

- The self-organising topology evolutionary algorithm (*SOTEA*) by Whitacre et al. in [75] is an EA designed for the sake of diversity maintenance. To this end, the authors focus on a self-organised population structure with the shape of a complex network. The network co-evolves with the EA by following two rules (from which a power law population structure emerges):

  1. Reproduction rule: When a new offspring is created, SOTEA adds a new node, this node is linked to its parent (asexual reproduction). The parent's connections are inherited by the offspring with certain probability P_{add}. In addition, all inherited connections are lost by the parent with probability P_{remove}.

2. Competition rule: A random selected individual competes with its less fit neighbour. From such a competition, the loser is killed and the winner inherits all its connections.

By following these two rules, SOTEA keeps a better population diversity than the Cellular and Panmictic GA used as a baseline for comparison.

- Along the same lines of self-organised algorithms, Wickramasinghe et al. present in [76] a P2P EA with two particular properties: autonomous selection and natural reproduction. Autonomous selection means that the individuals in the population decide on their own whether and when they want to reproduce and to survive without any central control. To this end, they use information on their own fitness and estimations about the total population to support decision making. The second special feature, natural reproduction, means that birth and death decoupled [18]. That is, an individual can be removed without being replaced by a child and a child can be born without removing an existing individual first. This is highly uncommon in EAs and as a consequence, the population size varies at run-time (just like in natural evolution) and a self-adjusting selection pressure mechanism is needed to prevent population implosions or explosions.

Nevertheless and to the best of our knowledge, none of these approaches perform an integral analysis of viability taking into account the issues of decentralisation, scalability and fault-tolerance in P2P systems at once. Therefore, this thesis aims to present and analyse a simple and easy-understandable model for P2P EAs that cope with the mentioned issues, so that, the general viability of P2P EAs can be concluded.

2.4 Summary

This chapter describes the main parallel and distributed EA models reviewing the possible advantages and drawbacks of adapting a given model to a P2P system. To this aim, parallelism has been considered at two different levels in the structure of an EA, evaluation and population levels. The first takes adventage of the parallel evaluation of individuals on different processors in an approach known as global parallelisation. A global parallel EA keeps a single and centralised population that evolves following the same scheme than in the sequential mode. The centralised management of the evolutionary loop represents a bottleneck on the master node and linear speedups can be expected up to a few processors imposing, this way, a limitation to the massive scalability of P2P systems. In the second case, parallelism is considered at population level in which the global population is divided into several demes running on different processors and communicating with each other, that is, the algorithm is spatially structured. Spatially structured EAs can be classified in coarse-grained and fine-grained approaches depending on the parallelisation grain. In the coarse-grained approach, the population is divided in several panmictic subpopulations while in the fine-grained approach, individuals interact with each other by means of the population structure, so that, potentially, they can be placed on different processors. In a similar way that the global parallel EA, the spatially structured coarse-grained model has some difficulties to take advantage of a large number of resources, specially, when tackling large and difficult problems. However, fine-grained approaches seems to be more suitable for massive parallelisation and the population structure can be defined with the shape of a complex network as in the case of P2P systems.

Finally, some of the most relevant works in the literature of P2P EAs are presented showing the interest of the scientific community in this field of research. Nevertheless, we miss an integral analysis of viability addressing the issues of decentralisation, scalability and fault-tolerance. This thesis aim to tackle such issues in the following chapters.

Chapter 3

Newscast in the context of Peer-to-Peer Computing

By the term Peer-to-Peer computing we will refer to those issues relating to protocol development, security management, analysis of performance and applications of P2P systems [66].

Peers (that will be also referred to as *nodes* in the context of communication graphs) are equal entities able to establish a self-organised and decentralised communication using their own routing mechanisms. Therefore, the working principle of the P2P technology is the ability of every *peer* to behave autonomously as a *servent*, that is, every peer acts as a SERVer and a cliENT.

Such a dual operational mode has been proved to be a good way to harness decentralised resources at the *edges* of the network (e.g. storage or computing power), giving a single coherent view of the system. Taking adventage of such features, the aim of this thesis is using P2P technology as a computing platform in the application area of Evolutionary Algorithms.

In order to explain the design decisions taken further on the following chapters and put EAs in the context of P2P technology, Section 3.1 reviews different taxonomies of P2P systems according to criteria such as system architectures or application areas. In such a framework, Section 3.2 analyses the specific protocol that will be used throughout this thesis, the *newscast* protocol. It consists in a fully-decentralised protocol firstly proposed in [43] by Jelasity and van Steen which has shown to succeed in the main issues related to P2P computing such as massive scalability or fault tolerance in e.g. [42, 73, 41]. Finally, Section 3.3 concludes with a summary about the described technologies stating how an easy understandable protocol such as newscast is representative of the P2P paradigm.

3.1 Taxonomies of P2P systems

In order to present the main issues involving P2P systems and how different applications approaches take adventage of P2P technology, this section explains two well-known methods for classifying P2P systems; the first one, in Section 3.1.1, classifies P2P systems according to the degree of decentralisation promoted by different kind of system architectures, meanwhile, the second one, in Section 3.1.2, provides a taxonomy on the different application areas of P2P systems.

3.1.1 System architecture

The harnessing of resources in P2P systems depends of three basic features of the architecture design, resource availability, resource discovery and resource retrieval. Availability and retrieval of resources refer to the way in which resources are disposed within the system, and therefore, the way in which they can be retrieved. Additionally, the discovery of resources addresses the *look-up* of contents in large and decentralised systems.

In Bittorrent [49], for example, files are available in pieces of equal size in a distributed network known as swarm. Two kind of special peers, seeders and trackers, have respective information on the location of pieces and peers. This way, seeders and trackers allow a client to discover the resources, so that, it can retrieve content from the swarm in a decentralised fashion.

Nevertheless, an efficient *look-up* mechanism turns out to be not trivial for large and decentralised systems. In this sense, the system architecture mainly depends on decisions about the indexing and discovery of resources and according to such design decisions, P2P architectures can be classified within the following three main blocks:

3.1.1.1 Centralised

The *look-up* service is implemented on central servers, and therefore, all the addressable content of the peers is centralised such as in Napster. The major drawback of centralised systems is that servers represent a possible bottleneck for massive scalability and a single point of failure. A good example is the history of Napster itself. Its controversial use for the illegal sharing of copyright protected files in addition with its centralised look-up architecture led soon to the intervention of the service in 2001 by simply closing the servers.

3.1.1.2 Decentralised

In order to cope with the limitations of centralised approaches, decentralised alternatives such as Gnutella 0.4 [29] have emerged during the last decade. In a nutshell, a decentralised system implies that many nodes are providers of the *look-up* service making the system robust to failures of punctual nodes.

Such kind of approach has shown to be massively scalable and is imposing within the current P2P architectures. The great number of P2P systems following a decentralised scheme requires of an additional subclassification. The most extended criterion for it is considering whether the peers do a pro-active effort to maintain a given network topology (in this case they are known as structured systems) or the properties of the network emerge from the collective behaviour of the peers (known as unstructured systems).

- **Structured systems**:

  It mainly refers to distributed hash table approaches (DHT) such as Chord [68], Pastry [60], Tapestry [77] or CAN [58]. DHTs consist in the distributed indexing of contents using pairs $(key, value)$. To this aim, the key space is composed of a set of integer values, e.g. from 0 to 2^{80}. Any node within the network receives a portion of key indexes and a routing table to l different nodes in such a way that entries point to nodes at the position $n + 2^{i-1}$, where $1 \leq i \leq l$ and n is the current node. This way, the *look-up* of a given *key* can be done in $O(log(n))$ steps.

- **Unstructured systems**:

  Some authors also refer to such systems as Pure P2P [66]. The main idea behind is that all elements in the system are distributed without any central component. Examples of these systems are Gnutella 0.4 [29] or newscast [43]. Decisions on the routing are taken locally by every peer, and therefore, the network topology is self-organised. In these systems, the *look-up* for contents is based on flooding mechanisms (a.k.a. gossipping or epidemic *look-up*), that is, a request is flooded through neighbour nodes taking a certain number of hops. The key to success is the small diameter of the network at the self-organised small-world relationship between nodes. Hence, a given *look-up* could take $log(n)$ hops with respect to a network of size n.

3.1.1.3 Hybrid or semi-centralised

Finally, there are hybrid (or semi-centralised) approaches consisting in a hub based architecture. Hubs, also known as Superpeers, have a higher degree of connections than the rest of peers, known as leafnodes. Despite hubs representing a sort of centralisation, they scale according to the network size. That is, the architecture allows massive scalability by increasing the number of superpeers with respect to the network size. On the other hand, the degree of leafnodes also scales following a power-law distribution. A good example of hybrid architecture is Gnutella 0.6 [45] whose aim was to reduce the load of the network with respect to the decentralised version, Gnutella 0.4.

3.1.1.4 Conclusions

As a summary, Figure 3.1 depicts the topologies of the three system architectures. In order to study the viability of P2P EAs, we have chosen newscast (a purely decentralised P2P system [43]) as the system architecture for this thesis. As it will be analysed in Section 3.2, newscast adopts the shape of a complex network and fits with all the requirements exposed in the previous Chapter for a fine-grained P2P EA such as decentralisation, massive-scalability or fault tolerance. In this sense, choosing a centralised approach would have imposed some limitations on the scalability in addition to the high vulnerability to failures on central nodes. On the other hand, hybrid approaches are the response that arise from issues related to load balancing the network traffic or the computing power. We find that such issues would correspond to decisions in later phases of design rather than in a preliminary study of viability.

3.1.2 Application areas

P2P systems have been widely used and mainly developed for file-sharing applications [61]. Nevertheless, the power for the harnessing of distributed resources -storage, computing power or human presence- has attracted the attention of researchers from different areas into P2P systems. Thus, Mobile ad-hoc networks (MANET) can been modelled following a P2P approach [11], Voice over IP (VoIP) protocols can be understood as an instance of P2P protocols [51] or idle cycles are used for computing applications [25]. In the widest sense, P2P is a key for understanding, approaching and modelling distributed resources providing a single coherent view of the system. The following points summarise four of the main application areas in which P2P systems are applied nowadays.

3.1.2.1 File-sharing

P2P systems were initially thought for this kind of applications. File-sharing P2P systems are platforms for the interchange of files between different final-users. Each one of these users is at the same time a service provider and a consumer requesting contents from other users. Therefore, contents had to be allocated in an addressable way in order

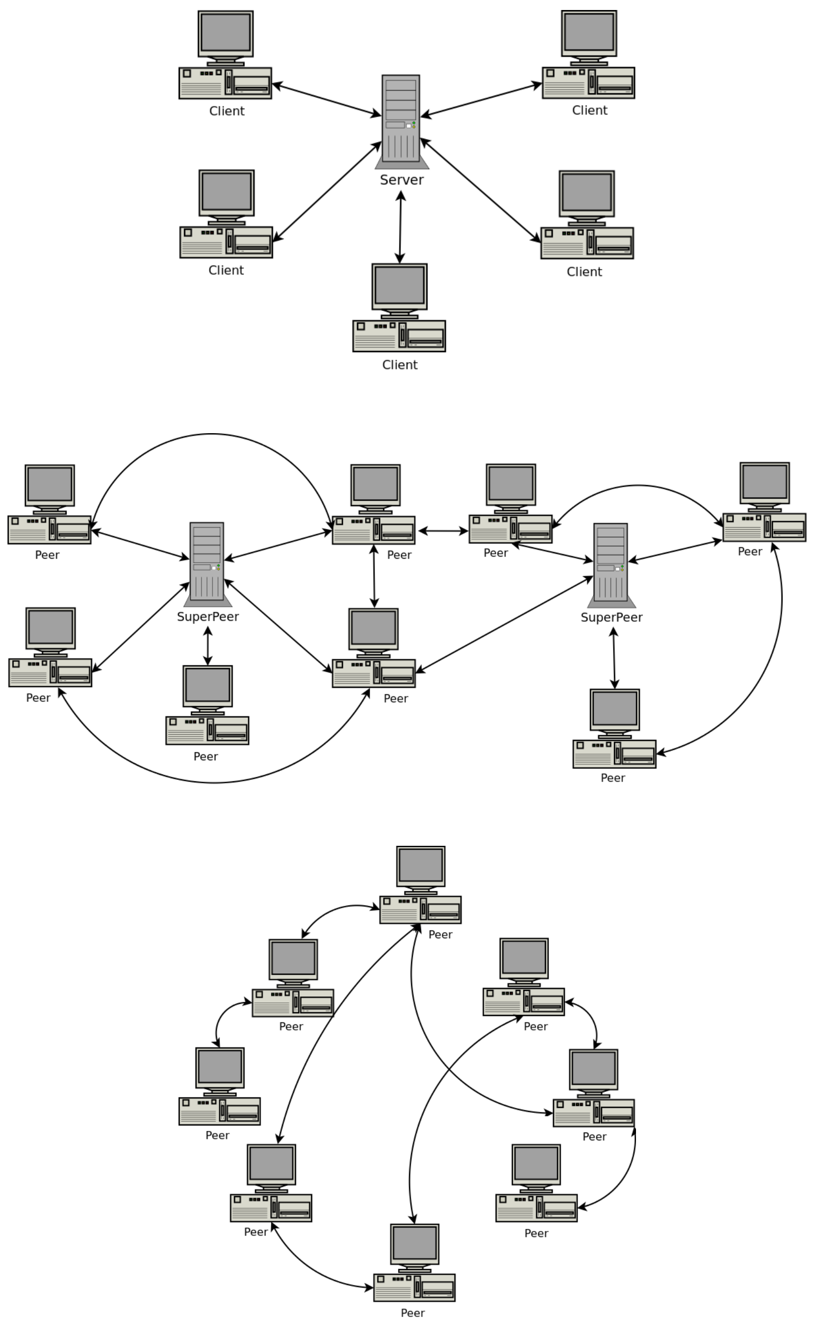


Fig. 3.1 — From top to bottom. Centralised, hybrid and decentralised architectures.

of being retrieved by any user willing to obtain them. Depending on the system architecture, the *look-up* problem can be addressed in a centralised way such in Napster [61], in a decentralised way such in Gnutella 0.4 [29] or in an hybrid way such in Gnutella 06 [45].

3.1.2.2 VoIP and Instant Messaging

Voice has been recently incorporated to traditional instant messaging technology. Applications such MSN Messenger or Net2Phone use traditional VoIP protocols such as SIP or H.323 [64, 52]. Nevertheless, the most extended VoIP application nowadays is Skype, a P2P based VoIP protocol. Lisha and Junzhou show in [51] that *QoS* is rather similar between traditional and P2P approaches. However, the great advantage of using such P2P based approach is that it can solve NAT and firewall problems.

3.1.2.3 Mobile ad-hoc networks (MANET)

MANETs are networks of sensors designed to collect information from the inside of a given phenomenon under study. The fields of application are numerous going from e.g. traffic jams control to glacial monitoring [44]. The basic features of sensors include sensing the environment, processing the data and communicate with each other in a collaborative way. To this aim, networking techniques require of self-organised protocols in order to cope with an ad-hoc and changing environment in which the position and persistence of the sensors is unreliable. At this point, P2P techniques are helpful and are widely used in MANETs for the management of decentralised topologies, the probabilistic multicasting of information in multi-hop architectures, the resilience of the system to sensors failures and the scale-up of the network size [11].

3.1.2.4 Cycle-sharing

Cycle-sharing P2P systems are distributed networks of heterogeneous single systems that contribute spare processor cycles for computing. A well-known example of application is the SETI@home project [5] searching for extraterrestrial intelligence patterns among a huge amount of radio signals collected from the spacial observatory of Arecibo in Puerto Rico. Despite strictly speaking SETI@home is based on Desktop Grid (DG) technology [6], the dividing line between both technologies is unclear and is mostly subject to differences in the network structure, centralised for DGs and decentralised for P2P. However, both technologies focus on the harnessing of computational power from a group of interconnected computers. Such source of computational power is based in one or both of the following approaches:

- **Volunteer Computing**:

  Volunteer Computing refers to distributed computing applications in which the source of computational power is aggregated from volunteers that willingly donate their idle CPU cycles to different research projects. In this case, cycle-sharing applications aim the harnessing of computing power at the edges of the

Internet [67].

- **Infrastructure applications**:

  There is no reason for limiting P2P technology to volunteer computing. As in the case of GRID technology, P2P systems can be hosted in private infrastructures. In fact, both technologies are compatible and they converge towards common places. Quoting Foster and Iamnitchi in [25] *"...both are concerned with the same general problem, namely, the organisation of resource sharing within virtual communities... ...both take the same general approach... ...each has made genuine technical advances".*

3.1.2.5 Conclusions

Among all the above mentioned application areas, this thesis might be catalogued as a study of viability on a cycle-sharing application. More concrete, an application based on distributed Evolutionary Algorithms. The main reason is that Evolutionary Algorithms performance depends on the complexity of the problems to be tackled. The more complex the problem instance, the larger the computing requirements. This fact leads to a sometimes prohibitively long time to solution that happens, for example, when tackling many real-world problems. In order to reduce the execution time of EAs, this thesis presents a cycle-sharing P2P system as an alternative platform for the harnessing of computer power at a very low cost.

3.2 Newscast Computing

As it has been mentioned at the introduction of this Chapter, we have chosen newscast to be the underlying P2P protocol in this thesis because it represents a good example of purely decentralised protocol that has shown to succeed in the main issues related to P2P computing such as massive scalability or fault tolerance. This section describes the design of its components and analyses the runtime dynamics of the protocol.

Newscast is a self-organised gossipping protocol for the maintenance of dynamic unstructured P2P overlay networks [43]. Without any central services or servers, newscast differs from other similar approaches [29, 68, 58, 60, 77] by its simplicity and scalability:

1. The membership management follows a extremely simple protocol: In order to join a group, a given node just has to contact any node within the system from which gets a list of its neighbours members. Additionally, to leave the group, the node just requires to stop communicating for a predefined time.

2. The dynamics of the system follow a probabilistic scheme able to keep a self-organised equilibrium at a macroscopic level. Such an equilibrium emerges from the loosely-coupled and decentralised run of the protocol within the different and independent nodes. The emerging macro-structure behaves as a small-world [74] allowing a scalable way for disseminating information and, therefore, making the system suitable for distributed computing.

3. Despite the simplicity of the scheme, newscast is fault-tolerant and exhibits a graceful degradation without requiring an extra mechanism other than its own emergent macro-behaviour [73].

Algorithm 2 shows the pseudo-code of the newscast protocol. Each node keeps its own set of neighbours in a cache that contains $c \in \mathbb{N}$ entries, referring to c other nodes in the network without duplicates. Each entry provides a reference to the node in which it was created and a time-stamp of the entry creation (allowing the replacement of old items).

Algorithm 2 Newscast protocol in $node_i$

```
Active Thread
loop
    wait $t_r$
    $node_j \Leftarrow$ Uniformly random selected node from $Cache_i$
    send $Cache_i$ to $node_j$
    receive $Cache_j$ from $node_j$
    $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_j$)
end loop

Passive Thread
loop
    wait $Cache_k$ from $node_k$
    send $Cache_i$ to $node_k$
    $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_k$)
end loop

$Cache_{aggregated} \Leftarrow$ Aggregate($Cache_a$,$Cache_b$)
$Cache_{aggregated} \Leftarrow Cache_a \cup Cache_b$
Keep the $c$ freshest items in $Cache_{aggregated}$ according with the time-stamp.
```

There are two different tasks that the algorithm carries out within each node. The active thread which pro-actively initiates a cache exchange once every cycle (one cycle takes t_r time units) and the passive thread that waits for data-exchange requests.

Every cycle each $node_i$ initiates a cache exchange. It selects randomly a neighbour $node_j$ from its $Cache_i$ with uniform probability. Then $node_i$ and $node_j$ exchange their caches and merge them following an aggregation function. In this case, the aggregation consists of picking the freshest c items from $Cache_i \cup Cache_j$ and merging them into a single cache that $node_i$ and $node_j$ will share.

3.2.1 Communication Graph Series

The dynamic sequence of relations between the different nodes defines a series of communication *digraphs* (directed graphs) D_t at time t starting, therefore, at D_0. In order to analyse the graph subseries $D_{it_r}(i = 0, \ldots, N)$, we assume that a complete communication cycle occurs within a time interval $[it_r, (i+1)t_r]$. Given the low communication exchange calls frequencies (this property will be analysed in Section 3.2.4 around the results in Fig. 3.9) this assumption is not unrealistic as long as the information exchange is short with respect to t_r.

In order to complete the characterisation of the graph series, we will analyse two of the most relevant properties in the communication graph G_t obtained from dropping the orientation in D_t: the average path length and the clustering coefficient. The average path length of a node n_i is the average of the minimal path between n_i and the rest of nodes while the clustering coefficient of n_i represents the fraction of its neighbours which are also neighbours between them.

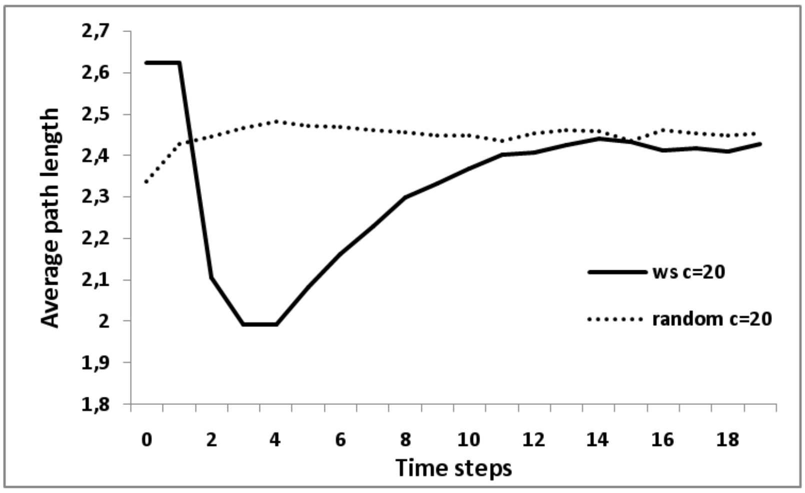
3.2.2 Bootstrapping and Convergence

This section studies the self-organised dynamics of the newscast communication graph series, showing that, independently of the initialisation criterion, the graph is able to bootstrap and converge to stable conditions. It reproduces the results of the newscast seminal paper in [43]. Figure 3.2 represents the convergence of the graph series bootstrapping from a random and a small-world graph at G_0. In quite an early stage, i.e. around G_{12t_r}, the graph has already converged to an state of dynamic equilibrium with independence of the initial graph G_0.

Nevertheless, the cache size (c) plays here an important role. It represents the maximum degree of a node, and therefore, influences the average path length and the clustering coefficient. For example, Figure 3.3 shows newscast converging to different values depending on c.

In addition, Figure 3.4 depicts the influence of the cache size on the average path length and the clustering coefficient for different network sizes. A smaller c implies a higher clustering coefficient and also a higher average path length.

Based on such features, a newscast graph can behave as a small-world by tuning c to the adequate values. In this sense, Jelasity and van Steen state in [43] that the intended normal setting of newscast is $c \ll n$; for $c \ll n$, newscast has the small average path length of random graphs but, as shown in Figure 3.5, a much higher clustering



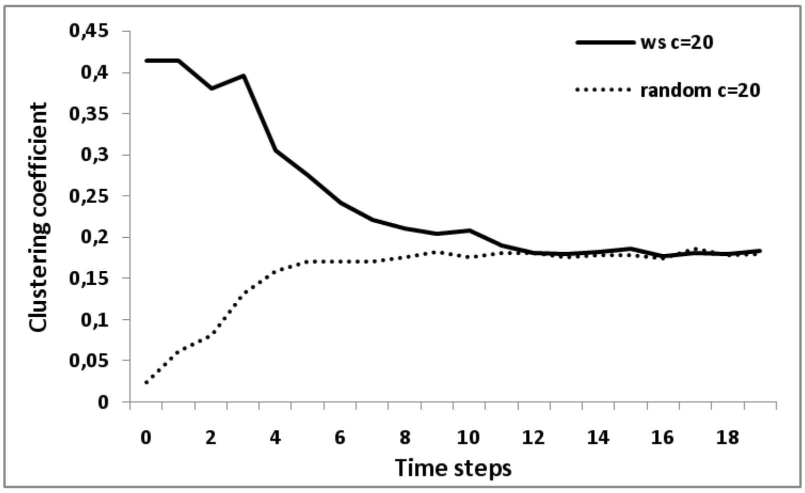
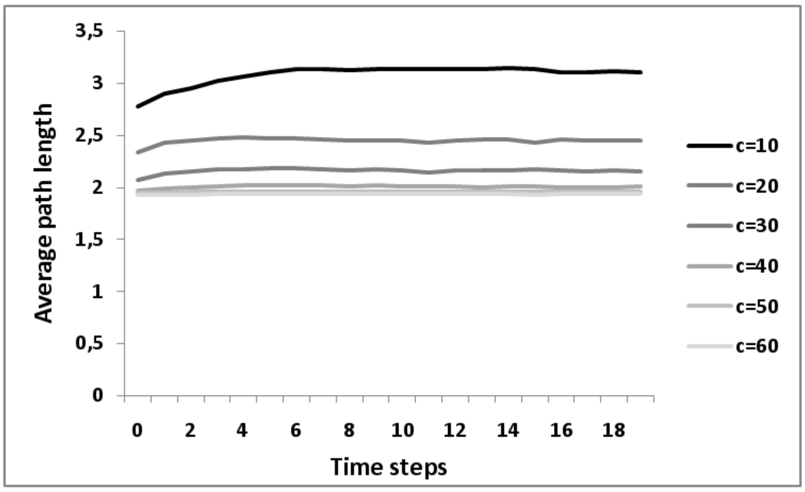


Fig. 3.2 — Convergence of the average path length (*up*) and clustering coefficient (*down*) bootstrapping from a random and a Watts-Strogatz (ws) graph for a number of nodes $n = 1600$. The graph can be seen to converge to the same values within the interval $G_{0t_r-20t_r}$ showing the independence of the protocol convergence with respect to the initialisation criterion.



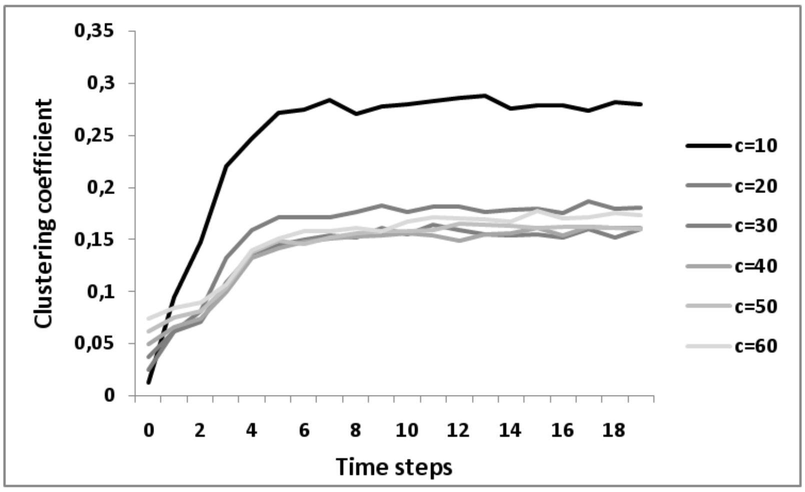
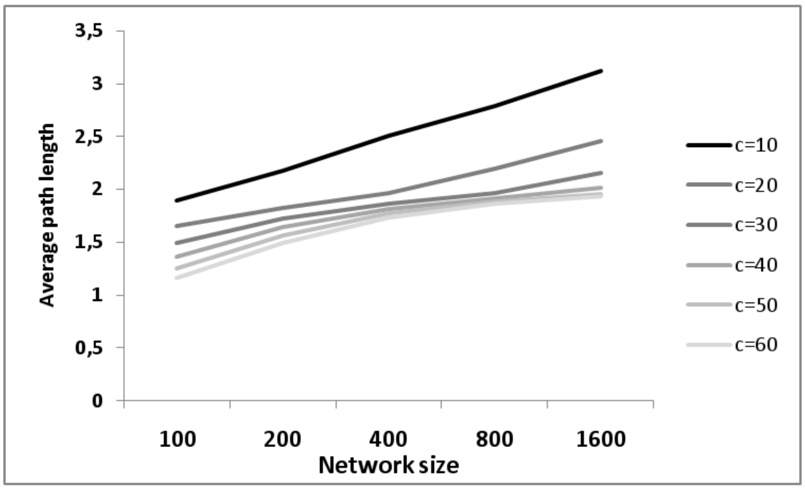


Fig. 3.3 — Convergence of the average path length (*up*) and clustering coefficient (*down*) bootstrapping from a random graph for different cache sizes and a number of nodes $n = 1600$



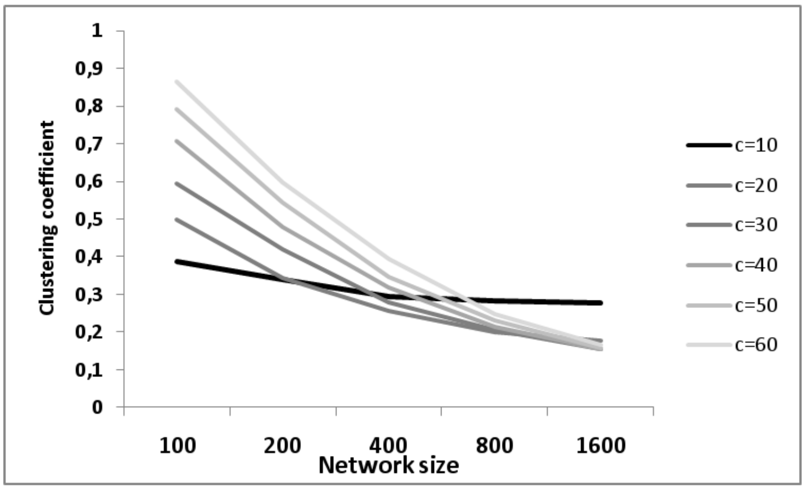


Fig. 3.4 — Converged average path length (*up*) and clustering coefficient (*down*) for G_{40t_r} and different network and cache sizes (c).

coefficient [73].

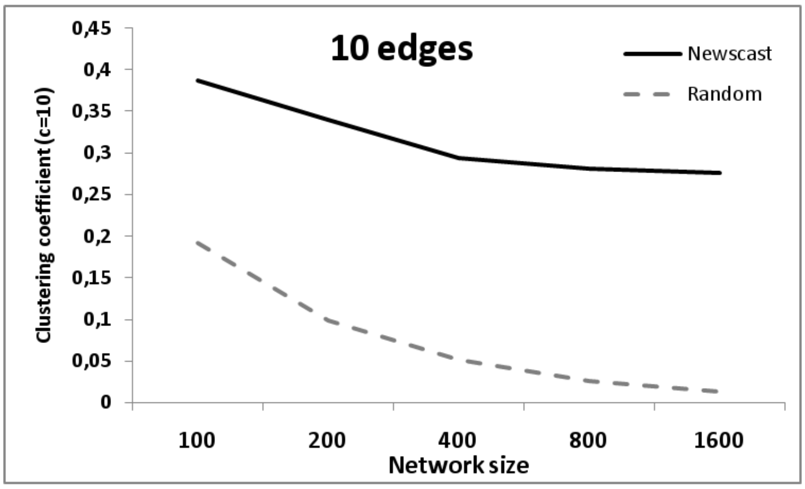
3.2.3 Robustness

One of the important issues regarding P2P computing is the robustness of the underlying protocols such as newscast, that is, protocols describe the dynamics of the communication graph series (G_t) in a continuous changing environment. In this section, we consider two important aspects regarding the robustness of the newscast protocol[1].

1. The probability of spontaneous partitioning of G_t.
2. The robustness of newscast to node removals.

The spontaneous partitioning of the communication graph series means that a subgraph becomes disconnected within the interval G_{0-t} as a consequence of the protocol dynamics. Figure 3.6 shows that in newscast the probability of spontaneous partitioning is mainly influenced by the cache size c. Despite the results showing that the null probability is reached for $c = 10$, it has to be considered that results are obtained over 50 independent runs for relatively small networks within the interval $G_{0-20000t_r}$ and, therefore, it might be still possible to get a partition considering larger networks, a larger number of runs or a larger time interval. In this sense, the seminal study in [43] extends such results and establishes that the probability of spontaneous partitioning is almost negligible from $c = 20$ ahead.

[1]Results are reproduced from [73].



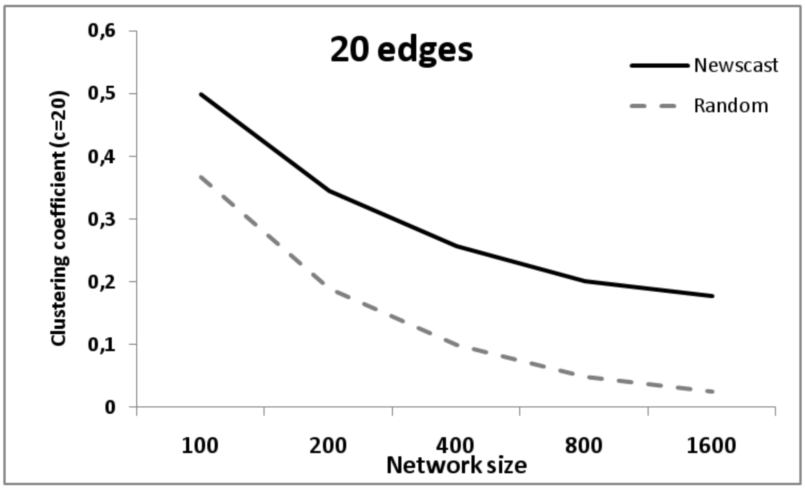


Fig. 3.5 — Clustering coefficients for equivalent random and newscast graphs (i.e. nodes have the same number of edges, ten on the upper figure and twenty at the bottom one). The higher values in the newscast graphs point to a small-world topology.

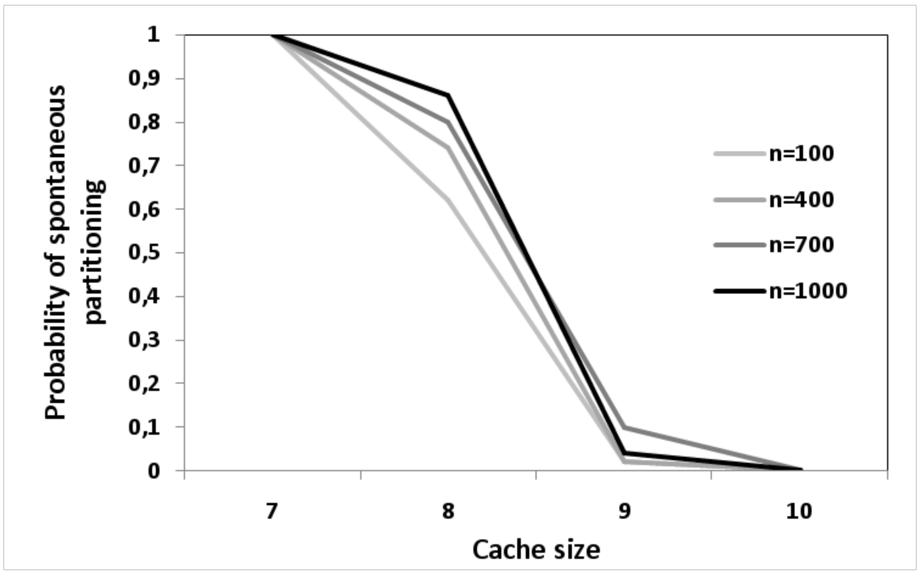
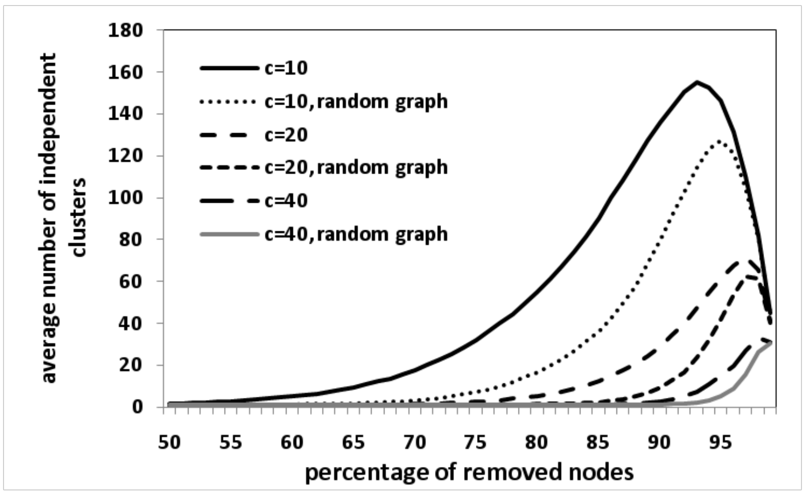


Fig. 3.6 — Probability of spontaneous graph partitioning as a function of the cache size. Probabilities are obtained from 50 independent runs starting at the random graph G_0 until G_{20000t_r} or partitioning.

Nevertheless, previous results on the spontaneous partitioning of G_t do not take into account peers failures which is an inherent feature of Peer-to-Peer systems known as *churn* [69]. Churn emerges from the collective dynamics of peers joining and leaving the system. Given that the running platforms use to be highly dynamic (e.g. the Internet) protocols have to be well-suited for tolerating faults. Therefore, we study in Figure 3.7 the robustness of newscast in an scenario in which nodes are removed up to none is left. It can be seen how newscast inherits the robust behaviour of random graphs, especially when c is large. On the one hand, the graph remains connected until a large percentage of nodes are removed, e.g. for $c = 40$, 90% of the nodes have to be removed to get a partition of the graph. On the other hand, most of the nodes remain in the larger cluster once the partition takes



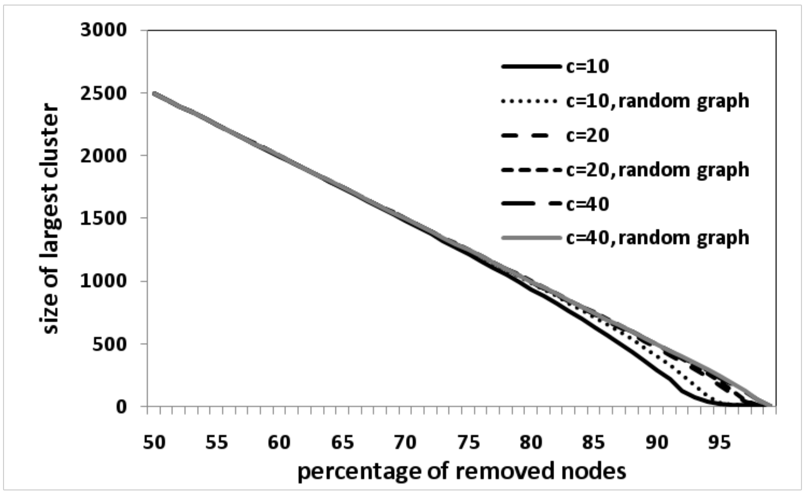


Fig. 3.7 — Partitioning of the communication graph as a function of the percentage of removed nodes in G_0 (random graph) and the newscast subgraph G_{20t_r}. Results are averaged from 50 independent runs and n=5000.

place.

3.2.4 Dissemination of the information

In order to make distributed computing possible, a key element in a distributed system is having a reliable information flow. Taking into account the decentralised and unreliable nature of P2P systems, newscast guarantees a scalable and reliable dissemination of the information by implementing a probabilistic *epidemic* multi-casting scheme. As in infectious diseases, a piece of information is able to "infect" the "healthy" neighbourhood of a given carrier node. The speed of dissemination is high at the small-world structure of the communication graph and the process is likely to "infect" the whole graph in few time steps.

Such an effect can be visualised in the following experiment. At time 0 one node produces a piece of information (so-called a news item) that is sent to one of the neighbours every time-step. Once that a node is "infected" begins to act as a replicant and sends the news item to its neighbourhood. This way, Figure 3.8 shows the proportion of "infected" nodes as a function of time. To this end, a complete graph and two different parameterised newscast graph were considered.

Similar curves in Figure 3.8 denote equivalent speeds in the dissemination of information induced by both kind of topologies. Nevertheless, the node degree in complete graphs of size n is $n-1$ while the average degree in newscast is approximately $2c$ pointing out a better scalability of the small-world approach given that the intended normal setting of

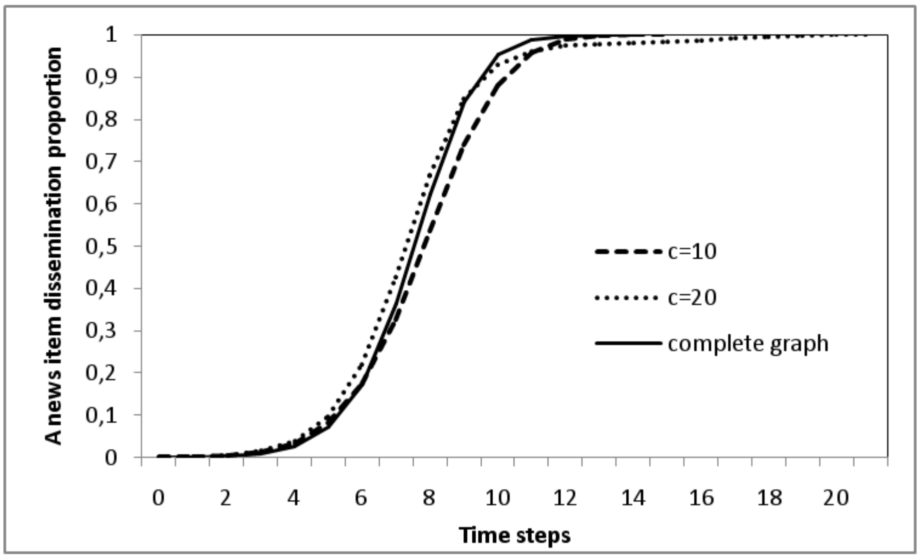


Fig. 3.8 — Dissemination speed of a news item through the entire network for a complete graph and two newscast graphs with $c = 10$ and $c = 20$. Results are averaged from 50 independent runs for a network size of $n = 1600$.

newscast is $c \ll n$.

In addition to a lower node degree than in complete graphs, the low frequencies of the information exchange calls are a good indicator of the scalability of the protocol as depicted in Figure 3.9. Independently of the network and cache sizes (i.e. assuming $c \ll n$), a node receives at any given cycle t_r less than 4 information requests with probability $p > 0.8$ and less than 8 with $p > 0.98$. Therefore, a randomly chosen node is likely to process few requests in a cycle. Whenever t_r is large enough with respect to a single request (e.g. 20 times larger) the protocol run does not represent a bottleneck on the system even when considering large network sizes.

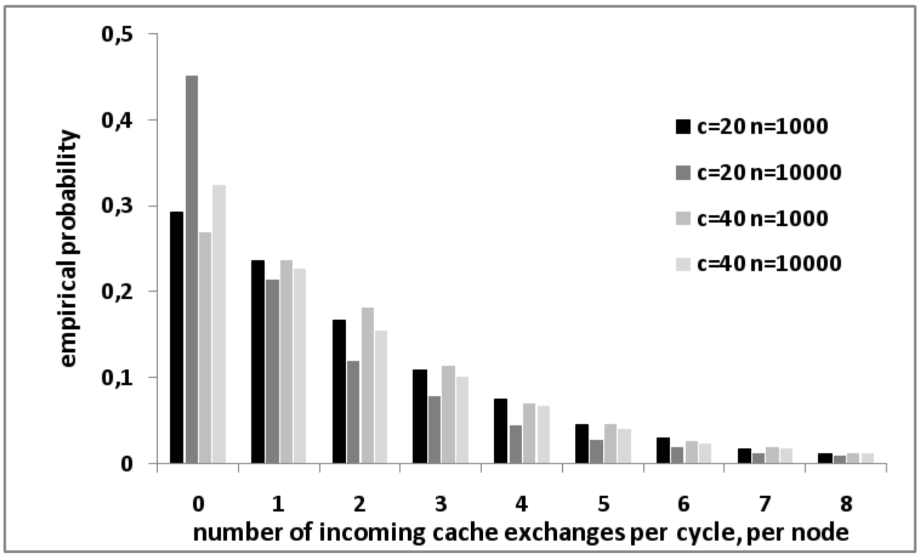


Fig. 3.9 — Information exchange calls probability in a fixed node during a time interval t_r (cycle). Results are estimated in the graph series $D_0 \dots D_{10000t_r}$ where D_0 is a random graph.

3.3 Summary

This chapter provides a general overview of the Peer-to-Peer Computing area. In order to provide an adequate understanding of P2P systems and a general taxonomy of the application areas, the system architectures and the different ways of approaching fully-decentralised schemes is introduced.

In addition, the chapter focuses on an exhaustive analysis of the newscast protocol that will be used within this thesis as the underlying running platform for the experiments. Newscast is a decentralised gossip-based protocol that follows an *epidemic* scheme for disseminating the information. Despite its simplicity, it is a robust and scalable protocol able to self-organise the relations between peers into a small-world communication graph. That is, independently of the initial conditions, the system converges to an state of dynamic equilibrium that behaves asymptotically as an small-world graph.

Chapter 4

A framework for P2P EAs: The Evolvable Agent Model

This chapter presents the Evolvable Agent Model as the framework to evaluate the viability of the P2P EA approach in the following chapters. In addition, it provides some keys on how the algorithm performance will be influenced by the underlying computing platform and the specific features of the problem to be tackled.

Going back to the model, it consists of a fine-grained spatially-structured EA (that will be detailed in Section 4.1) in which every agent schedules the evolution process of a single individual and self-organises its neighbourhood via the newscast protocol. As explained in

Section 3.2, newscast runs on every node and defines the self-organising graph that dynamically maintains constant some graphs properties such as a low average path length or a high clustering coefficient from which emerges a small-world behaviour.

This makes the algorithm inherently suited for parallel execution in a peer-to-peer fashion which, in turn, offers great advantages when dealing with computationally expensive problems because distributed execution implies a speedup of the algorithm. In this sense, Section 4.2 provides a first insight on the computational performance of the algorithm from the perspective of the execution time. It considers the limitations of our approach when running either locally to a computer or in a parallel infrastructure:

- Given that every EvAg can be potentially scheduled within a thread, we show in Section 4.2.1 how a multi-threading population is able to scale seamlessly in desktop computers without any effort from the programmer. To this aim, we measure how the algorithm speed scales by conducting experiments in a Single and a Dual-Core Processor architectures.

- Section 4.2.2 shows that, in terms of computing speedup, the parallel performance of the approach mainly depends on the fitness evaluation cost and the underlying computing platform. For very demanding problem instances and high performance computer architectures, the EvAg model is estimated to hold linear speedups up to thousands of processors.

Such results help understanding the performance of the approach in terms of execution time but just considering simplified models of real-world applications. It will be within the following chapters of this thesis where the viability of the approach is tackled in detail from an algorithmic perspective.

4.1 The Evolvable Agent Model

Given that the EvAg model is an agent system, different EvAgs might implement independent strategies of evolution e.g. using different operators, following different evolutionary schemes or self-adapting their own schedule at run-time as in [76]. Nevertheless, we have adopted a symmetric implementation of the agents in order to simplify the study of viability of the approach. This way, EvAgs will follow within this thesis the same evolutionary scheme of Cellular Evolutionary Algorithms (CEAs) explained in Section 2.1.3.

Algorithm 3 shows the pseudo-code of an $EvAg_i \in [EvAg_1 \dots EvAg_n]$ where $i \in [1 \dots n]$ and n is the population size. Despite the model not having a population in the canonical sense, neighbours EvAgs provide each other with the genetic material that individuals require to evolve.

Every agent acts at two different levels; the evolutionary level for carrying out the main steps of evolutionary computation (selection, variation and evaluation of individuals [19]) and the neighbour level that has to adopt a neighbour policy for the population structure, in this thesis such a policy will consist in the newscast protocol:

- **Evolutionary level.**

  The key element at this level is the locally executable selection. Crossover and mutation never involve many individuals, but selection in EAs usually requires a comparison among all individuals in the population. In the EvAg model, the mate selection takes place locally within a given neighbourhood where each agent selects the current individuals from other agents (e.g. Ind_{actual_h} and Ind_{actual_k} in Algorithm 3).

  Selected individuals are stored in a $Pool_i$ ready to be used by the recombination and mutation operators. Within this process a new individual $Ind_{new'_i}$ is generated.

  In the current implementation, the replacement policy adopts a replace if worst scheme, that is, if the newly generated individual $Ind_{new'_i}$ is better than the current one Ind_{actual_i}, Ind_{actual_i} becomes $Ind_{new'_i}$, otherwise, Ind_{actual_i} remain the same for the next iteration.

  Finally, every EvAg iterates till a termination condition is met.

- **Neighbour Policy: Newscast.**

  In principle, our method places no restrictions on the choice of a population structure, however, such a choice will have an impact on the dynamics of the algorithm. In this thesis, the newscast protocol is considered as neighbourhood policy because, this way, the model is suitable for a P2P execution. The main reasons for the choice of such a protocol have been pointed in Chapter 3 and can be summarised by the robust an scalable behaviour of the approach [73]. In addition, the small-world features emerging from the collective dynamics of the protocol has been shown in Section 2.2 to induce similar environmental selection pressures on the algorithm than panmictic populations, however, scalability is much better at the smaller node degree of small-world population structures as can be observed in Figure 4.1.

Algorithm 3 Pseudo-code of an Evolvable Agent ($EvAg_i$)

```
Evolutionary level

$Ind_{actual_i} \Leftarrow$ Initialise Agent
while not termination condition do
    $Pool_i \Leftarrow$ Local Selection($Neighbours_{EvAg_i}$)
    $Ind_{new_i} \Leftarrow$ Recombination($Pool_i$,$P_c$)
    $Ind_{new'_i} \Leftarrow$ Mutation($Ind_{new_i}$,$P_m$)
    Evaluate($Ind_{new'_i}$)
    if $Ind_{new'_i}$ better than $Ind_{actual_i}$ then
        $Ind_{actual_i} \Leftarrow Ind_{new'_i}$
    end if
end while

Local Selection($Neighbours_{EvAg_i}$)
$[Ind_{actual_h} \in EvAg_h, Ind_{actual_k} \in EvAg_k] \Leftarrow$ Random selected nodes from $Cache_i$

Neighbour Policy: Newscast

Active Thread
loop
    wait $t_r$
    $EvAg_j \Leftarrow$ Random selected node from $Cache_i$
    send $Cache_i$ to $EvAg_j$
    receive $Cache_j$ from $EvAg_j$
    $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_j$)
end loop

Passive Thread
loop
    wait $Cache_j$ from $EvAg_j$
    send $Cache_i$ to $EvAg_j$
    $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_j$)
end loop
```

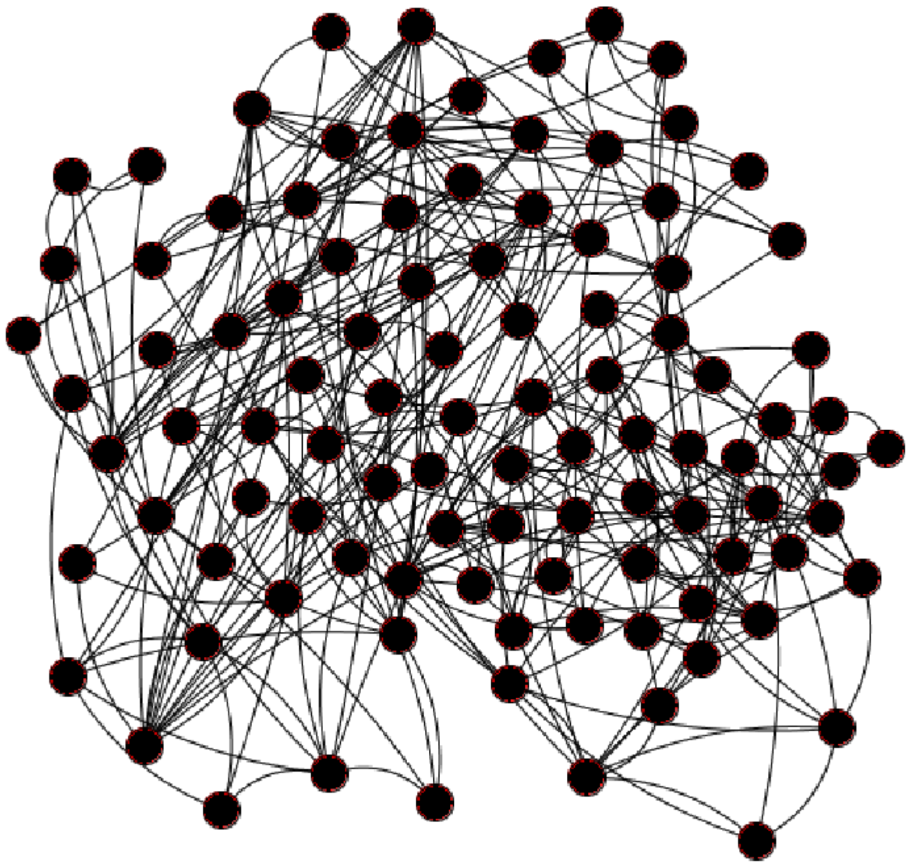


Fig. 4.1 — A snapshot of a newscast population structure for $c = 4$ and $n = 100$ (direction of the edges has been removed). It is straightforward to see that the node degree in newscast is smaller than in the panmictic case that would be represented by a complete graph with $\frac{n(n-1)}{2}$ edges.

4.2 Practical issues

As previously explained, this thesis focuses on studying the viability of the EvAg model from an algorithmic point of view. Nevertheless, this Section tries to provide some keys for the interpretation of results from a real-world application perspective in which the dominant time factor for an EA uses to be the fitness evaluation. Therefore, the following sections analyse the limitations on local and parallel performances imposed by the fitness evaluation cost and conditioned by the underlying computing platform.

4.2.1 Local performance

Locally to a computer, every EvAg can be scheduled within a thread and dispatched by the operative system. The multi-threading nature of the model implies an impact on the local throughput, expressed as:

$$Throughput_{EA} = \frac{Computational \quad Effort}{Time} \tag{4.1}$$

where Computational Effort is usually understood in EC as the number of fitness evaluations.

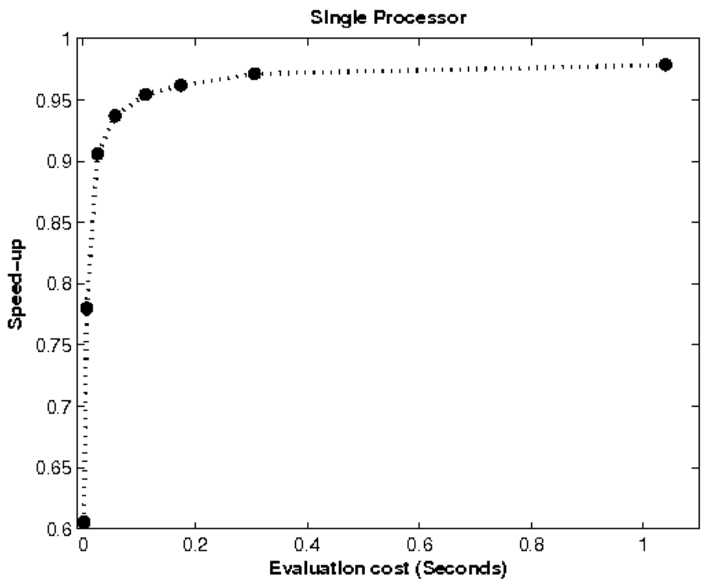
Either the context exchange of the threads or the mutual exclusion mechanisms have a computational cost which is avoid in sequential approaches.

Nevertheless, Symmetric Multiprocessing (SMP) architectures are becoming nowadays very popular as desktop machines and sequential approaches are unable to take advantage of more than a single processor. This way, the local performance of the EvAg model can be assessed by measuring the speedup in the throughput with respect to a sequential GA (sGA):

$$Speedup = \frac{Throughput_{EvAg}}{Throughput_{sGA}} \tag{4.2}$$

To this end, the computational cost of the evaluation function (i.e. the independent variable in the throughput equation) is scaled from few milliseconds to one second.

Figure 4.2 shows that the throughput speeds up asymptotically, having a limit on the number of processors. Therefore, the performance in single processor machines tends to be equivalent to sequential approaches as the evaluation cost increases while is clearly outperformed in SMP machines. An additional advantage of the EvAg model is that the load balance at this level is transparent for the programmer since it is carried out by the operative system.



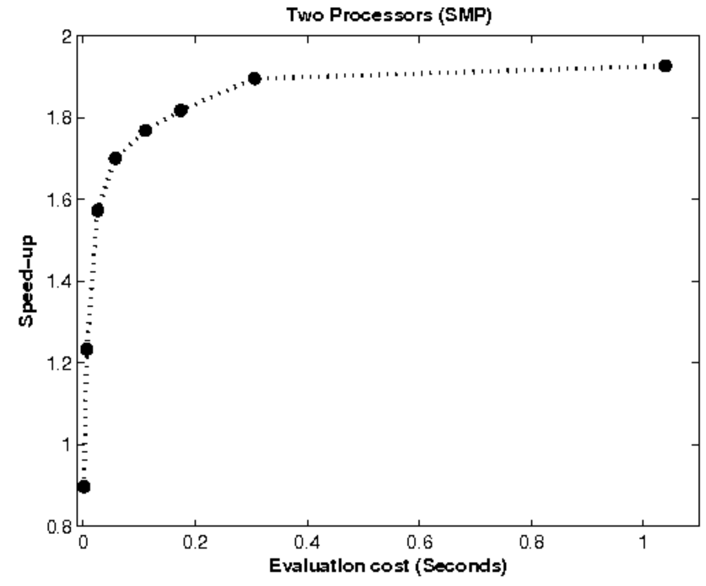


Fig. 4.2 — The figure depicts how the EvAg throughput speeds up with respect to the sGA one when the evaluation function cost scales for a population size of 400 individuals. The test-bed is a single processor (*upper figure*) and a dual-core processor (*bottom figure*).

4.2.2 Parallel performance

In parallel infrastructures, the speedup is a commonly used metric to measure the parallel performance of a distributed algorithm over the sequential one. Gagné et al. define in [27] the speedup of a distributed EA assuming that the fitness evaluation is the main consuming task:

$$Speedup = \frac{NT_f}{T_p} \tag{4.3}$$

where N is the population size, T_f is the time needed to evaluate the fitness of a single individual and T_p is the time needed to evaluate all the individuals using P processors.

In the case of a P2P system, every individual is placed in a single processor and therefore $P = N$ and $T_p = T_N$. Then:

$$T_N = \underbrace{T_f}_{computation} + \underbrace{\frac{N}{m} T_{comm}}_{communication} + \underbrace{T_{lat}}_{latency} \tag{4.4}$$

where T_{comm} is the communication time required by every EvAg at each evolutionary cycle and m is the number of hub/gateways of the physical network from which all the network traffic is dispatched. Being T_{ind} the time to transmit a single individual, $T_{comm} = 2T_{ind}$ in the case of using a selection operator such as binary tournament. Finally, the latency time, T_{lat}, will depend on the infrastructure of the physical network. This way, having a single queue gateway $M/M/1$ (as depicted in Figure

4.3), the latency time can be explained as the mean response time of the gateway [40]:

$$T_{lat_{M/M/1}} = \frac{\underbrace{\frac{1}{\mu}}_{throughput}}{1 - \underbrace{\frac{\lambda}{\mu}}_{utilisation}} \tag{4.5}$$

where λ represents the incoming traffic rate and μ the throughput rate. This way, the traffic intensity (or utilisation of the gateway) can be expressed as $\rho = \frac{\lambda}{\mu}$ and the latency time extended to m gateways following a $M/M/m$ scheme:

$$T_{lat_{M/M/m}} = \frac{1}{\mu}(1 + \frac{\rho}{m(1-\rho)}) \tag{4.6}$$

Under such a framework, the following experiment aims to provide a guideline on how the fitness evaluation cost and the underlying computing platform may influence the parallel performance of the EvAg approach.

According to Thierens in [70], the population size of an EA (N) roughly scales with an order $O(l^{\alpha})$, where l is the chromosome length in bits and α is a constant that depends on the algorithm and the problem complexity. Therefore, we can analyse the parallel performance of our P2P EA model under the following assumptions by scaling $l \in [1 \ldots 100]$:

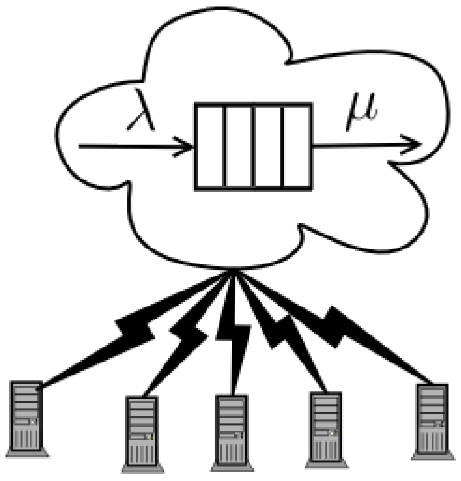


Fig. 4.3 — Scheme of a M/M/1 queueing system. λ stands for the incoming traffic rate and μ for the throughput rate.

- Since N scales with an order $O(l^{\alpha})$, we have set $N = l^{\alpha}$. Taking into account the empirical results on the Thierens paper, a problem of intermediate difficulty can scale with $\alpha = 2$.

- We assume that $T_{ind} = l$, meaning that the time to transmit an individual through the network scales linearly with respect to the chromosome length.

- Given that λ represents the incoming traffic rate in a gateway, it can be explained by $\lambda = \frac{N}{m} 2T_{ind}$ if we assume a perfect load balancing of all communications through the m gateways.

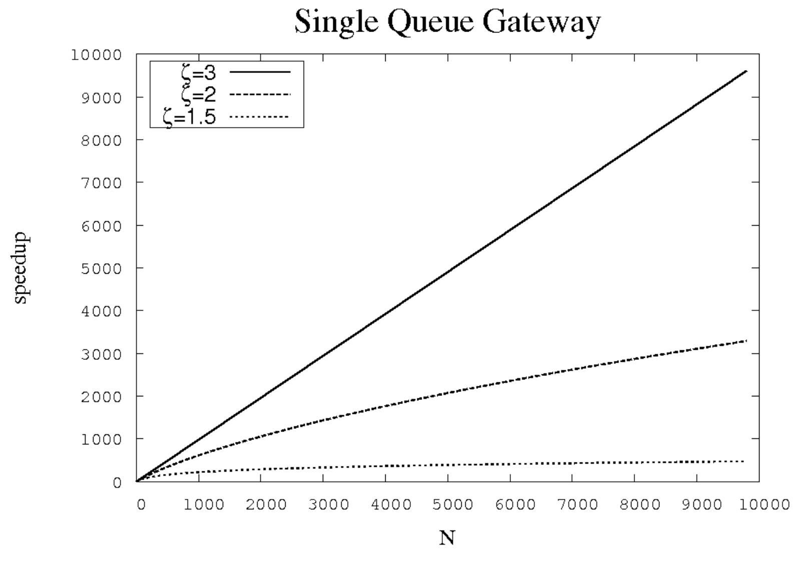
- μ has been adjusted to 20000 in such a way that the single gateway queueing $M/M/1$ would suffer overflow beyond $l = 100$.

- To analyse the influence of the fitness evaluation cost and the network infrastructure on the speedup, we have considered six possible scenarios for both variables:

- The fitness evaluation cost is obviously a problem-dependent variable. In real-world problems, the complexity orders tend to be high. Therefore, by simply assuming polynomial orders of complexity $O(l^{\zeta})$, the fitness evaluation can be expressed as $T_f = l^{\zeta}$. This experiment consider $\zeta = [\frac{3}{2}, 2, 3]$ which are rather optimistic values considering real-world problems.
- In order to assess the influence of the physical network on the parallel performance, we consider two possible infrastructures, the first using a single queue gateway $m = 1$ (as it could be the case in a LAN) and the second one using a eight queueing system $m = 8$.

Figure 4.4 shows the maximum speedup that the P2P approach can reach on every of the six scenarios under study. Beyond the numerical fit of the assumptions, results provide a qualitative estimation on how the fitness evaluation cost and the underlying computing platform influence the performance.

On the one hand, the scalability of the fitness evaluation cost is key to hold linear speedups. In addition, not all the problem instances but those requiring larger genomes can apply for massive parallelisation. In this sense, the scalability of the population size impose a limitation on the number of processors to be used since $P = N$ at a higher bound. On the other hand, the underlying architecture plays an important role on



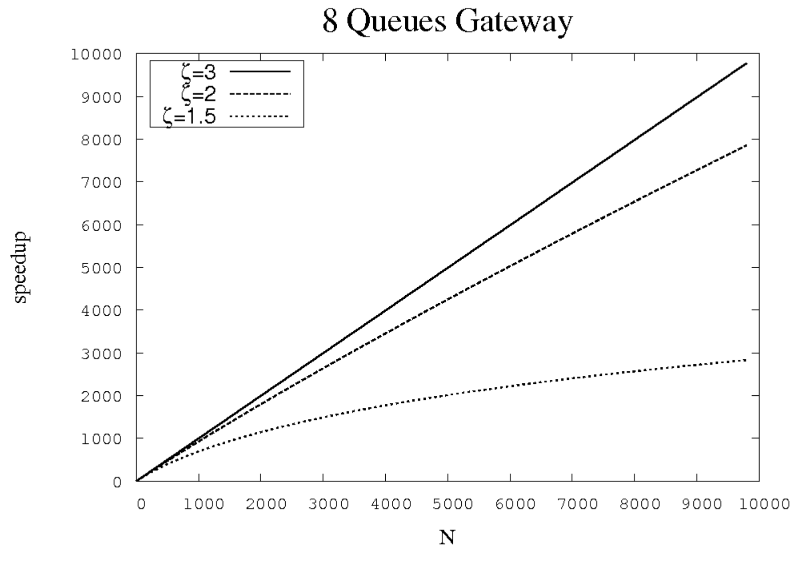


Fig. 4.4 — Speedup curves for $\zeta = [\frac{3}{2}, 2, 3]$ as a function of the population size (or alternatively the number of processors since $N = P$). Results are obtained for a single queue gateway architecture (*up*) and a 8 queues gateway system (*down*).

the communication time and therefore on the overall performance. It can be seen how the problems scaling with sublinear speedups (e.g. for $\zeta = 2$) perform much better for infrastructures with a higher capacity.

4.3 Summary

The Evolvable Agent model is a fine-grained spatially-structured EA composed of a population of Agents. Every EvAg acts at two levels; evolving a single individual with the mate selection locally restricted within a neighbourhood and dynamically structuring such a neighbourhood by means of the protocol newscast. This makes the approach suitable for a P2P execution in which every EvAg can be potentially placed in a different peer. In such a loosely-coupled scenario, the execution of the model is decentralised and lacks of a central clock which implies the asynchronous evolution of the individuals.

Inherent to the model is that such an asynchronous execution allows to take a seamless advantage of the several processors in SMP computers, outperforming, this way, the throughput of a sequential EA for demanding fitness evaluation functions. Additionally, the parallel performance of the model has been shown to scale up to thousands of computers with a linear speedup. Nevertheless, it has to be taken into account that such a case would require of a highly expensive fitness evaluation, a large number of individuals and a parallel infrastructure able to support all the network traffic.

Therefore, to this point, massive scalability of the EvAg model remain in an idealised scenario that do not account for the algorithmic issues of the approach itself, such as the asynchrony, nodes failures or dynamic changes on the population structure. This way, the following chapter will tackle the *algorithmic performance* of the model from the perspective of the experimental analysis.

Chapter 5

Experimental Analysis

In the previous chapter we have presented the EvAg model as the framework in which this thesis will analyse the viability of the P2P EC paradigm. In addition, a first insight on the *computational performance* of the approach has been provided showing that, for very demanding problems, it is able to scale holding linear speed-ups.

Nevertheless, such results do not take into account whether the algorithm is able to converge to good solutions in spite of the run-time dynamics of P2P systems. Hence, in this chapter, we propose the experimental analysis of the model in a simulated P2P environment so that the viability of the P2P EA can be drawn from the *algorithmic performance* of the approach. All the source code for the experiments has been published under a GPL v3 license and is available from our Subversion repository at https://forja.rediris.es/svn/geneura/evogen.

Simulations will allow the execution of controlled experiments tackling the goals described in Section 5.1 in which a set of representative test-cases are proposed for studying the viability of the model. In order to define the system with a good level of detail, Section 5.2 describes the decision-making process that we have followed for the experimental analysis. Taking into account such decisions, Section 5.3 presents the overall methodology that will be followed in the test-case 1 (Section 5.4), test-case 2 (Section 5.5) and test-case 3 (Section 5.6). Finally, conclusions on the *scalability* and *fault tolerance* of the model are drawn in Section 5.7 based on the analysis of results.

5.1 Goals

As exposed at the introduction of this thesis, one of the main motivations behind P2P EAs is tackling those large problem instances in which, due to memory or computational constraints, sequential approaches are unsuitable. In this sense, analysing the scalability of the EvAg model is key to determine the viability of the approach. Additionally, as the sizes of the problem instances increase, the number of computing nodes required to tackle the problem scales and, therefore, failures become more likely at run-time. Taking that into account, we propose an experimental analysis focused on the following goals in order to prove the viability of the model:

1. Analysing the *scalability* of the model to demonstrate that the approach is suitable for tackling large problem instances in a

failure-free environment.

2. Studying the *fault-tolerance* of the model under churn conditions to demonstrate that the model follows a *graceful degradation* and is able to tackle large problem instances in spite of nodes departing from the system.

Studying the scalability makes possible not only to analyse instances under examination but also predicting the model behaviour when tackling larger instances. On the other hand, fault tolerance is a key issue in a P2P EA since churn is inherent to P2P systems given that peers are prone to failures.

In order to tackle such goals, the following test-cases have been designed to assess the algorithm performance:

1. *Scalability of the model in failure-free environments against sequential approaches.* The EvAg model is compared against a canonical SSGA and a GGA to demonstrate that such a spatially-structured EA scales better than panmictic schemes of evolution.

2. *Influence of the population structure on the algorithm performance.* As explained in Section 4.1, there is no reason preventing the EvAg model to use population structures other than newscast. Therefore, this test-case aims analysing the scalability of the model using two common topologies in fine-grained approaches: a ring lattice [30] and a Watts-Strogatz [32] population structures.

3. *Fault tolerance of the model.* In this test-case, we will analyse the scalability of the model using several scenarios of churn in which the system degrades under different failure rates.

Before proceeding with the experimental analysis, the following section describes all the decisions related with the implementation of the different test-cases.

5.2 Rationale for the experimental analysis

Given the huge complexity of either P2P systems or EAs, a detailed analysis on their interactions in the P2P EC paradigm remains beyond the scope of this thesis. Hence, a certain number of decisions has to be made in order to focus the analysis on the specifications of the goals. Therefore, this Section aims to justify the simplifications and assumptions made on the model.

5.2.1 Simplifications

In order to simplify the analysis on the model performance, the following considerations were made on the design of experiments:

- The experimental analysis in this thesis will concentrate on *binary-coded GAs* [55], excluding the rest of EC paradigms such as ES

[59], EP [22] or GP [46]. The main reason for this restriction is that the population sizing theory [35] that we will follow to perform analysis on the scalability of the algorithm focuses on such a kind of EAs. Nevertheless, taking into account that *binary-coded GAs* follow the same evolutionary scheme than other EAs such as *real-coded GAs* [20] or GP, conclusions should be easily extended to other paradigms.

- With the aim of establishing a worst-case analysis, evolutionary operators will not apply specific knowledge on the problem. In this context, we will use *uniform crossover*, *bit-flip mutation* as operators and *binary tournament* as decentralised selection mechanism throughout all the experiments [19]. Taking into account that we will use trap functions as benchmark (described in Section 5.3.1), either *uniform crossover* or *bit-flip mutation* prevent the algorithm search to form higher order BBs, thus challenging the GA's search mechanisms. On an averaged sense, such operators are fooled by traps as described by Deb and Goldberg in [15].

- In the same line as the previous point, individuals will be *initialised at random*. Every gene will have an uniform probability of 0.5 of being either 1 or 0, so that, on average, the randomly generated initial population is placed on local optimum attracting regions for the problem landscapes proposed in Section 5.3.1.

5.2.2 Assumptions

The experimental analysis in this chapter examines the main variables influencing the EvAg performance. To that aim, experiments were designed in such a way that variables under study can be analysed by assuming fixed conditions for the rest of factors. Such assumptions are detailed in the following points:

- We have used selectorecombinative versions of the algorithms (without mutation) for estimating the population sizes. Lobo and Lima state in [53] that the assumption of a selectorecombinative GA is commonly made in population sizing studies. That way, the only source of diversity is then the initial population which stands for a worst case analysis. However, this thesis complements the study analysing the convergence of the approach using mutation.
- We assume that every EvAg behaves as a *virtual node* [16] in such a way that every physical node can host more than a single EvAg. Hence, the number of *virtual nodes* hosted in a physical one can be decided at a load-balancing level depending on node capacities and a heterogeneous system such as P2P can be assumed as homogeneous at a virtual level.
- Despite having assumed homogeneous conditions in the previous point, the lack of a central clock in a decentralised scheme implies an asynchronous execution. In this sense, the most commonly used methods to simulate asynchrony either in cellular au-

tomaton [63] or cellular evolutionary algorithms [71] can not be applied in this case except for the *Uniform choice* policy, more appropriate to model a decentralised run in which there is no guarantee of any sequential order in the update of the individuals. This way, we have adopted such a policy to simulate the asynchronous update of EvAgs.

- With respect to the update frecuency of an EvAg , it has to be consider that there are two independent tasks running in parallel within every agent, the evolutionary loop and the newscast protocol. In this sense, we have assumed that every time an EvAg performs a fitness evaluation, it also initiates a cache exchange of the newscast protocol, i.e. $t_r = cycle$ where t_r is the parameter for the newscast updating frequency exposed in Section 3.2 and *cycle* is defined as the time the algorithm takes to perform n evaluations in a population of n EvAgs using *uniform choice*.

- As it has already been shown in Section 3.2.2, the newscast protocol takes at around 12 cycles to converge to an state of dynamic equilibrium for a network size of 1600 which is the maximum size that we have used here. In this sense, the evolutionary algorithm will begin at $t_r = 20$ in order to guarantee that the protocol bootstraps and converges. We have assumed a synchronised start up of the experiments.

- The cache size (c) is the only tunable parameter in newscast and we use $c = 20$ within all the settings of the experiments. Such value takes into account Jelasity and van Steen recommendations in [43] stating that the intended normal setting of newscast is $c \ll n$ and demonstrating that values from $c = 20$ prevent the spontaneous partitioning of the graph even when it becomes very large (see Section 3.2.3 for more details). In addition, we make experiments for the fine-tuning of the parameter in the appendix A of this thesis showing a lack of influence of c on the EvAg performance when $c \in [0.01n, 0.16n]$, where n is the population size.

- We have assumed that the time required for communications is negligible with respect to the fitness computing time. Such an assumption might be unrealistic for small problem instances, but it turns feasible for problem instances becoming large, as it has been shown in the analysis of the parallel performance of the model in Section 4.2.2, communications do not inflict a penalisation on the algorithm speed-up for problems with a very demanding fitness evaluation cost.

Once assumptions have been laid out, we will continue describing the experimental methodology next.

5.3 Experimental Methodology

In order to analyse the scalability of the EvAg model, experiments are conducted on trap functions (described in Section 5.3.1). The purpose is investigating how population sizes scale with increasing problem size and difficulty. To this end, we follow the population sizing theory [35] which states that there is an optimal population size for a given problem instance that can be determined under some conditions. In particular, we use the bisection method (which will be explained in Section 5.3.2) that determines the minimum population size N for a selectorecombinative GA. In addition, we look at the scale-up properties of the average number of evaluations to a solution (AES), which is an algorithmic independent measure of computational effort that can be used for all EAs, with or without mutation [19]. Finally, the study is complemented switching mutation on, so that the algorithm convergence is analysed for the most demanding problem instances using the metrics that will be proposed in Section 5.3.3. Such results will be statistically analysed using the non-parametric Wilcoxon test [28] within the following Sections.

5.3.1 Test-suite for the experiments

Experiments were conducted on deceptive, quasi-deceptive, and non-deceptive trap functions [2] following Lobo and Lima's recommendations in [53] about choosing a test suite with known population requirements and investigating the scalability on landscapes with different characteristics. These functions represent a set of decomposable problems based on unitation and composed of sub-functions in which the total fitness is additively calculated by summing the partial fitness of every sub-function. Hence, it is easy to scale the problem from small to large instances by considering a smaller or larger number of sub-functions. In addition, each sub-function is composed of (k) bits representing the BB size, every sub-function is then composed of 2^k combinations from which only one belongs to the optimal solution. Considered values include $k = 2$, $k = 3$ and $k = 4$ (forming non-deceptive, quasi-deceptive and deceptive problems respectively). Studying the scalability for different settings of k can offer a better understanding of the model since not only the scalability is analysed but also how the scalability changes as the problem difficulty increases.

There are two distinct regions in the search space of trap functions, one leading to a global optimum and the other leading to the local optimum (see Figure 5.1). In general, a trap function is defined by the following equation:

$$trap(u(\overrightarrow{x})) = \begin{cases} \frac{a}{z}(z - u(\overrightarrow{x})), & \text{if} \quad u(\overrightarrow{x}) \leq z \\ \frac{b}{l-z}(u(\overrightarrow{x}) - z), & \text{otherwise} \end{cases} \tag{5.1}$$

where $u(\overrightarrow{x})$ is the unitation function, a is the local optimum, b is the global optimum, l is the problem size and z is a slope-change location separating the attraction basin of the two optima.

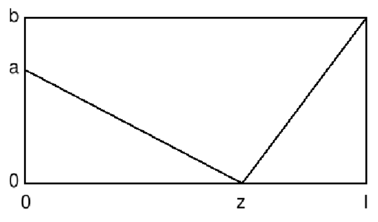


Fig. 5.1 — Generalised *l-trap* function.

For the following experiments, 2-trap, 3-trap and 4-trap functions were designed with the following parameter values: $a = l - 1$, $b = l$, and $z = l - 1$. With these settings[1], 2-trap is not deceptive, 4-trap is deceptive and 3-trap lies in the region between deception and non-deception. Under these conditions, it is possible not only to examine the scalability on trap functions, but also to investigate how the scalability varies when changing from non-deceptive to deceptive search landscapes. Scalability tests were performed by juxtaposing m trap functions in binary strings of length L and summing the fitness of each sub-function to obtain the total fitness.

5.3.2 A method for estimating the population size

Sastry proposes in [62] a method based on bisection to estimate the optimal population size N to solve a problem instance, that is, the lowest N for which 98% of the runs find the problem optimum. To this end, a selectorecombinative GA is used to search the minimum population size such that using random initialisation it is able to converge to the optimum without any other mechanism than recombination and selection.

Algorithm 4 depicts the method based on bisection. The method begins with a small population size which is doubled until the algorithm ensures a reliable convergence. After that, the interval (min, max) is halved several times and the population size adjusted within such a range until $\frac{max-min}{min} > threshold$, where min and max stand respec-

[1]Originally, Ackley's trap functions use $z = \frac{3l}{4}$, however, Deb and Goldberg demonstrate in [15] that trap functions are fully easy under such settings.

Algorithm 4 Population tuning algorithm based on bisection

```
$N$ = Initial Population Size (20)
while GA reliability ($N$) $< 98\%$ do
  $min = N; max, N = 2N$
end while
while $\frac{max-min}{min} > \frac{1}{16}$ do
  $N = \frac{max+min}{2}$
  if GA reliability($N$) $< 98\%$ then
    $min = N$
  else
    $max = N$
  end if
end while
```

tively for the minimum and maximum population size estimated and *threshold* for the accuracy of the adjustment within such a range. This parameter has been set to $\frac{1}{16}$ in order to obtain a reasonable adjustment of the population size, e.g. the algorithm would estimate a population size of $N = 310$ if we consider an optimal size of $N = 315$.

5.3.3 Metrics

The following metrics will be used for assessing the performance of the model in the experimental analysis. To allow the comparison of results with those in the literature such metrics were chosen to be standard in EAs. Either Tomassini in [71] or Eiben and Smith in [19] make an extensive revision of them.

- The success rate (SR) measures the algorithm quality as the proportion in which the algorithm is able to find the problem optimum out of all the runs.

- The average number of evaluations to solution (AES) stands for the number of evaluations that the algorithm spends in those runs that yield success. Since a preliminary analysis on the normality of results shows that they do not follow a normal distribution, we have chosen the number of evaluations to solution in the third quartile ($AESQ_3$) as a baseline value, meaning that 75% of the runs will stay below such a value.

- The Mean Best Fitness (MBF) is used to depict the algorithm convergence as the averaged values of the best fitness.

- The genotypic distance entropy (GE) is a measure of the population diversity defined on the genotypic distances ($H_g(P)$).

$$H_g(P) = -\sum_{j=1}^{N} g_j log(g_j) \tag{5.2}$$

where g_j is the fraction $\frac{n_j}{N}$ of individuals in P having a Hamming distance j to a genotype of reference (we have used the optimal genotype to that end), and N is the number of different distances.

5.3.4 Summary

This section has presented the experimental methodology that will be followed for analysing the viability of the EvAg approach. To that aim, we will have to prove that the algorithm is *scalable* and *fault-tolerant* in a simulated P2P environment.

In order to tackle such goals, we will perform an experimental analysis of the three test-cases summarised in Table 5.1. In every case, the analysis will follow the same methodology, consisting in the study of the algorithmic scalability, the analysis of the algorithm convergence and population diversity at run-time and a comparison of the results to show whether they present statistical differences or not.

Test-Case	Methodology
1 Scalability in a failure-free environment vs. sequential GAs	1.1 Scalability selectorecombinative GA 1.2 Analysis of the convergence of the larger problem instance 1.3 Statistical analysis
2 Influence of the population structure on the algorithm performance	2.1 Scalability selectorecombinative GA 2.2 Analysis of the convergence of the larger problem instance 2.3 Statistical analysis
3 Fault tolerance of the model	3.1 Scalability selectorecombinative GA 3.2 Analysis of the convergence under different churn scenarios 3.3 Statistical analysis

Table 5.1 — Summary of the experimental methodology.

5.4 Test-Case 1: Scalability of the model in failure-free environments

In order to investigate the scalability of the EvAg model, experiments were conducted on different trap functions and compared against two canonical GAs, a steady-state GA (SSGA) and a generational GA (GGA). Whereas the population structure of the EvAg model is defined by the newscast protocol, the canonical GAs are panmictic. Following the experimental methodology of Section 5.3, two series of experiments were conducted:

In the first series that will be presented in Section 5.4.1, experiments use selectorecombinative versions of the GAs to estimate optimal population sizes for the different problem instances. The reason for using selectorecombinative GAs is that there are well defined models to establish the population size and the number of evaluations required to solve a given trap function instance [37], however, to the best of our knowledge there are no such models when using mutation. To this end, in Section 5.3.2 we have described a method for estimating the population size. The underlying idea is that without mutation, the population size becomes the only source of diversity. In this context, Thierens demonstrates in [70] the necessity of larger population sizes when tackling larger problem instances.

In the second series presented in Section 5.4.2, the analysis focuses on the performance of the different approaches when they are equally parametrised. Specifically, large instances of 2-Trap, 3-Trap and 4-

Trap are considered to analyse the convergence of the fitness and the evolution of genotypic diversity. Finally, such results are statistically analysed in order to determine whether the difference in performances are significant or not.

5.4.1 Scalability analysis

In this first series of experiments, different approaches have no mutation in order to meet the selectorecombinative criterion of the bisection method. All settings are summarised in Table 5.2 taking into account decisions in Section 5.2 about choosing operators that do not take advantage of the problem structure or setting the cache size. Since the methodology imposes a SR of 0.98 in the results, the $AESQ_3$ has been used as an appropriate metric to measure the computational effort to reach the success criterion. A more efficient algorithm will need a smaller number of evaluations.

Trap instances	
BB size	$2, 3, 4$
Individual Length (L)	$12, 24, 36, 48, 60$
GA settings	
GA	selectorecombinative SSGA
	selectorecombinative GGA
	selectorecombinative EvAg
Population size	Tuning algorithm
Selection of Parents	Binary Tournament
Recombination	Uniform crossover, $p_c = 1.0$
Newscast settings	
Cache size	20

Table 5.2 — Test-case 1: Parameters of the experiments in the analysis of scalability.

Figure 5.2 depicts the scalability of the population size (N) and the required number of evaluations to reach the problem optimum ($AESQ_3$) for the three approaches under study on 2, 3, and 4-trap functions. All the graphics show that either the population size or the computational effort fit with a potential order of scalability with base the length of the chromosome (L) and different exponents depending on the problem difficulty and the approach itself.

The first conclusion that can be easily drawn from results is a better scalability of the EvAg approach with respect to the population size, specially when the problem difficulty increases from 2 to 4-trap. That is, increasing the problem difficulty makes the GGA and SSGA face extreme difficulties to track the problem optimum, thus requiring a higher population size N to prevent that the algorithm gets stuck in local optima. From a computational perspective, this fact can be translated into a more efficient use of the running platform since the EvAg approach will require a smaller amount of computational resources. Additionally, results on $AESQ_3$ are clearly correlated to the population size, SSGA scales better than GGA and it is roughly similar to EvAg in 2 and 3-trap. Nevertheless, as the problem difficulty increases to 4-trap, EvAg scales clearly better.

In order to gain some insight on the influence of mutation in the scalability order, we consider a more realistic GA set-up by switching mutation on. This implies that we have to specify values for the mutation rate parameter p_m. Strictly speaking, we should also recalibrate population sizes, since the bisection method only gives good estimates for selectorecombinative GAs. However, an extensive parameter sweep

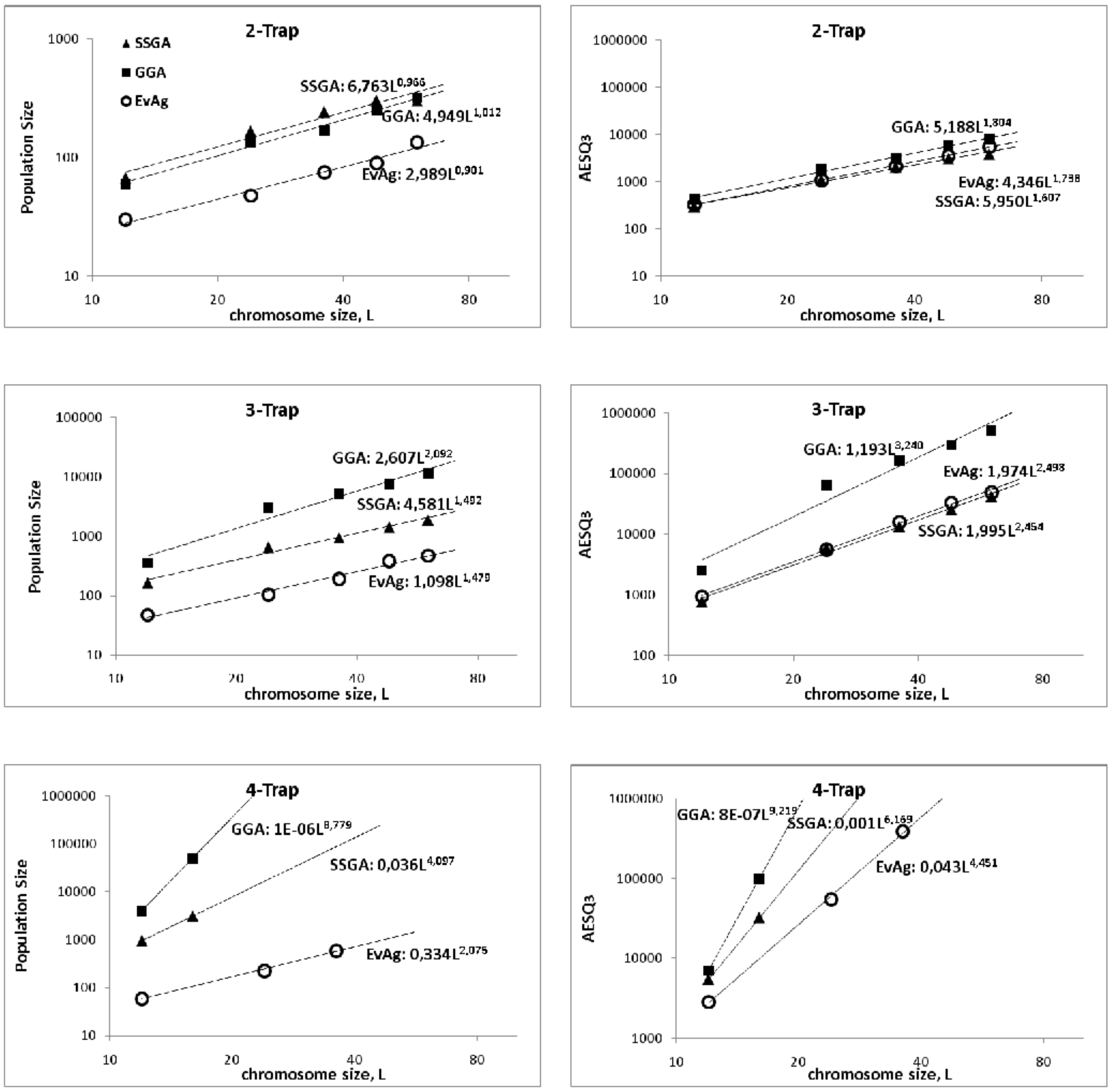


Fig. 5.2 — Scalability in trap functions based on the population tuning algorithm and the selectorecombinative versions of the generational GA (GGA), steady-state GA (SSGA) and the Evolvable Agent (EvAg). On the left the estimated population sizes N and the evaluations to solution in third quartile $AESQ_3$ on the right. Results are obtained by bisection and depicted in a *log-log* scale as a function of the length of the chromosome, L.

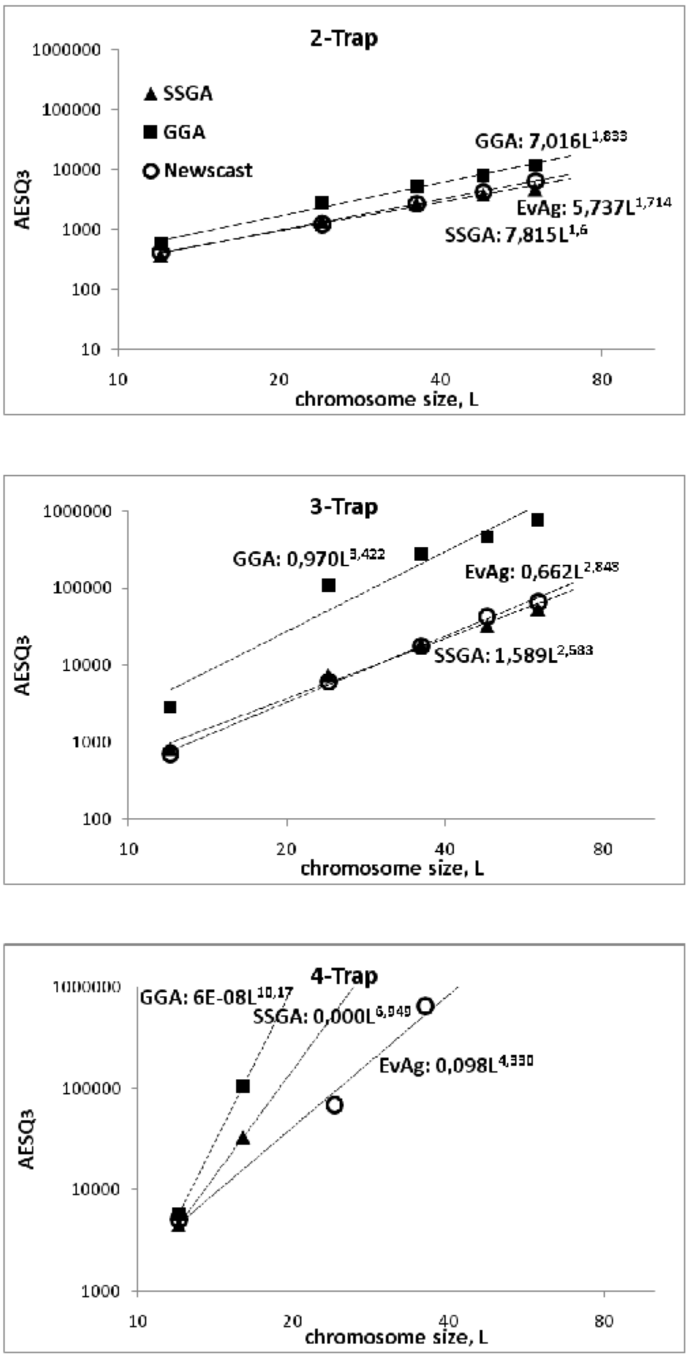


Fig. 5.3 — Reproduction of the results in Figure 5.2 with mutation switched on.

goes far beyond the scope of this thesis and therefore we will use the common "$\frac{1}{L}$ heuristic" for setting the mutation rates and keep the population sizes that were used in the first series of experiments. Obviously, in this case we only need to look at the $AESQ_3$ results. The outcomes of these experiments are shown in Figure 5.3 in which curves appear shifted with respect to the selectorecombinative version in Figure 5.2 but approximately keeping the same scalability order. Table 5.3 compares such estimated complexity orders with mutation switched off and on, showing that mutation does not alter the order in algorithm performance, and exponents are roughly the same, with only the constant in the power law changing.

	GGA		SSGA		EvAg	
	Mutation		Mutation		Mutation	
	off	on	off	on	off	on
2-Trap	$O(L^{1.804}))$	$O(L^{1.833})$	$O(L^{1.607})$	$O(L^{1.6})$	$O(L^{1.738})$	$O(L^{1.714})$
3-Trap	$O(L^{3.24})$	$O(L^{3.422})$	$O(L^{2.454})$	$O(L^{2.583})$	$O(L^{2.498})$	$O(L^{2.843})$
4-Trap	$O(L^{9.219})$	$O(L^{10.17})$	$O(L^{6.169})$	$O(L^{6.949})$	$O(L^{4.451})$	$O(L^{4.33})$

Table 5.3 — Complexity orders of the $AESQ_3$ scalability in O notation.

5.4.2 Analysis of the algorithmic convergence

In the second series of experiments we try to gain more detailed insights in the differences between the three approaches to population management using mutation. To this end, we run the GGA, SSGA, and EvAg models using the same settings on a problem instance as summarised in Table 5.4 (recall that in the previous experiments, GGA, SSGA, and

EvAg used different population sizes on any given problem instance). We choose large instances of each problem under study for this purpose (i.e. $L = 36$ in 4-trap, $L = 99$ in 3-trap and $L = 400$ in 2-trap), where the differences in scalability between the approaches are most visible.

To estimate the population sizes and the maximum number of evaluations in each case, we have used the scalability orders for the selectorecombinative EvAg on the previous series, e.g. $L = 99$ in 3-trap will require a population size of $1.098L^{1.479} = 942$ and a maximum number of evaluations of $1.974L^{2.498} = 190740$. With these settings, switching mutation on implies that such values are oversized for the EvAg approach since mutation represents a new source of diversity, however, the highest scalability orders of SSGA and GGA indicate that such population sizes will remain undersized for both approaches.

The results in Figure 5.4 show that the fitness of the SSGA and GGA approaches stagnates at early stages of the search in the different instances. Despite the SSGA performing a more exploitative search than the GGA, as shown by the genotypic entropy converging to zero, both cases lost track of the optimum, a fact that might be explained by an undersized population size. On the other hand, the evolution of diversity of the EvAg approach indicates that the algorithm converges to the optimum in 2 and 3-trap and is still converging in 4-trap when the termination criterion is met. In this last case, it is straightforward to see that the population size is oversized since a selectorecombinative EvAg is able to find the optimum in such number of evaluations. Given that the three approaches are equally parametrised, it is within the spatially structured scheme of reproduction of the EvAg model where

Trap instances	
2-Trap	
Individual Length (L)	400
Population size	660
Termination Condition	Max. Eval. =144700
3-Trap	
Individual Length (L)	99
Population size	942
Termination Condition	Max. Eval. =190740
4-Trap	
Individual Length (L)	36
Population size	600
Termination Condition	Max. Eval. =393000

GA settings	
GA	SSGA
	GGA
	EvAg
Selection of Parents	Binary Tournament
Recombination	Uniform crossover, $p_c = 1.0$
Mutation	Bit-flip mutation, $p_m = \frac{1}{L}$

Newscast settings	
Cache size	20

Table 5.4 — Test-case 1: Parameters of the experiments for the analysis of convergence.

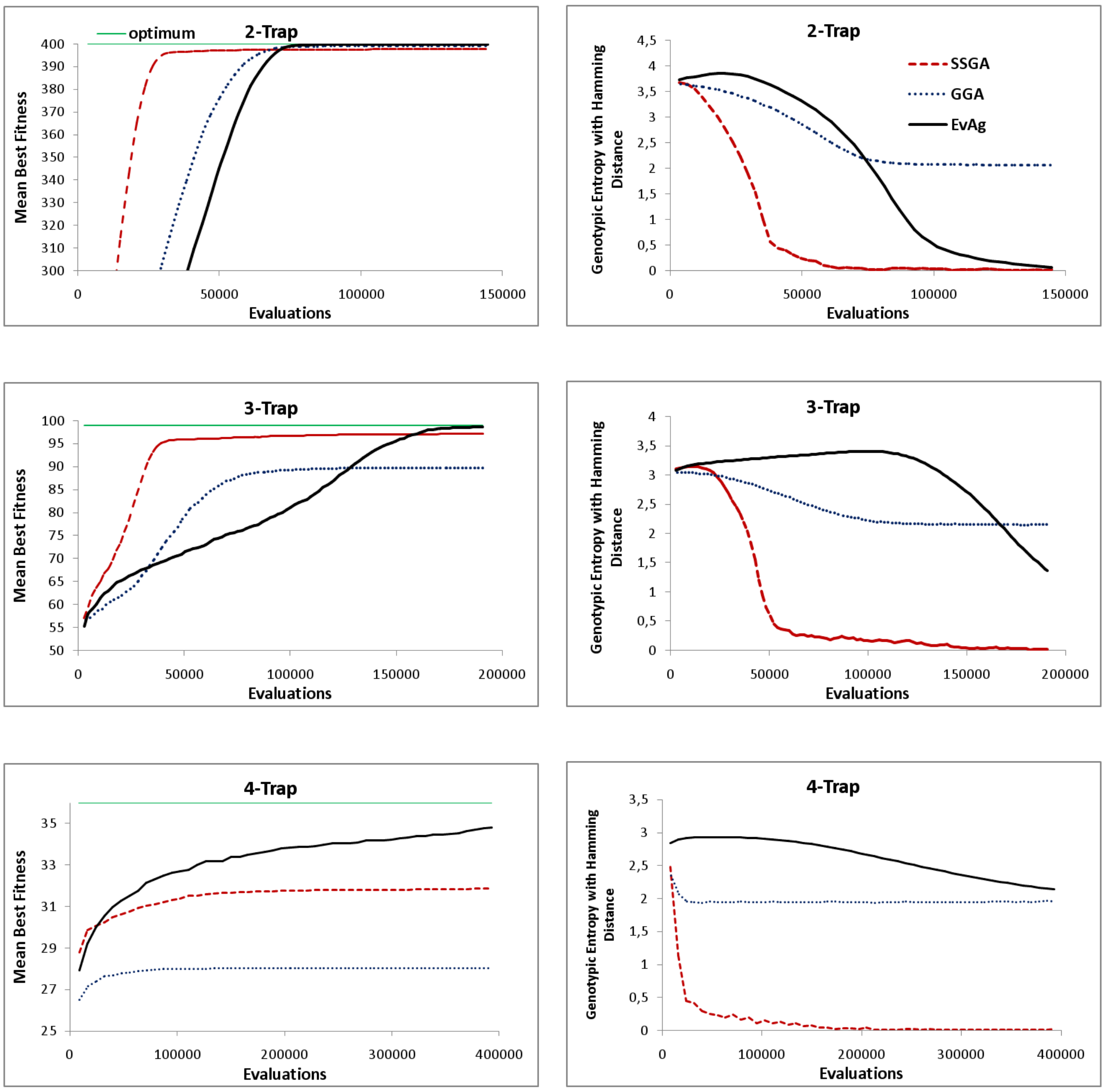


Fig. 5.4 — Test-case 1: Best fitness convergence (*left*) and evolution of the diversity expressed as the entropy based on the Hamming distances between the genotypes (*right*). Graphs plotted represent the average of 50 independent runs.

the genetic diversity is preserved at a higher level and consequently the population size N can be reduced. Hence, the EvAg approach scales better on difficult problems. With a lower optimal N, EvAg needs fewer evaluations to reach the optimum when compared to panmictic GAs.

5.4.3 Statistical analysis

The results shown in the previous Section are analysed in Table 5.5 using the Anderson-Darling test to refute the null hypothesis on the normality of the data. The small p-values at the best fitness distributions show that results do not follow a normal distribution. This way, a non-parametric Wilcoxon test is used to compare the quality of fitness between the EvAg and the SSGA and GGA approaches.

Problem Instance	*Algorithm*	*Anderson-Darling*	*Normal distribution?*
Trap2	GGA	A=3.5 p-value=6.1e-09	no
	SSGA	A=1.8 p-value=6.6e-05	no
	EvAg	A=17 p-value<2.2e-16	no
Trap3	GGA	A=0.9 p-value=0.01	no
	SSGA	A=1.8 p-value=9.9e-05	no
	EvAg	A=11.7 p-value<2.2e-16	no
Trap4	GGA	A=2.4 p-value=2.035e-06	no
	SSGA	A=1.2 p-value=0.002	no
	EvAg	A=12 p-value<2.2e-16	no

Table 5.5 — Anderson-Darling test on the normality of the best fitness distributions. Results are obtained over 50 independent runs. We have considered $p - values > 0.1$ for a distribution to be normal.

Table 5.6 presents the Wilcoxon analysis of the data showing significant differences between the EvAg and canonical approaches in all the problem instances. Therefore, it can be concluded that EvAg outperforms SSGA and GGA when tackling large instances of trap functions.

Problem Instance	*Algorithm*	*Avg. Fitness ±σ*	*Wilcoxon Test*	*Significantly different?*
Trap2	GGA	399.08±1.01	W=624 p-value=1.637e-07	yes
L=400	SSGA	397.8±1.56	W=142 p-value<2.2e-16	yes
N=660				
M. Eval.= 144700	EvAg	**399.92±0.27**	-	-
Trap3	GGA	89.78±2.8	W=0 p-value<2.2e-16	yes
L=99	SSGA	97.16±1.46	W=402 p-value=3.623e-10	yes
N=942				
M. Eval.= 190740	EvAg	**98.72±0.60**	-	-
Trap4	GGA	28.02±0.14	W=2500 p-value < 2.22e-16	yes
L=36	SSGA	31.88±1.15	W=2476 p-value<2.22e-16	yes
N=600				
M. Eval.= 393000	EvAg	**34.82±0.66**	-	-

Table 5.6 — Wilcoxon test comparing the best fitness distributions of equally parametrised SSGA, GGA and EvAg in 2,3 and 4-trap. Results are obtained over 50 independent runs.

5.4.4 Summary

The EvAg model has been proved to have good scalability on trap functions with search landscapes of different degrees of difficulty. We found that EvAg scales better than a GGA and an SSGA that only differ from it in the population structure. Based on various scenarios with and without mutation we can conclude that EvAg needs fewer evaluations to reach a solution in addition to requiring smaller populations. The improvement is much more noticeable as the problem difficulty increases showing thereby the adequacy of the P2P approach for tackling large instances of difficult problems. These results are specially remarkable in the deceptive case for a BB size of 4. It illustrates the good scalability of the parallel approach when tackling difficult problems.

To gain more detailed insights, the runtime behaviour of equally parametrised EvAg, SSGA, and GGA approaches have been analysed on large problem instances where the differences of scalability are more outstanding. The results show that EvAg can maintain higher population diversity and better progress in fitness. As a consequence, an oversized population for the EvAg model still remains undersized for the canonical approaches that get lost in local optima.

5.5 Test-Case 2: Influence of the population structure on the algorithm performance

As described in Section 4.1, the EvAg model imposes no restrictions on the choice of a population structure. Within that context, the newscast protocol has been considered throughout this thesis in order to allow a decentralised execution of the approach in a P2P system. Nevertheless, different structures will influence the dynamics of the algorithm in a different way and therefore, the algorithmic performance. Hence, the aim of this test-case is the comparison of performances of the approach using three different population structures. To that aim, the ring and the Watts-Strogatz method explained in Section 2.2 have been considered for comparison against newscast (Figure 5.5 shows snapshots for the different population structures).

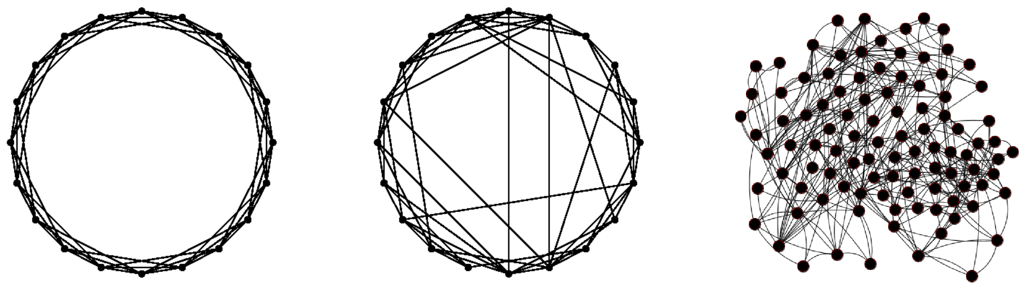


Fig. 5.5 — From left to right: ring, Watts-Strogatz and newscast population structures.

We have chosen a ring population structure as an instance to compare the performance of regular lattices against newscast since most of the works in the literature considering fine-grained approaches use to focus in regular lattices (some examples have been already mentioned in this thesis including [36, 3, 71, 17, 31, 30]). The main reason for such a choice is that the bisection method is not applicable for some other regular lattices such as a toroid or a grid in which the rectangular dimensions of the topology do not allow doubling the size of the population in the process of estimating the correct size.

In addition, the Watts-Strogatz method represents an easy and understandable model for creating a small-world population structure. This way, it will be possible to compare two different methods (i.e. Watts-Strogatz and newscast) for generating the same sub-type of complex network. The interest here goes a step further than in the case of the ring since there are many P2P protocols designed to work as small-world networks. Therefore, we aim to establish whether the properties on scalability of the newscast population structure lie in its small-world structure so that may extend to other protocols implementing the same kind of topologies (e.g. Gnutella 0.4 [29] or any DHT [68, 60, 77, 58]).

Figure 5.6 depicts the influence of the three methods in the environmental selection pressure by their respective takeover times. It shows both small-world populations inducing similar pressures in spite of being a bit stronger in the case of newscast. Meanwhile, the ring structure shows a delay in the takeover time given that the network diameter is larger at such a kind of topology.

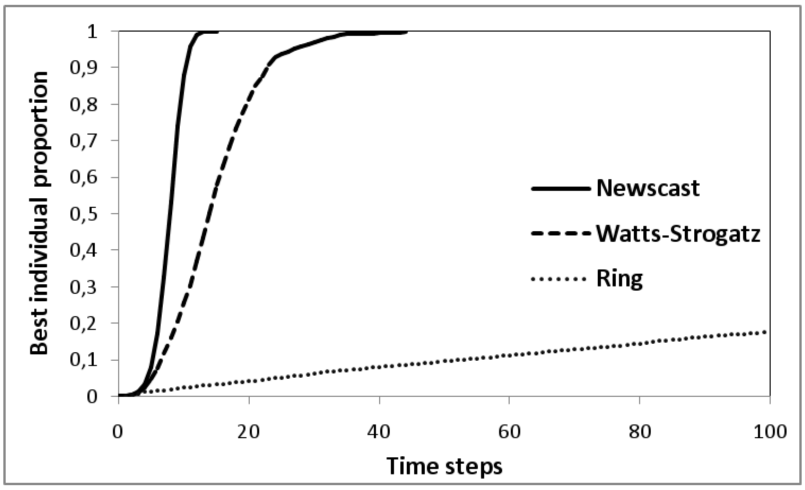


Fig. 5.6 — Takeover time curves in a ring, Watts-Strogatz with $p = 0.2$ and newscast population structures. All results averaged from 50 independent runs, for a population size of $n = 1600$ and binary tournament.

5.5.1 Scalability analysis

The following experiment aims to compare the influence of previous decentralised population structures on the scalability of the EvAg model when tackling trap functions. Table 5.7 summarises the settings in these series including the ring, Watts-Strogatz and newscast.

Figure 5.7 depicts the scalability of the population size and the computational effort for the different population structures in 2, 3 and 4-trap functions. Results show that the ring structure is able to scale better than its counterparts with respect to the population size. In that context, the ring has been shown to relax the environmental selection pressure and will consequently preserve the genetic diversity. This way, the population size can be reduced at every instance and the scalability order improves. Nevertheless, the analysis drastically changes with

Trap instances	
BB size	$2, 3, 4$
Individual Length (L)	$12, 24, 36, 48, 60$
GA settings	
GA	selectorecombinative EvAg
Population size	Tuning algorithm
Selection of Parents	Binary Tournament
Recombination	Uniform crossover, $p_c = 1.0$
Population settings	
Population structure	Ring
	Watts-Strogatz, $p = 0.2$
	Newscast
Node degree	20

Table 5.7 — Test-case 2: parameters of the experiments in the analysis of scalability.

respect to the computational efforts. In such case, the ring population scales worse than the small-world ones requiring therefore a larger time to converge to optimal solutions.

In addition, the comparison between the two small-world methods shows that the scalability of the population size using the Watts-Strogatz method is slightly better than in the case of newscast. Nevertheless, results are quite similar with respect to the scalability of the computational efforts in which there is no clear trend of an approach outperforming the other when changing the problem landscape difficulty. Taking into account that results are estimations based on the bisection method, such orders of scalability point out that either Watts-Strogatz or newscast perform roughly the same, the newscast approach needing a slightly larger population size than Watts-Strogatz but requiring equivalent times to solution.

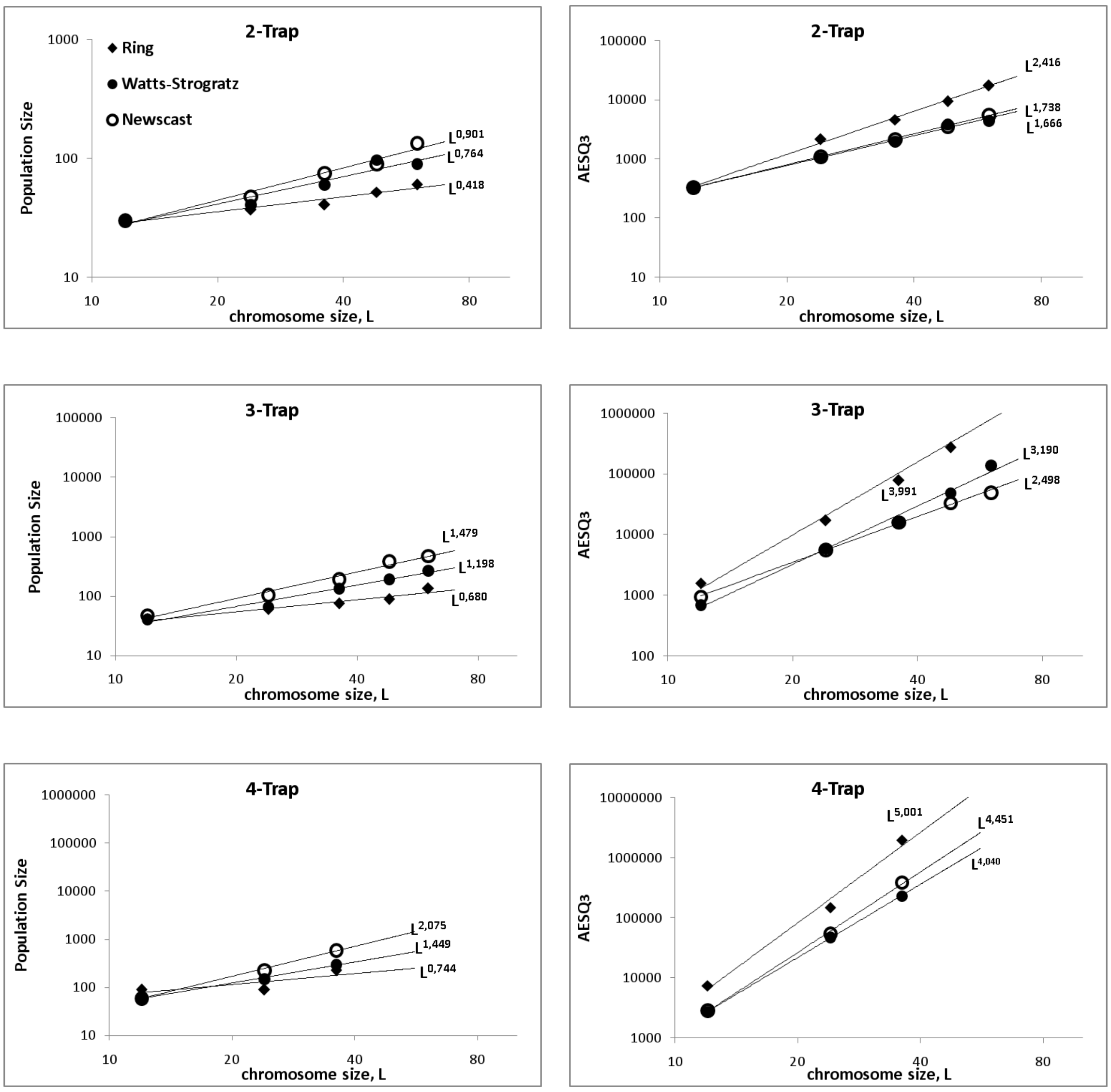


Fig. 5.7 — Scalability of the EvAg model using a Ring, Watts-Strogatz and Newscast population structures in trap functions for the estimated population sizes N *(left)* and the evaluations to solution in third quartile *(right)*. Results are obtained by bisection and depicted in a *log-log* scale as a function of the length of the chromosome, L.

5.5.2 Analysis of the algorithmic convergence

This section analyses the convergence of the EvAg approach in large instances of 2, 3 and 4-trap for different population structures. All settings for the experiments are summarised in Table 5.8. In order to establish the population sizes and the maximum number of evaluations for the different instances, we have chosen previous values obtained by bisection for the selectorecombinative EvAg using newscast. The aim here is to provide better insights on the algorithmic convergence for the three topologies when the EvAg model is equally parametrised.

Trap instances	
2-Trap	
Individual Length (L)	60
Population size	135
Termination Condition	Max. Eval. = 5535
3-Trap	
Individual Length (L)	60
Population size	480
Termination Condition	Max. Eval. =49920
4-Trap	
Individual Length (L)	36
Population size	600
Termination Condition	Max. Eval. =393000
GA settings	
GA	EvAg
Selection of Parents	Binary Tournament
Recombination	Uniform crossover, $p_c = 1.0$
Mutation	Bit-flip mutation, $p_m = \frac{1}{L}$
Population settings	
Population structure	Ring
	Watts-Strogatz
	Newscast
Node degree	20

Table 5.8 — Test-Case 2: Parameters of the experiments for the analysis of convergence.

Figure 5.8 shows both small-world approaches having a better progress in fitness than the ring one in every instance under study. In fact, either Watts-Strogatz or newscast reach the same qualities in solutions at the maximum number of evaluations.

In addition, results for the genotypic entropy provide some keys on the influence of the population structure on the algorithm. The ring preserves the genetic diversity at a higher level than its counterparts delaying this way the convergence of fitness. Besides, the same shape in curves of the Watts-Strogatz and newscast approaches indicate that both population structures belong to a same kind of topology. Nevertheless, newscast promotes a stronger selection pressure as was shown by the takeover time curves in Figure 5.6.

5.5.3 Statistical analysis

The Wilcoxon analysis in Table 5.9 shows that differences in fitness between newscast and ring population structures are statistically significant which confirms previous results on the different convergences of the approaches. Nevertheless, such differences do not appear when comparing newscast with the Watts-Strogatz model in 2 and 4-trap. Additionally, the $p - value = 0.045$ in the 3-trap instance points to a close related distribution of fitness between both approaches.

Therefore, the small-world population structures generated by both methods promote equivalent algorithmic performances. This fact is promising for the exploration of other small-world based P2P protocols

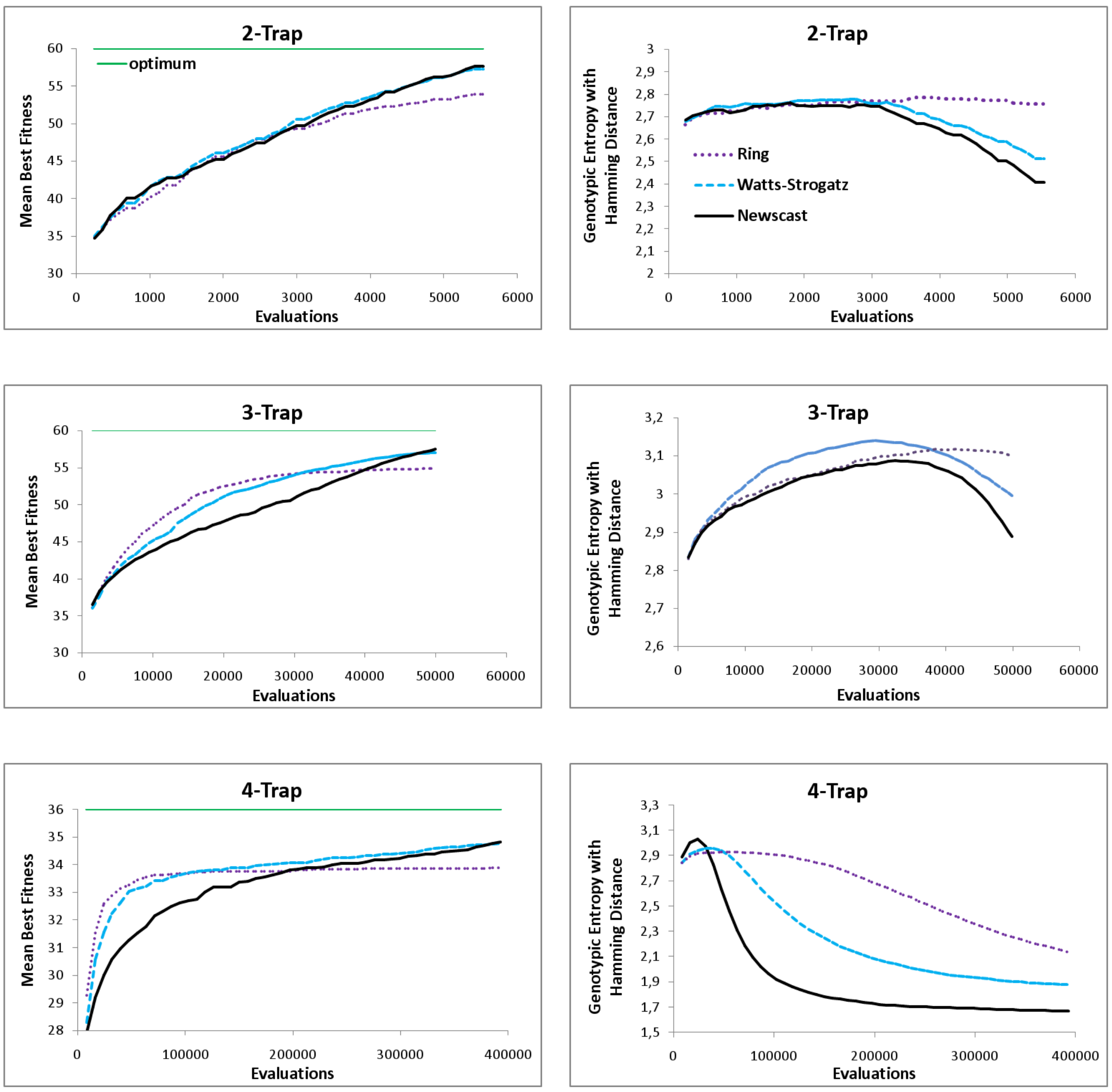


Fig. 5.8 — Test-Case 2: Best fitness convergence (*left*) and evolution of the diversity expressed as the entropy based on the Hamming distances between the genotypes (*right*). Graphs plotted represent the average of 50 independent runs.

in the context of the EC paradigm.

Problem Instance	*Algorithm*	*Avg. Fitness* $\pm\sigma$	*Wilcoxon Test*	*Significantly different?*
Trap2				
L=60	Ring	53.88±1.21	W=2429 p-value=2.22e-16	yes
N=135	**Watts-Strogatz**	**57.26±1.52**	**W=1386 p-value=0.335**	**no**
M. Eval.= 5535	**Newscast**	**57.6±1.38**	-	-
Trap3				
L=60	Ring	54.9±0.97	W=2135 p-value=6.46e-10	yes
N=480	**Watts-Strogatz**	**57.04±1.07**	**W=1536 p-value=0.045**	**yes**
M. Eval.= 49920	**Newscast**	**57.5±2.04**	-	-
Trap4				
L=36	Ring	33.88±0.69	W=2044 p-value=4.38e-09	yes
N=600	**Watts-Strogatz**	**34.78±0.65**	**W=1288 p-value=0.77**	**no**
M. Eval.= 393000	**Newscast**	**34.82±0.66**	-	-

Table 5.9 — Wilcoxon test comparing the best fitness distribution of the EvAg model using a Ring, Watts-Strogatz and Newscast population structures. Results are obtained over 50 independent runs.

5.5.4 Summary

In this test-case, we have analysed the EvAg model using different decentralised population structures in order to assess their influence on the performance of the algorithm. A ring topology and the Watts-Strogatz method were considered for comparison against the newscast method which allows a decentralised execution of the approach in a P2P system.

The ring topology has been chosen as the instance to compare regular lattices performances against newscast given that such kind of population structures are commonly used in fine-grained approaches. In addition, the Watts-Strogatz method is an easy method for generating small-world topologies. The aim here is to establish whether population structures based on small-world networks have equivalent performances so that EvAg can be extended to other P2P protocols implementing such a kind of topologies.

Results show that the ring approach needs smaller population sizes than newscast to guarantee a reliable convergence but, in turn, it requires of a larger number of evaluations which translates into larger times to solution. On the other hand, the Watts-Strogatz method exerts a slightly more relaxed selection pressure in the algorithm than newscast, however, results on the computational scalability and on the algorithmic convergence show that both approaches have similar performances that do not present statistical differences in qualities of solutions. Therefore, it can be concluded that both methods promote similar behaviours in the algorithm performance. We find that fact promising since such a property may extend to other small-world based P2P protocols.

5.6 Test-Case 3: Fault tolerance of the model

As explained in chapter 3, P2P systems are large networks of volatile resources in which the collective dynamics of peers joining and departing from the system is known as *churn*. This way, addressing churn in a P2P EA turns into a requirement of design since failures in peers are inherent to the system.

Following the work by Stutzbach and Rejaie in [69], there are two main group-level properties of *churn* characterising the behaviour of every participating peer: The *inter-arrival time* and the *session length*, respectively, the time between two sessions and the time from the beginning to the end of a session. In this test-case, we have assumed that all peers start at the same time with a certain *session length* and avoiding *inter-arrivals*. Therefore, once a peer leaves the system, it does not re-join again so that the system *degrades* from the initial configuration.

The *session length* can be modelled randomly from a Weibull distribution using the following formula:

$$X = \lambda(-ln(U))^{\frac{1}{k}} \tag{5.3}$$

where U is drawn from the uniform distribution, k stands for the shape of the *degradation* and λ for the time scale.

In this context, Stutzbach and Rejaie analyse the runtime dynamics of three real P2P systems and conclude that all *session lengths* fit with a Weibull distribution of shape $k \approx 0.40$ but differing on the time dimension λ. Given that experiments are conducted in a simulator and a simulator cycle represents different time units in real time, λ parameter was pre-adjusted to simulate different failure rates in such a way that the system degrades up to 90% in the worst case. In concrete, we use the following values for $\lambda = 400, 2500$ (depicted in Figure 5.9). It shows the Weibull cumulative distribution functions for such values, representing the percentage of remaining nodes at each moment of a experiment (e.g. in the cycle 2000, ~15% of the peers remain for $\lambda = 400$ and ~75% for $\lambda = 2500$).

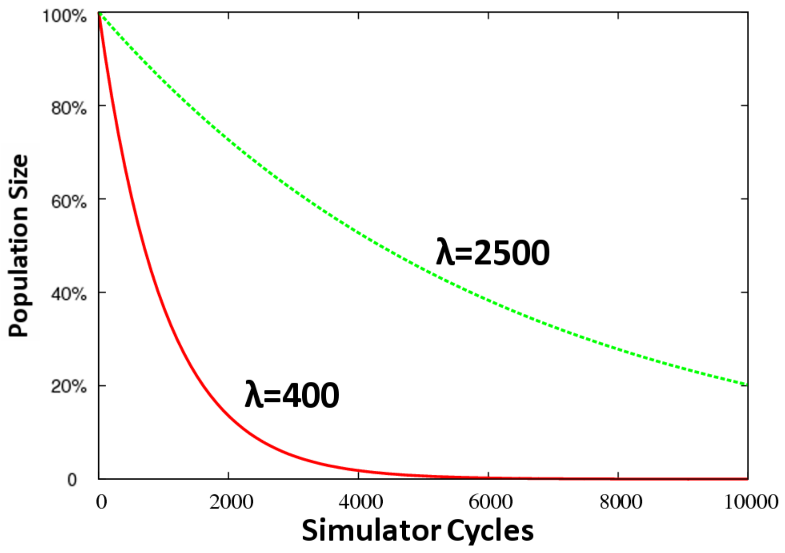


Fig. 5.9 — Complementary cumulative distribution functions for a Weibull distribution in function of the simulator cycles. Percentages are obtained as the ratio between the total number of failures and the initial population size.

5.6.1 Scalability analysis

Using previous *degradation* rates, several instances of 2, 3 and 4-trap have been analysed in order to assess the impact of churn in the EvAg model. All settings for the experiments are summarised in Table 5.10.

This way, there will be three variables affecting the performance of the algorithm: The size of the problem (L), which will conduct to a scalability analysis, the intensity of churn (λ) and the initial[2] population size (N). Being λ and L two independent variables under the condition of obtaining a SR of 0.98, the initial population size can be expressed as a function $f(\lambda, L) = N$ and empirically estimated using the bisection method.

Trap instances	
BB size	$2, 3, 4$
Individual Length (L)	$12, 24, 36, 48, 60$
GA settings	
GA	selectorecombinative EvAg
Population size	Tuning algorithm
Selection of Parents	Binary Tournament
Recombination	Uniform crossover, $p_c = 1.0$
Newscast settings	
Cache Size	20
Scenarios of churn	
λ	400,2500
k	0.4

Table 5.10 — Test-case 3: parameters of the experiments in the analysis of scalability.

Figure 5.10 shows the scalability of the population size (N) as a

[2]Given that the system degrades, the initial population size will not correspond to the final one.

function of L, that is, L scales and λ remains fixed in $f(\lambda, L) = N$. *Churn* does not seem to damage the scalability of the approach since estimated curves $f(\lambda, L)$ roughly follow the same orders of scalability and just appear shifted by a constant which is churn dependent. The more intensive the churn, the bigger the constant. In this sense, a small increase on the initial population size is enough to provide resilience to system failures since orders of scalability go from $O(L^{0.901})$ to $O(L^{0.928})$ in 2-trap, $O(L^{1.479})$ to $O(L^{1.799]})$ in 3-trap and are estimated to $O(L^{2.075})$ for any scenario in 4-trap.

In addition, graphics on the $AESQ_3$ show that the computational efforts required for tackling any given instance are independent from the churn scenario. Given that churn does not affect the scalability of the computational efforts, results exclusively depend on the problem instance size L and, therefore, EvAg degrades gracefully. We will require the same computational efforts under any churn scenario if we ensure enough resources to satisfy the condition of a SR of 0.98.

Figure 5.11 provides a better idea of the extent of these results. It represents the percentage of individuals of N for which each experiment is expected to end. The effects of churn are more pernicious as the instances scale. In the worst case (i.e. $L = 36$ in 4-trap and $\lambda = 400$), the initial population ends with a $\sim 10\%$ of the individuals, still guaranteeing a reliable convergence.

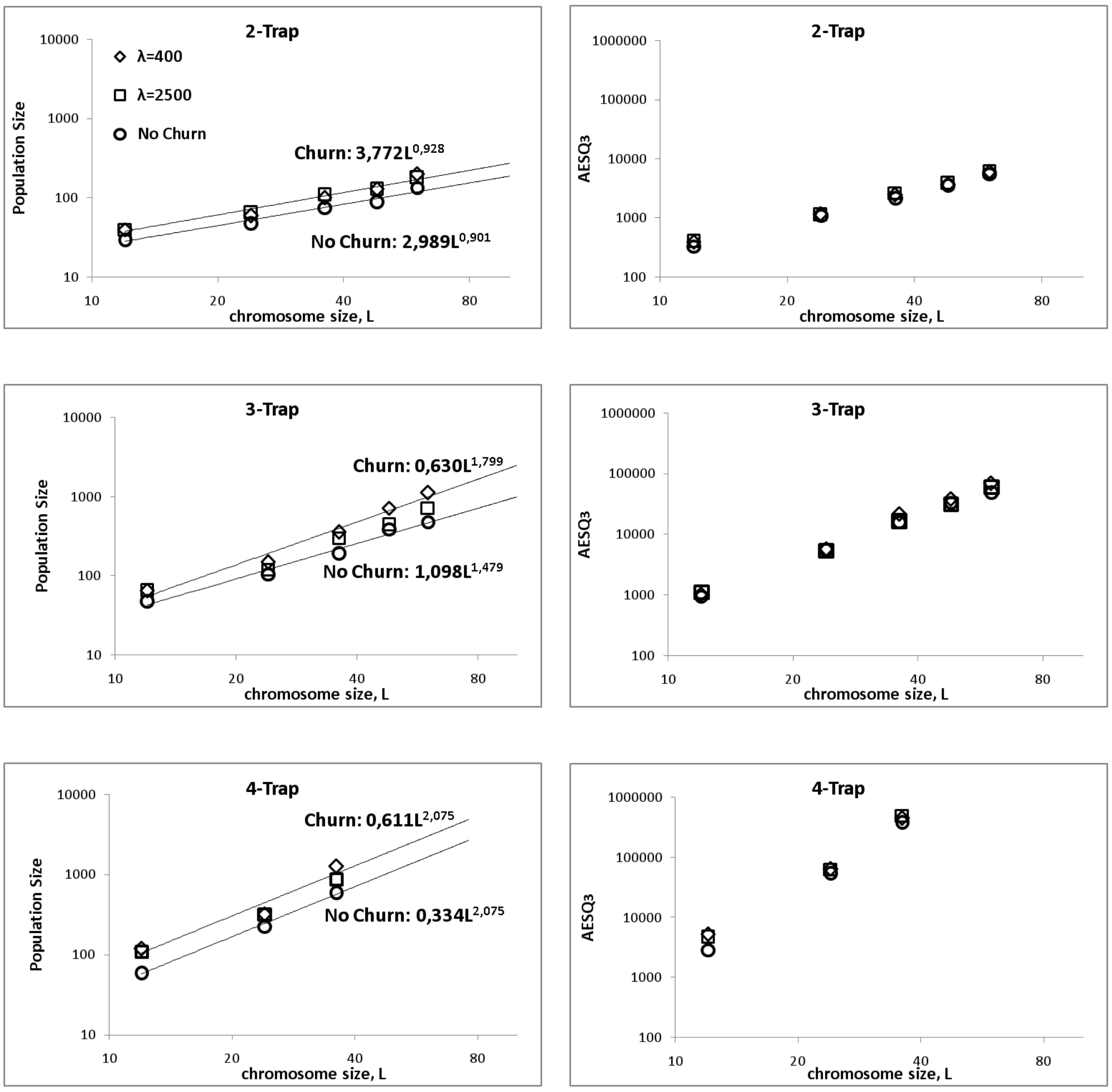


Fig. 5.10 — Scalability of the EvAg model for two different scenarios of churn (λ) and a failure-free environment in trap functions for the estimated population sizes N *(left)* and the evaluations to solution in third quartile *(right)*. Results are obtained by bisection and depicted in a *log-log* scale as a function of the length of the chromosome, L.

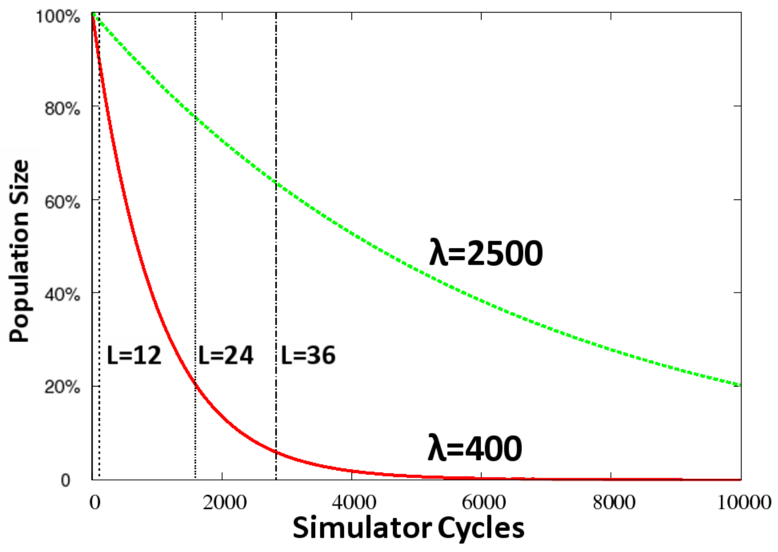


Fig. 5.11 — Degradation of the system for the different failure rates and execution times in three instances of the 4-trap function.

5.6.2 Analysis of the algorithmic convergence

This section analyses the convergence of the EvAg approach in 2, 3 and 4-trap instances for different degradation rates $\lambda = 400, 2500$ with respect to a failure-free execution. All settings are summarised in Table 5.11.

Initial population sizes have been set to those values obtained in the previous series for the failure-free run of the selectorecombinative EvAg. Using such values means that populations will be oversized for the failure-free counterpart in these series that use mutation. Nevertheless, that might not be the case when the system degrades. As previously seen, a small increase on the initial population size is sufficient condition for tolerating faults. Given that populations are oversized for a failure-free run using mutation, values will approximate an optimal size whenever the system degrades.

Trap instances	
2-Trap	
Individual Length (L)	60
Population size	135
Termination Condition	Max. Eval. = 5535
3-Trap	
Individual Length (L)	60
Population size	480
Termination Condition	Max. Eval. =49920
4-Trap	
Individual Length (L)	36
Population size	600
Termination Condition	Max. Eval. =393000
GA settings	
GA	EvAg
Selection of Parents	Binary Tournament
Recombination	Uniform crossover, $p_c = 1.0$
Mutation	Bit-flip mutation, $p_m = \frac{1}{L}$
Newscast settings	
Cache size	20
Scenarios of churn	
λ	400,2500
k	0.4

Table 5.11 — Test-Case 3: Parameters of the experiments for the analysis of convergence.

Figure 5.12 shows indeed that the EvAg model converges better when the system degrades, specially in the worst case for $\lambda = 400$. This fact is remarkable since the degradation of the system does not inflict a penalisation in the quality of solutions as could be expected but it is able to outperform the failure-free run despite EvAgs departing possibly contain valid solutions.

Besides, genetic diversity decreases faster as the system degradation turns more intensive which can be translated into a more exploitative behaviour of the algorithm. Such conclusion points out a possible explanation for the fault-tolerance of the model; as it has already been shown in Sections 5.4 and 5.5, EvAg is good at the preservation of genetic diversity and this way, it is able to balance the effect of an increasing exploitation component when the system degrades.

Finally and providing a quantitative view on the run-time dynamics of *churn*, Figure 5.13 depicts the degradation of the population for the different runs in the 4-trap instance. It shows how the system degrades up to $\sim 90\%$ in the worst case for $\lambda = 400$.

5.6.3 Statistical analysis

In order to provide previous results with an adequate statistical processing, Table 5.12 presents the Wilcoxon analysis of the data showing significant differences between the fitness distributions for the EvAg model with and without failures in the system.

The statistical analysis confirms the conclusion on the system degradation outperforming the failure-free run. In other words, the EvAg model is inherently fault-tolerant and degrades gracefully.

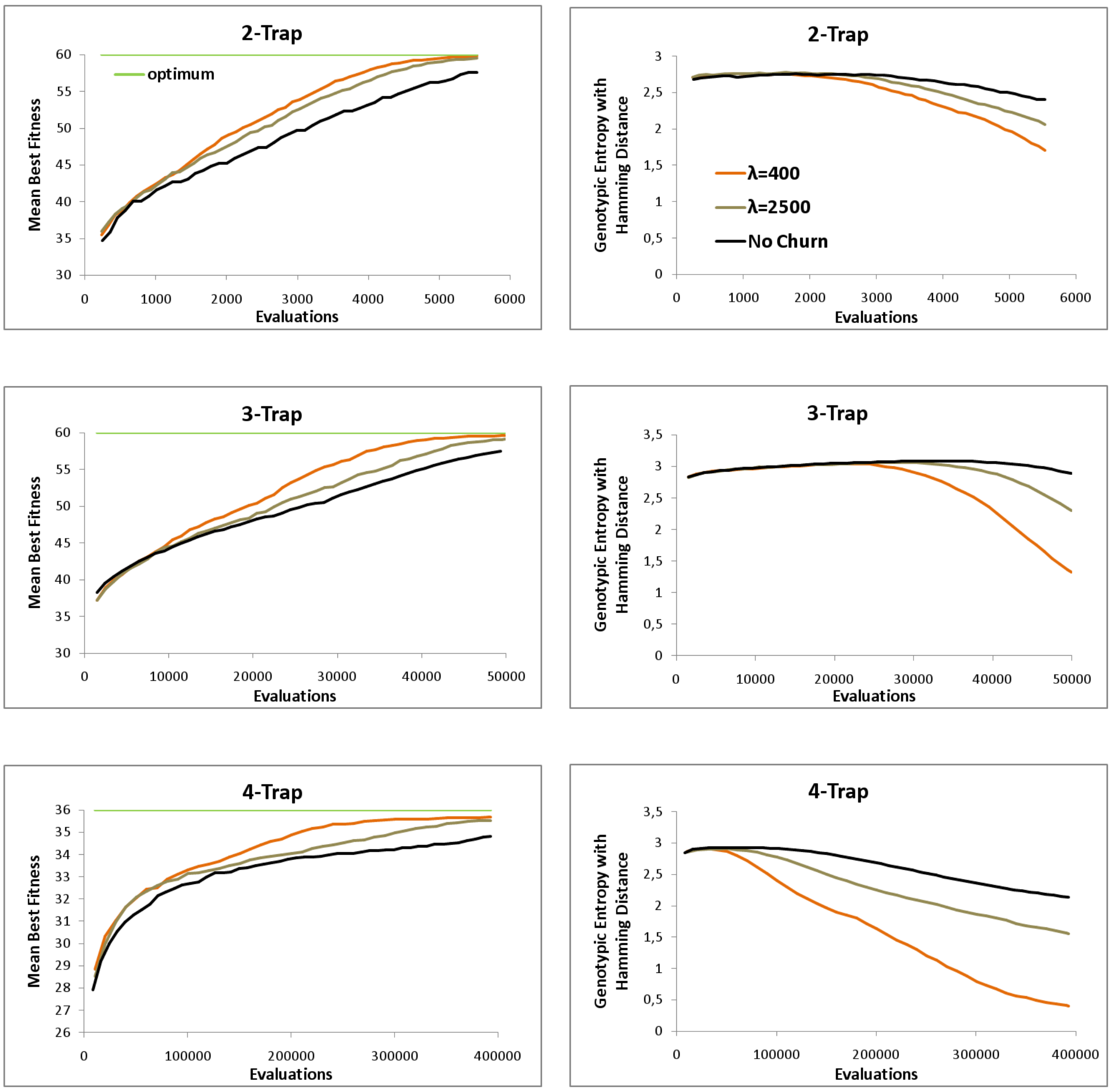


Fig. 5.12 — Test-case 3: Best fitness convergence (*left*) and evolution of the diversity expressed as the entropy based on the Hamming distances between the genotypes (*right*). Graphs plotted represent the average of 50 independent runs.

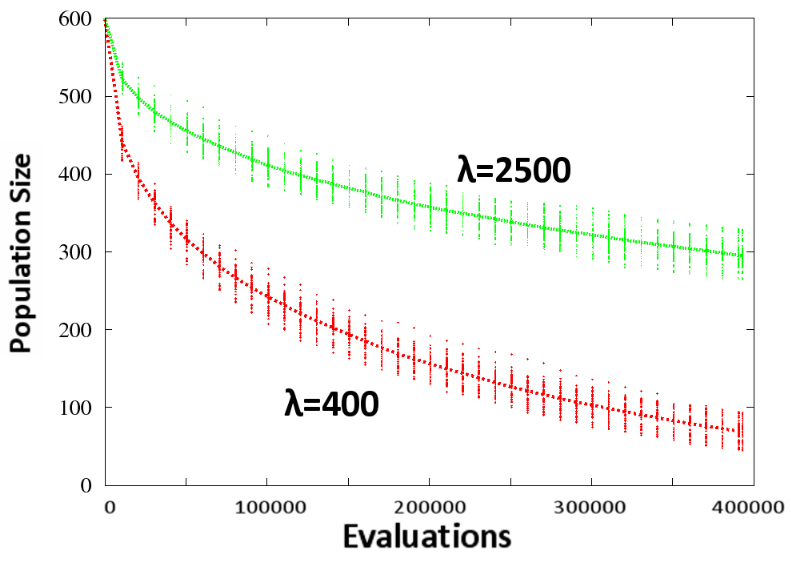


Fig. 5.13 — Population dynamics in the 4-trap instance for $L = 36$. Population sizes of the different runs are represented as points and the respective averaged values as lines for $\lambda = 400$ and $\lambda = 2500$.

Problem Instance	*Churn*	*Avg. Fitness* $\pm\sigma$	*Wilcoxon Test*	*Significantly different?*
Trap2				
L=60	λ = 400	**59.82±0.43**	**W=184 p-value=6.32e-15**	**yes**
N=135	λ = 2500	59.5±0.78	W=320 p-value=2.9e-11	yes
M. Eval.= 5535	*No Churn*	*57.6±1.38*	-	-
Trap3				
L=60	λ = 400	**59.62±0.62**	**W=414 p-value=1.37e-9**	**yes**
N=480	λ = 2500	59.14±1.2	W=614 p-value=6.2e-6	yes
M. Eval.= 49920	*No Churn*	*57.5±2.04*	-	-
Trap4				
L=36	λ = 400	**35.68±0.46**	**W=447 p-value=1.89e-9**	**yes**
N=600	λ = 2500	35.54±0.69	W=593 p-value=1.2e-6	yes
M. Eval.= 393000	*No Churn*	*34.82±0.66*	-	-

Table 5.12 — Wilcoxon test comparing the best fitness distribution of the EvAg model under different *degradation* rates. Results are obtained over 50 independent runs.

5.6.4 Summary

In this test-case, we have analysed the fault tolerance of the EvAg model when running on a computing platform that degrades following the modelling of the churn dynamics established by Stutzbach and Rejaie in [69]. To that aim, experiments were conducted on several instances of 2, 3 and 4-trap functions for different degradation rates.

Through the experimental results we can conclude that *churn* does not damage the scalability order of the algorithm and a small increase on the initial population size is enough to keep a reliable convergence towards optimal solutions. In that sense, large instances of trap functions have been tackled with success in spite of aggressive *churn* conditions in which peers depart from the system until 90% of the initial configuration is left.

In addition, the approach shows to be resilient to *churn* with respect to execution time; once that the estimated population size guarantees a reliable convergence to the problem optimum, the departure of nodes does not inflict a penalisation on the computational effort.

Therefore, the EvAg model is fault-tolerant and implements a *graceful degradation* without any other extra mechanism than the emergent behaviour of the approach itself.

5.7 Conclusions

In this chapter, we have carried out an experimental analysis on the performance of the EvAg model. To that aim, we investigate the scalability and fault-tolerance of the approach in a simulated P2P environment. Such goals have been tackled in three different test-cases in which experiments are conducted on trap functions in order to assess the scalability of population sizes and computational efforts with increasing problem size and difficulty.

In the first test-case, the scalability of the P2P model has been shown to outperform canonical panmictic approaches either in the population size or the computational efforts required to tackle the different problem instances. Furthermore, the improvement is specially outstanding as the problem difficulty increases pointing out that the EvAg model is suitable for tackling large instances of difficult problems.

In the second test-case, we have analysed the influence of different population structures on the scalability of the approach. In that context, the choice of newscast as population structure has a positive effect on the algorithmic performance showing better times to solution than the ring lattice in every problem instance and equivalent performances to the Watts-Strogatz approach.

Finally, the third test-case focuses on the fault-tolerance of the model when running on failure prone platforms that degrade following two different churn rates. Results show that a small increase on the initial population size is a sufficient mechanism to guarantee the convergence of the approach, in turn, such an increase does not inflict a penalisation on the computational effort.

This way and going back to the goals pose at the beginning, it can be concluded that the EvAg model is scalable and fault-tolerant.

Chapter 6

Conclusions

This thesis studies the viability of the Peer-to-Peer Evolutionary Computation paradigm in the context of fine-grained spatially-structured Evolutionary Algorithms. To that end, the Evolvable Agent model has been presented and assessed empirically under different scenarios using additively-decomposable trap-functions as a benchmark. Given that trap-functions have been designed to be difficult for Evolutionary Algorithms, the results should be easily extended to more general discrete or combinatorial optimization problems.

In particular, this thesis provides the following contributions to the understanding of distributed Evolutionary Computation in Peer-to-Peer infrastructures:

The viability of the P2P EC paradigm has been analysed around the issues of decentralisation, scalability and

fault-tolerance. A main challenge on the design of P2P EAs is to provide a single coherent view of the system despite the nature of peers being decentralised. This way, P2P systems create powerful parallel infrastructures able to constitute a single virtual computer composed of a potentially large number of interconnected resources. In order to take full advantage of such systems, the motivation of a P2P EA is tackling large instances of difficult problems at the increasing computing requirements that such instances claim. In that context of massive-scalability, failures of peers become inherent to the system and the approach has to demonstrate fault-tolerance in order to hold good performances.

The EvAg model is proposed as a simple and decentralised approach for P2P EC. The Evolvable Agent model is a fine-grained spatially-structured EA that defines a decentralised population structure by means of the gossiping protocol newscast. This makes the approach suitable for a P2P execution in which every EvAg can be potentially placed in a different peer. In that context, EvAgs evolve asynchronously in a loosely-coupled fashion with the mate selection locally restricted within their respective neighbourhoods.

Linear speed-ups can be hold for large instances of demanding problems. In problems with very expensive fitness evaluation cost, the parallel performance of the EvAg model has been shown to depend on the underlying computing platform. This way, for very demanding problem instances and high performance computer architectures, the approach has been estimated to hold linear speed-ups up to thousands of processors. In addition, the multi-threading nature of the approach is able to take a seamless advantage of the different cores in desktop machines implementing a SMP architecture.

The experimental analysis is based on the correct sizing of the populations. Setting an adequate population size is a key to obtain good performances in EAs, that is, to preserve a good quality in the solutions without spending extra computational efforts. In order to estimate optimal population sizes, we have used an empirical method based on bisection. This way, investigating scalability is possible by observing how population sizes and computational efforts scale for different problem instances.

The source code of the simulator is released as open-source. We find that practitioners can benefit from it either reproducing experiments or extending the framework. The simulator can be found at the subversion repository `https://forja.rediris.es/svn/geneura/evogen`, published under GPL v3 license.

P2P EAs scale well for increasing problem sizes and difficulties. EvAg has been shown to scale better than canonical approaches requiring of a smaller population size and number of evaluations as problem instances become large. The improvement is much more visible as the problem difficulty increases showing the adequacy of the P2P approach for tackling large instances of difficult problems. In addition, the comparison against a ring structured population shows that such a regular lattice population requires of a larger number of evaluations having therefore, a worst algorithmic performance.

Small-world population structures offer good trade-offs between exploitative and explorative components of EAs. The inhomogeneities of small-world structured populations play an important role in the preservation of the genetic diversity and have a positive effect on scalability. On the one hand, the influence of the environmental selection pressure is not so high as in panmictic populations so that population sizes and times to solutions can be reduced. In addition, the progress of the genetic diversity points out a more exploitative behaviour than the one induced by regular lattices. This way, the number of evaluations required to find optimal solutions can be minimised.

In that context, we have compared the performance of the Watts-Strogatz and newscast methods generating small-world population structures. Both methods have been shown to yield similar results outperforming, in turn, the algorithmic scalability of the rest of approaches in this thesis.

P2P EAs suffer a graceful degradation under churn. The EvAg approach has been shown to be robust under different degradation rates of *churn*; the departure of nodes does not inflict a penalisation in the execution time and the quality of solutions can be hold by simply increasing the initial population size.

We find that previous contributions represent a great advance to a better understanding of the P2P EC paradigm by pointing out the main issues and showing the viability of the approach. Nevertheless, there is much work to be done for a whole comprehension of P2P EAs and, more in general, parallel EAs. In that line, we plan to focus in future works on the following challenges that have been identified throughout the development of the thesis:

Validation of the model in a real P2P infrastructure: After studying the viability of the EvAg model, the validation of the approach is a logical further step. To that aim, we plan to deploy the algorithm in a real P2P platform for tackling demanding problem sets such as the very large instances of the vehicle routing problem [50]. This way, it will be possible to focus on engineering issues such as the real impact of the latency and bandwidth on the algorithm performance or the exploration of load-balancing methods to cope with the issue of heterogeneity in peers.

Exploration of other P2P protocols as population structures: This thesis has shown that different methods generating small-world population structures have equivalent performances in EAs. This fact is remarkable since there are many P2P protocols designed to work as small-world networks as Gnutella 0.4 [29] or any DHT [68, 60, 77, 58]. Therefore, we aim to explore some of these P2P protocols as population structures for EAs so that P2P EC can benefit from existing P2P platforms.

Extension of the P2P concept to other optimisation meta-heuristics: To this point, we have focused in P2P optimisation within the concrete field of EC, however, there are no restrictions imposing a limitation of the P2P approach to other paradigms as they could be EDAs, ACO or PSO. In fact, PSO has already received some attention in the literature, e.g. the P2P-MOPSO approach by Scriben et al. in [65] or the one by Bánhelyi et al. in [9].

As a final consideration and despite some future lines of work have been exposed, we feel that P2P optimisation is on its beginnings yet and this thesis can be a source of ideas and motivation for practitioners and theoreticians to do further investigations in the area.

6.1 Published papers related to the thesis

During the development of this thesis, and directly related to it, the following papers were published on different peer reviewed journals and conference proceedings:

Peer reviewed journal papers :

1. Juan Luis Jimenez Laredo, Agoston E. Eiben, Maarten van Steen, and Juan Julian Merelo. Evag: A scalable peer-to-peer evolutionary algorithm. Genetic Programming and

Evolvable Machines, 2010. http://dx.doi.org/10.1007/s10710-009-9096-z.

2. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M. Mora, Juan Julian Merelo, and Carlos Fernandes. Resilience to churn of a peer-to-peer evolutionary algorithm. Int. J. High Performance Systems Architecture, 1(4):260-268, 2009.
3. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio Miguel Mora, and Juan Julian Merelo Guervos. Evolvable agents, a fine grained approach for distributed evolutionary computing: walking towards the peer-to-peer computing frontiers. Soft Computing - A Fusion of Foundations, Methodologies and Applications, 12(12):1145-1156, 2008.

Peer reviewed conference papers and book chapters :

1. Juan Luis Jimenez Laredo, Juan Julian Merelo Guervos, and Pedro Angel Castillo Valdivieso. Paral. and Distrib. Comp. Intel., volume 269 of SCI, chapter Evolvable Agents: A Framework for Peer-to-Peer Evolutionary Algorithms, pages 43-62. Springer-Verlag Berlin Heidelberg, 2010.
2. Juan Luis Jimenez Laredo, Carlos Fernandes, Juan Julian Merelo, and Christian Gagne. Improving genetic algorithms performance via deterministic population shrinkage. In GECCO' 09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pages 819-826, New York, NY, USA, ACM, 2009.

3. Juan Luis Jimenez Laredo, Carlos Fernandes, Antonio Mora, Pedro A. Castillo, Pablo Garcia-Sanchez, and Juan Julian Merelo. Studying the cache size in a gossip-based evolutionary algorithm. In G.A. Papadopoulos and C. Badica, editors, Proceedings of the 3rd International Symposium on Intelligent Distributed Computing, volume 237 of Studies in Computational Intelligence, pages 131-140. Springer-Verlag Berlin Heidelberg, 2009.

4. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M. Mora, Carlos Fernandes, and Juan Julian Merelo. Addressing Churn in a Peer-to-Peer Evolutionary Algorithm. In Juan Lanchares, Francisco Fernandez, and Jose L. Risco-Martin, editors, WPABA' 08 - First International Workshop on Parallel Architectures and Bioinspired Algorithms, Toronto, Canada, pages 5-12. Complutense University Of Madrid, 2008.

5. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio Miguel Mora, Juan Julian Merelo Guervos, Agostinho Claudio da Rosa, and Carlos Fernandes. Evolvable agents in static and dynamic optimization problems. In Rudolph et al., editors. Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, Proceedings, volume 5199 of Lecture Notes in Computer Science, pages 488-497. Springer, 2008

6. Juan Luis Jimenez Laredo, Agoston E. Eiben, Maarten van Steen, and Juan Julian Merelo Guervos. On the run-time

dynamics of a peer-to-peer evolutionary algorithm. In Rudolph et al., editors. Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, Proceedings, volume 5199 of Lecture Notes in Computer Science, pages 236-245. Springer, 2008.

7. Juan Luis Jimenez Laredo, Agoston E. Eiben, Maarten van Steen, Pedro A. Castillo, Antonio Miguel Mora, and Juan Julian Merelo Guervos. P2P evolutionary algorithms: A suitable approach for tackling large instances in hard optimization problems. In Emilio Luque et al., editors, Euro-Par, volume 5168 of Lecture Notes in Computer Science, pages 622-631. Springer, 2008.

8. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M. Mora, and Juan Julian Merelo. Exploring population structures for locally concurrent and massively parallel evolutionary algorithms. In IEEE Congress on Evolutionary Computation (CEC2008), WCCI2008 Proceedings, pages 2610-2617. IEEE Press, Hong Kong, June 2008.

9. Agoston E. Eiben, Marc Schoenauer, Juan Luis Jimenez Laredo, Pedro A. Castillo Valdivieso, Antonio Miguel Mora, and Juan Julian Merelo. Exploring selection mechanisms for an agent-based distributed evolutionary algorithm. In Dirk Thierens, editor, GECCO' 07, pages 2801-2808. ACM, 2007.

10. Juan Luis Jimenez Laredo, Pedro A. Castillo Valdivieso, Ben Paechter, Antonio Miguel Mora, Eva Alfaro-Cid, Anna

Esparcia-Alcazar, and Juan Julian Merelo Guervos. Empirical validation of a gossiping communication mechanism for parallel EAs. In Mario Giacobini et al., editors, EvoWorkshops, volume 4448 of Lecture Notes in Computer Science, pages 129-136. Springer, 2007.

11. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M. Mora, and Juan Julian Merelo. Escalado con un sistema de agentes evolutivos distribuido. In Francisco Almeida-Rodriguez et al., editors, Actas MAEB 2007, pages 111-118, 2007.

12. Juan Luis Jimenez Laredo, Pedro A. Castillo, Gustavo Romero, Antonio M. Mora, Juan Julian Merelo, and Maribel G. Arenas. Validacion de un sistema computacional P2P mediante un estudio empirico. In XVII Jornadas de Paralelismo - XVII JP, pages 235-240, September 2006.

13. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M. Mora, and Juan Julian Merelo. Estudio preliminar sobre autoadaptacion en agentes evolutivos sobre arquitecturas heterogeneas. In XVII Jornadas de Paralelismo - XVII JP, pages 389-394, September 2006.

Chapter 7

Conclusiones

Esta tesis estudia la viabilidad del paradigma de computo evolutivo P2P en el contexto de los algoritmos evolutivos paralelos de grano fino espacialmente estructurados. Con este objetivo, el modelo de Agente Evolutivo ha sido presentado, y analizado empíricamente en diversos escenarios usando funciones trampa como problemas de prueba. Dado que las funciones trampa han sido diseñadas para ser difíciles para los algoritmos evolutivos, los resultados deberían poder extenderse fácilmente a otros problemas de optimización combinatoria.

En concreto, esta tesis proporciona las siguientes contribuciones al área de la computación evolutiva distribuida en infraestructuras P2P:

La viabilidad del paradigma de computación evolutiva P2P ha sido analizado alrededor de los temas de descentralización, escalabilidad y tolerancia a fallos. El reto principal al que se enfrenta un algoritmo evolutivo P2P es el de proporcionar una visión coherente al sistema a pesar de su naturaleza descentralizada. Para aprovechar estos sistemas, la motivación subyacente será abordar instancias grandes de problemas difíciles dados los altos requisitos computacionales que tales instancias requieren. En este contexto de escalabilidad masiva, los fallos son inherentes al sistema y cualquier enfoque tendrá que demostrar ser tolerante a fallos de forma que se puedan sostener los buenos rendimientos.

El modelo de Agente Evolutivo es propuesto como un enfoque simple y descentralizado para la computación evolutiva P2P. Dicho modelo consiste en un algoritmo evolutivo de grano fino que define la estructura de la población por medio del protocolo P2P newscast. Este hecho permite su ejecución en sistemas P2P en la que cada Agente Evolutivo puede ser potencialmente albergado en un nodo distinto de modo que el sistema evoluciona asíncronamente de forma desligada y con la selección de los padres localmente restringida a las respectivas vecindades.

Las ganancias lineales son posibles para intancias grandes de problemas computacionalmente pesados. En problemas con funciones de evaluación muy costosas, el rendimiento paralelo del modelo muestra ser dependiente de la plataforma de cómputo subyacente. De esta forma, se ha estimado que el modelo es capaz de sostener ganancias lineales hasta cientos de procesadores en plataformas de alto rendimiento para instancias de problema costosas.

El análisis experimental esta basado en una dimensionalidad correcta del tamaño de la población. Determinar un tamaño adecuado de población es clave para obtener buenos rendimientos en los algoritmos evolutivos, es decir, preservar buenas calidades en las soluciones sin derrochar esfuerzos computacionales. Con el propósito de estimar los tamaños de población óptimos para los distintos problemas y modelos estudiados, se ha usado un método empírico basado en bisección. De esta forma, investigar la escalabilidad de los distintos modelos ha sido posible al observar como los tamaños de la población y los esfuerzos computacionales escalan para distintas instancias de problema.

El código fuente del simulador utilizado para la experimentación ha sido liberado como software libre. De esta forma, creemos que los profesionales del área pueden beneficiarse de él, tanto en la reproducción de los experimentos de esta tesis como extendiendo el propio simulador. El código se haya disponible en un repositorio subversion en `https://forja.rediris.es/svn/geneura/evogen` publicado con licencia GPL v3.

Los algoritmos evolutivos P2P escalan bien en problemas difíciles e instancias grandes. El modelo de Agente Evolutivo ha mostrado escalar mejor que enfoques panmícticos clásicos al requerir menores tamaños de población así como un menor número de evaluaciones conforme las instancias del problema crecen. La mejora es mucho más perceptible conforme el problema se vuelve más complejo, mostrando por tanto, la aptitud del modelo para abordar instancias grandes de problemas difíciles. Además, la comparación con respecto al algoritmo de población estructurada en anillo muestra que dicha estructura conlleva un mayor número de evaluaciones y que por lo tanto tiene un peor rendimiento algorítmico.

Las poblaciones con estructura de mundo-pequeño ofrecen una buena solución de compromiso entre los componente explorativos y explotativos de los algoritmos evolutivos. Las inhomogeneidades de las poblaciones estructuradas como mundo-pequeño juegan un importante rol en la preservación de la diversidad genética y tienen un efecto positivo en la escalabilidad del algoritmo. Por un lado, la influencia de tal tipo de estructuras en la presión selectiva no es tan alta como en poblaciones panmícticas, de foma que los tamaños de población y los tiempos de ejecución pueden ser reducidos. Por otro lado, el progreso de la diversidad genética apunta que el comportamiento es más explotativo que en el caso de las estructuras de población regulares. Por tanto, el número de evaluaciones requeridas para encontrar soluciones óptimas es minimizado.

En este contexto, hemos comparado el rendimiento de dos métodos que generan estructuras de población mundo-pequeño: el protocolo Newscast y el método de Watts y Strogatz. Ambos métodos han mostrado alcanzar resultados similares superando, en cambio, la escalabilidad algorítmica del resto de enfoques en esta tesis.

Los algoritmos evolutivos P2P sufren una degradación grácil bajo condiciones de churn. El modelo de Agente Evolutivo ha mostrado ser robusto bajo distinatas tasas de degradación del sistema modeladas como churn; el fallo de los nodos no infringe penalizaciones a los tiempos de ejecución del algoritmo mientras que la calidad de las soluciones es mantenida mediante el incremento del tamaño de población inicial.

Dichas contribuciones representan un gran avance al entendimiento del paradigma de computo evolutivo sobre sistemas P2P puesto que apuntan los principales problemas de diseño y demuestran la viabilidad del enfoque. No obstante, creemos que aún se deben de realizar avances importantes en el área de cara a una total comprensión de los algoritmos evolutivos P2P, y más en general, de los algoritmos evolutivos paralelos. En este sentido, como trabajo futuro planeamos centrarnos en las siguientes líneas de investigación que hemos identificado a lo largo del desarrollo de esta tesis:

Validación del modelo en una infraestructura P2P real. Después de estudiar la viabilidad el modelo de Agente Evolutivo, la validación del enfoque es el siguiente paso lógico. Con ese propósito, planeamos implementar el algoritmo en una plataforma P2P real en la que se aborden conjuntos de problemas con alto coste computacional, como sería el caso de las instancias grandes del problema de encaminamiento de vehículos [50]. De esta forma, será posible centrarse en otros problemas que dependen de la infraestructura de cómputo como son el impacto de la latencia y el ancho de banda en el rendimiento del algoritmo o la investigación de métodos de balanceo de carga que solucionen la problemática de la heterogeneidad de los nodos de cómputo.

Investigación de otros protocolos P2P como estructuras de población. Esta tesis muestra que diferentes métodos que generan estructuras de población mundo pequeño tienen rendimientos parecidos en los algoritmos evolutivos. Este hecho es importante puesto que muchos de los protocolos P2P existentes, como son Gnutella 0.4 [29] o cualquier DHT [68, 60, 77, 58], están diseñados para funcionar como mundo-pequeño. Por lo tanto, pretendemos explorar algunos de ellos como estructuras de población para algoritmos evolutivos de tal forma que el ámbito de la computación evolutiva se pueda beneficiar de plataformas P2P existentes.

Extender el concepto P2P a otras meta-heurísticas de optimización. Hasta este punto, nos hemos centrado en la optimización P2P dentro del campo concreto de la computación evolutiva, sin embargo, no existen restricciones que limiten la aplicación del enfoque P2P a otros paradigmas, como podrían ser los algoritmos de estimación de distribuciones, la optimización basada en colonias de hormigas o los enjambres de partículas. De hecho, este último ya ha recibido alguna atención en la literatura, por ejemplo, el algoritmo P2P-MOPSO de Scriben et al. en [65] o el debido a Bánhelyi et al. en [9].

Como consideración final y a pesar de que se han expuesto algunas líneas de investigación futura, creemos que el campo de optimización P2P está aún en sus comienzos y esperamos que esta tesis sea una fuente de ideas e inspiración para los profesionales del área y así sigan contribuyendo a su desarrollo y entendimiento.

7.1 Publicaciones relacionadas con la tesis

Durante el desarrollo de esta tesis, y directamente relacionada con ella, los siguientes artículos fueron publicados en diferentes revistas y conferencias:

Artículos en revistas :

1. Juan Luis Jimenez Laredo, Agoston E. Eiben, Maarten van Steen, and Juan Julian Merelo. Evag: A scalable peer-to-peer evolutionary algorithm. Genetic Programming and Evolvable Machines, 2010. http://dx.doi.org/10.1007/s10710-009-9096-z.
2. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M. Mora, Juan Julian Merelo, and Carlos Fernandes. Resilience to churn of a peer-to-peer evolutionary algorithm. Int. J. High Performance Systems Architecture, 1(4):260-268, 2009.
3. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio Miguel Mora, and Juan Julian Merelo Guervos. Evolvable agents, a

fine grained approach for distributed evolutionary computing: walking towards the peer-to-peer computing frontiers. Soft Computing - A Fusion of Foundations, Methodologies and Applications, 12(12):1145-1156, 2008.

Artículos en conferencias y capítulos de libro :

1. Juan Luis Jimenez Laredo, Juan Julian Merelo Guervos, and Pedro Angel Castillo Valdivieso. Paral. and Distrib. Comp. Intel., volume 269 of SCI, chapter Evolvable Agents: A Framework for Peer-to-Peer Evolutionary Algorithms, pages 43-62. Springer-Verlag Berlin Heidelberg, 2010.
2. Juan Luis Jimenez Laredo, Carlos Fernandes, Juan Julian Merelo, and Christian Gagne. Improving genetic algorithms performance via deterministic population shrinkage. In GECCO' 09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pages 819-826, New York, NY, USA, ACM, 2009.
3. Juan Luis Jimenez Laredo, Carlos Fernandes, Antonio Mora, Pedro A. Castillo, Pablo Garcia-Sanchez, and Juan Julian Merelo. Studying the cache size in a gossip-based evolutionary algorithm. In G.A. Papadopoulos and C. Badica, editors, Proceedings of the 3rd International Symposium on Intelligent Distributed Computing, volume 237 of Studies in Computational Intelligence, pages 131-140. Springer-Verlag Berlin Heidelberg, 2009.
4. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M.

Mora, Carlos Fernandes, and Juan Julian Merelo. Addressing Churn in a Peer-to-Peer Evolutionary Algorithm. In Juan Lanchares, Francisco Fernandez, and Jose L. Risco-Martin, editors, WPABA' 08 - First International Workshop on Parallel Architectures and Bioinspired Algorithms, Toronto, Canada, pages 5-12. Complutense University Of Madrid, 2008.

5. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio Miguel Mora, Juan Julian Merelo Guervos, Agostinho Claudio da Rosa, and Carlos Fernandes. Evolvable agents in static and dynamic optimization problems. In Rudolph et al., editors. Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, Proceedings, volume 5199 of Lecture Notes in Computer Science, pages 488-497. Springer, 2008

6. Juan Luis Jimenez Laredo, Agoston E. Eiben, Maarten van Steen, and Juan Julian Merelo Guervos. On the run-time dynamics of a peer-to-peer evolutionary algorithm. In Rudolph et al., editors. Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, Proceedings, volume 5199 of Lecture Notes in Computer Science, pages 236-245. Springer, 2008.

7. Juan Luis Jimenez Laredo, Agoston E. Eiben, Maarten van Steen, Pedro A. Castillo, Antonio Miguel Mora, and Juan Julian Merelo Guervos. P2P evolutionary algorithms: A suitable approach for tackling large instances in hard opti-

mization problems. In Emilio Luque et al., editors, Euro-Par, volume 5168 of Lecture Notes in Computer Science, pages 622-631. Springer, 2008.

8. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M. Mora, and Juan Julian Merelo. Exploring population structures for locally concurrent and massively parallel evolutionary algorithms. In IEEE Congress on Evolutionary Computation (CEC2008), WCCI2008 Proceedings, pages 2610-2617. IEEE Press, Hong Kong, June 2008.

9. Agoston E. Eiben, Marc Schoenauer, Juan Luis Jimenez Laredo, Pedro A. Castillo Valdivieso, Antonio Miguel Mora, and Juan Julian Merelo. Exploring selection mechanisms for an agent-based distributed evolutionary algorithm. In Dirk Thierens, editor, GECCO' 07, pages 2801-2808. ACM, 2007.

10. Juan Luis Jimenez Laredo, Pedro A. Castillo Valdivieso, Ben Paechter, Antonio Miguel Mora, Eva Alfaro-Cid, Anna Esparcia-Alcazar, and Juan Julian Merelo Guervos. Empirical validation of a gossiping communication mechanism for parallel EAs. In Mario Giacobini et al., editors, EvoWorkshops, volume 4448 of Lecture Notes in Computer Science, pages 129-136. Springer, 2007.

11. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M. Mora, and Juan Julian Merelo. Escalado con un sistema de agentes evolutivos distribuido. In Francisco Almeida-Rodriguez et al., editors, Actas MAEB 2007, pages 111-118,

2007.

12. Juan Luis Jimenez Laredo, Pedro A. Castillo, Gustavo Romero, Antonio M. Mora, Juan Julian Merelo, and Maribel G. Arenas. Validacion de un sistema computacional P2P mediante un estudio empirico. In XVII Jornadas de Paralelismo - XVII JP, pages 235-240, September 2006.

13. Juan Luis Jimenez Laredo, Pedro A. Castillo, Antonio M. Mora, and Juan Julian Merelo. Estudio preliminar sobre autoadaptacion en agentes evolutivos sobre arquitecturas heterogeneas. In XVII Jornadas de Paralelismo - XVII JP, pages 389-394, September 2006.

Appendix A

Tuning of the Cache size parameter

Parallelising an EA implies the use of new parameters to control issues such as migration rates or the topology management. These new components drastically increase the complexity for the fine-tuning of the algorithm. In this sense, Cantú-Paz in [12] and Hidalgo and Fernandez in [39] analyse up to seven parameters derived from the parallelisation using islands. Within the EvAg model instead, newscast simplifies all the decision making for the parallelisation by tuning a single parameter, the cache size. The cache acts as a routing table in which every peer holds a list of neighbours peers. Therefore, the cache size represents the maximum number of connections (edges) that a peer could have. Taking into account the recommendations of Jelasity et al. in [42], the cache size has to be much smaller than the number of peers in order to get small-world features such as a small network diameter or a high clustering coefficient.

Given the importance of the parameter, this appendix tries to calibrate an adequate cache size for the problems under study in this thesis. To this aim, we measure the influence of different cache and population sizes on the algorithm performance when tackling a 4-trap instance. As will be shown, results indicate that success rates of the algorithm remain constant independently of the different cache sizes. This fact points to the robustness of the parallel model and helps decision making. Any cache size guaranteeing a small-world topology will produce equivalent performances.

Table A.1 summarizes the settings for the experiments in which eleven different cache sizes and two different population sizes have been tested.

Problem instance	
Problem	4-trap
Chromosome length	36
EvAg settings	
Population size	400, 600 individuals
Selection	Binary Tournament
Recombination	Uniform Crossover, $p_c = 1.0$
Max. Eval.	500000
Newscast settings	
$Cache_{size}$	10,14,18,22,26,30,34,38,42,46,50

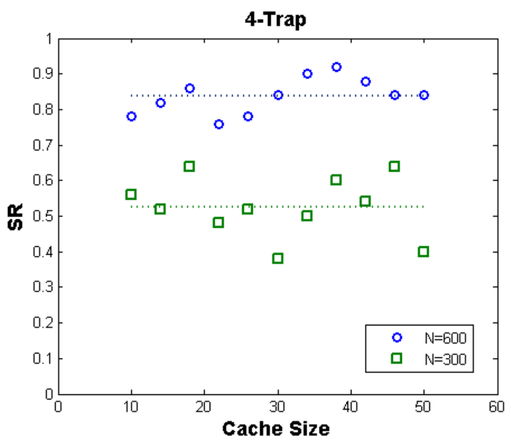
Table A.1 — Settings for the experiments.

A.1 Analysis of Results

Figure A.1 shows the SR and AES of the EvAg model when tackling the 4-trap instance using different cache and population sizes. It can be observed how the population size has an impact on the algorithm performance while the cache size does not.

The P2P EA is sensitive to the population size since the algorithm converges with a SR ~ 0.85 for $N = 600$ while it decreases to ~ 0.52 for $N = 400$. Nevertheless, results for different cache sizes move around the same SRs and AES. This way, neither the algorithm quality nor the execution time seems to be altered when using different settings for the cache size.

This fact translates into the robustness of the EvAg model with respect to such a parameter and is specially remarkable from the point of view of tuning the parameters of the algorithm. Choosing any cache size within the range $[10 \ldots 50]$ will not alter the EvAg performance.



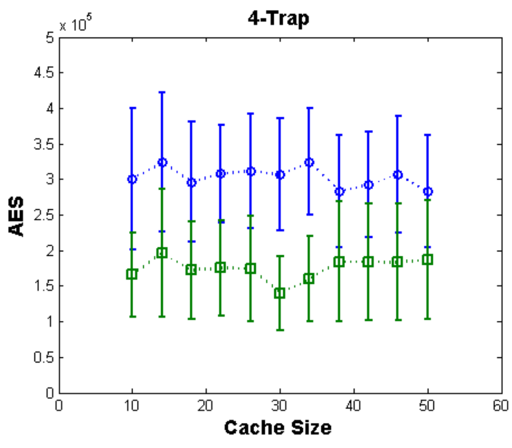


Fig. A.1 — Success Rate of the EvAg model (*left*) and Average Evaluation to Solution with standard deviation (*right*) for a 4-trap instance using population sizes of $N = 400$ and $N = 600$. Averaged values for the different cache sizes in dotted lines.

Appendix B

Numerical Results

In this appendix, numerical results for the scalability analysis of the different test cases are presented. The aim of providing such values is to allow a numerical comparison of results in addition to the respective one of the graphs depicted in chapter 5. As saw in Section 5.3.3, central position values (e.g. the mean) are not representative of the distributions on the computational efforts since they do not keep normality conditions. Therefore, the following tables provide the first, second and third quartiles of the distributions as values of reference.

Problem Instance	GGA				SSGA				EvAg			
	Selectorecombinative											
2-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	60	258	365	426	67	182	220	296	30	187	240	330
24	135	1495	1767	2136	165	886	974	1086	48	828	960	1092
36	170	2735	2906	3077	240	1813	1913	2052	75	1725	1875	2156
48	250	5019	5395	5772	300	2743	2906	3066	90	2700	2970	3577
60	320	7382	7832	8024	300	3318	3530	3801	135	4590	4927	5535
3-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	360	811	1443	2526	165	472	637	783	48	336	528	948
24	3120	53056	56177	64759	660	4678	5453	6034	105	2861	4200	5670
36	5280	142586	153148	163710	960	11346	12257	13596	195	10383	12772	15990
48	7680	268834	284196	299558	1440	23024	24810	26182	390	26227	28665	33345
60	11520	472360	483881	506923	1920	37986	39553	42100	480	39240	44880	49920
4-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	2880	2880	2880	5761	960	1502	2651	5470	60	510	1110	2880
16	53760	53760	53760	107521	3120	10475	18128	32272	-	-	-	-
24	-	-	-	-	-	-	-	-	225	33862	42525	55293
36	-	-	-	-	-	-	-	-	**600**	267750	314400	**393000**
	Using a mutation rate of $\frac{1}{L}$											
2-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	*	304	426	593	*	214	305	374	*	210	300	420
24	*	2039	2447	2885	*	1080	1244	1418	*	1008	1128	1248
36	*	4103	4616	5300	*	2270	2497	2736	*	2175	2475	2737
48	*	7278	7780	8031	*	3543	3709	3880	*	3330	3690	4320
60	*	10592	11232	11876	*	4283	4449	4722	*	5535	6075	6581
3-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	*	1082	1443	2887	*	456	599	826	*	432	528	720
24	*	86606	99871	109234	*	5925	6631	7454	*	3701	4830	6195
36	*	237644	253487	279892	*	15346	16638	18613	*	11700	15404	17842
48	*	422454	445497	466619	*	29021	30570	33498	*	32955	37050	43582
60	*	725822	748864	794948	*	48089	51137	54229	*	54360	61200	66480
4-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	*	2880	2880	5761	*	1526	2510	4517	*	675	1950	5205
16	*	53760	53760	107521	*	11055	21898	33353	*	-	-	-
24	*	-	-	-	*	-	-	-	*	32456	51750	69525
36	*	-	-	-	*	-	-	-	*	434550	531900	655350

Table B.1 — Test-Case 1: Number of evaluations to solution in Q_1, Q_2 and Q_3 for different trap functions and chromosome lengths L. Population sizes of the problem instances are estimated by bisection using a selectorecombinative GA.

Problem	EvAg				EvAg				EvAg			
Instance	Ring				Watts-Strogatz				Newscast			
2-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	30	210	255	330	30	187	240	330	30	187	240	330
24	37	999	1628	2136	41	738	861	1055	48	828	960	1092
36	41	2870	3567	4612	60	1440	1620	1980	75	1725	1875	2156
48	52	5226	7748	9451	97	2910	3492	3880	90	2700	2970	3577
60	60	11010	14850	17460	90	3600	3920	4410	**135**	4590	4927	**5535**
3-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
1241	369	738	1588	41	246	410	697	48	336	528	948	
24	60	6840	11730	17220	67	2646	3785	5845	105	2861	4200	5670
36	75	35943	58800	78393	135	8741	11542	16436	195	10383	12772	15990
48	90	138172	184005	272182	195	21206	37927	48701	390	26227	28665	33345
60	135	570780	770850	1203491	270	62167	94905	138307	**480**	39240	44880	**49920**
4-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	90	990	2565	7470	60	630	1500	2775	60	510	1110	2880
24	90	63922	98055	149355	150	27412	39150	48075	225	33862	42525	55293
36	225	957206	1304325	1989056	300	158325	193500	232650	**600**	267750	314400	**393000**

Table B.2 — Test-Case 2: Number of evaluations to solution in Q_1, Q_2 and Q_3 for different trap functions and chromosome lengths L. Population sizes of the problem instances are estimated by bisection using a selectorecombinative GA..

Problem Instance	Churn $\lambda = 400$				Churn $\lambda = 2500$				No Churn			
2-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	40	142	264	392	40	135	258	401	30	187	240	330
24	60	805	952	1150	65	752	956	1120	48	828	960	1092
36	100	1705	2019	2482	110	1895	2146	2642	75	1725	1875	2156
48	130	2815	3007	3625	130	2915	3309	3825	90	2700	2970	3577
60	200	4901	5149	6012	180	4870	5161	6030	**135**	4590	4927	**5535**
3-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	65	422	749	1050	65	437	653	1120	48	336	528	948
24	150	2980	4014	5723	120	3008	4203	5432	105	2861	4200	5670
36	360	8026	14955	21712	300	10702	13665	16722	195	10383	12772	15990
48	720	28334	33687	38137	440	22170	27310	31543	390	26227	28665	33345
60	1120	56971	61970	69790	720	46713	52646	60910	**480**	39240	44880	**49920**
4-Trap	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3	N	Q_1	Q_2	Q_3
12	120	2890	3349	5314	110	1080	2733	4712	60	510	1110	2880
24	320	38278	49862	64120	320	35738	52912	63615	225	33862	42525	55293
36	1280	28764	367377	451320	880	260914	385931	498027	**600**	267750	314400	**393000**

Table B.3 — Test-Case 3: Number of evaluations to solution in Q_1, Q_2 and Q_3 for different trap functions and chromosome lengths L. Population sizes of the problem instances are estimated by bisection using a selectorecombinative GA.

Bibliography

[1] D. Abramson, G. Millis, and S. Perkins. Parallelization of a genetic algorithm for the computation of efficient train schedules. In D. Arnold et al., editor, *Proceedings of the Parallel Computing and Transputers Conference*, pages 139–149. IOS Press, 1993.

[2] David H. Ackley. *A connectionist machine for genetic hillclimbing.* Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[3] Enrique Alba and Bernabe Dorronsoro. *Cellular Genetic Algorithms*, volume 42 of *Operations Research Computer Science Interfaces Series.* Springer, 2008.

[4] Enrique Alba and Marco Tomassini. Parallelism and evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 6(5):443–462, 2002.

[5] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.

[6] D.P. Anderson. BOINC: a system for public-resource computing and storage. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10, 2004.

[7] M.G. Arenas, Pierre Collet, A.E. Eiben, M. Jelasity, J.J. Merelo, Ben Paechter, Mike Preuss, and Marc Schoenauer. A framework for distributed evolutionary algorithms. In *Parallel Problem Solving from Nature - PPSN VII, Granada, Spain*, number 2439 in Lecture Notes in Computer Science, LNCS, pages 665–675. Springer-Verlag, 2002.

[8] Balázs Bánhelyi, Marco Biazzini, Alberto Montresor, and Márk Jelasity. Peer-to-peer optimization in large unreliable networks with branch-and-bound and particle swarms. In Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni A. Di Caro, Anikó Ekárt, Anna Isabel Esparcia-Alcázar, Muddassar Farooq, Andreas Fink, and Penousal Machado, editors, *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, pages 87–92. Springer, 2009.

[9] Balázs Bánhelyi, Marco Biazzini, Alberto Montresor, and Márk Jelasity. Peer-to-peer optimization in large unreliable networks with branch-and-bound and particle swarms. In *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, pages 87–92, Berlin, Heidelberg, 2009. Springer-Verlag.

[10] Johan Berntsson. G2DGA: an adaptive framework for internet-based distributed genetic algorithms. In *GECCO '05: Proceedings*

of the 2005 workshops on Genetic and evolutionary computation, pages 346–349, 2005.

[11] Bartosz Biskupski, Jim Dowling, and Jan Sacha. Properties and mechanisms of self-organizing manet and p2p systems. *ACM Trans. Auton. Adapt. Syst.*, 2(1):1, 2007.

[12] Erick Cantú-Paz. Topologies, migration rates, and multi-population parallel genetic algorithms. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 91–98, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[13] Erick Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[14] Erick Cantú-Paz and David E. Goldberg. Are multiple runs of genetic algorithms better than one? In *Genetic and Evolutionary Computation - GECCO 2003*, volume 2723 of *LNCS*, pages 801–812, 2003.

[15] K. Deb and D.E. Goldberg. Analyzing deception in trap functions. In *Foundations of Genetic Algorithms, Morgan Kaufmann*, pages 93–108. Morgan Kaufmann, 1991.

[16] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dy-

namo: amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220, 2007.

[17] B. Dorronsoro, E. Alba, M. Giacobini, and M. Tomassini. The influence of grid shape and asynchronicity on cellular evolutionary algorithms. In Y. Shi, editor, *IEEE International Conference on Evolutionary Computation*, pages 2152–2158, Portland, Oregon, June 20–23 2004. IEEE Press.

[18] AE. Eiben, AR. Griffioen, and E. Haasdijk. Population-based adaptive systems: concepts, issues, and the platform new ties. In *Proceedings of European Conference on Complex Systems, Dresden, Germany*, 2007. URL http://www.cs.vu.nl/g̃usz/papers/2007-ECCS-PAS.pdf.

[19] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* SpringerVerlag, 2003.

[20] L.J. Eshelman and J.D. Schaffer. Real-coded genetic algorithms and interval-schemata. In *Foundations of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann, San Mateo, CA, 1993.

[21] Carlos M. Fernandes and Agostinho C. Rosa. Genetic algorithms with assortative mating in static and dynamic environments. *Advances in Evolutionary Computation*, pages 181–206, 2008.

[22] L.J. Fogel. Evolutionary programming in perspective: the top-down view. In J.M. Zurada, R.J. Marks, and C. Robinson, editors, *Computational Intelligence: Imitating Life.* IEEE Press, Piscataway, NJ, 1994.

[23] Gianluigi Folino and Giandomenico Spezzano. P-cage: An environment for evolutionary computation in peer-to-peer systems. In Pierre Collet, Marco Tomassini, Marc Ebner, Steven Gustafson, and Anikó Ekárt, editors, *EuroGP*, volume 3905 of *Lecture Notes in Computer Science*, pages 341–350. Springer, 2006.

[24] Ian Foster. The anatomy of the grid: Enabling scalable virtual organizations. In *Euro-Par 2001 Parallel Processing*, volume 2150 of *LNCS*, pages 1–4. Springer Berlin / Heidelberg, 2001.

[25] Ian Foster and Adriana Iamnitchi. *Peer-to-Peer Systems II*, volume 2735 of *LNCS*, chapter On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing, pages 118–128. Springer Berlin / Heidelberg, 2003.

[26] Roy Friedman, Anne-Marie Kermarrec, and Michel Raynal. Modularity: a first class concept to address distributed systems. *SIGACT News*, 39(2):91–110, 2008.

[27] Christian Gagné, Marc Parizeau, and Marc Dubreuil. The master-slave architecture for evolutionary computations revisited. In *Genetic and Evolutionary Computation - GECCO 2003*, volume 2724 of *LNCS*, pages 1578–1579. Springer-Verlag Berlin Heidelberg, 2003.

[28] Salvador Garcia, Daniel Molina, Manuel Lozano, and Francisco Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'

2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6):617–644, 2009.

[29] The Gnutella Developer Forum GDF. The annotated gnutella protocol specification v0.4, 2001.

[30] M. Giacobini, M. Tomassini, A. Tettamanzi, and E. Alba. Selection intensity in cellular evolutionary algorithms for regular lattices. *IEEE Transactions on Evolutionary Computation*, 9(5):489–505, 2005.

[31] Mario Giacobini, Enrique Alba, Andrea Tettamanzi, and Marco Tomassini. Modeling selection intensity for toroidal cellular evolutionary algorithms. In *GECCO '04: Proceedings of the 2004 conference on Genetic and evolutionary computation*, LNCS, pages 1138–1149. Springer Berlin / Heidelberg, 2004.

[32] Mario Giacobini, Mike Preuss, and Marco Tomassini. Effects of scale-free and small-world topologies on binary coded self-adaptive CEA. In Jens Gottlieb and Günther R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2006*, volume 3906 of *LNCS*, pages 85–96, Budapest, 10-12 April 2006. Springer Verlag.

[33] Mario Giacobini, Marco Tomassini, and Andrea Tettamanzi. Takeover time curves in random and small-world structured populations. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1333–1340, New York, NY, USA, 2005. ACM.

[34] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[35] D.E. Goldberg. *The Design of Innovation - Lessons from and for Competent Genetic Algorithms.* Kluwer Academic Publishers, Norwell, MA, 2002.

[36] Martina Gorges-Schleuter. Asparagos an asynchronous parallel genetic optimization strategy. In *Proceedings of the third international conference on Genetic algorithms*, pages 422–427, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[37] G. Harik, E. Cantú-Paz, D.E. Goldberg, and B. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999.

[38] R. Hauser and R. Männer. Implementation of standard genetic algorithms on mimd machines. In *Parallel Problem Solving from Nature III*, LNCS, pages 504–514. Springer-Verlag, Berlin, 1994.

[39] I. Hidalgo and F. Fernández. Balancing the computation effort in genetic algorithms. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1645–1652. IEEE Press, 2005.

[40] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* Wiley Professional Computing, 1991.

[41] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 79–98, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[42] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[43] Márk Jelasity and Maarten van Steen. Large-scale newscast computing on the Internet. Technical Report IR-503, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, October 2002.

[44] Nicholas R. Jennings and Alex Rogers. *Intelligent Distributed Computing III*, chapter Computational Service Economies: Design and Applications, pages 1–7. Springer Berlin / Heidelberg, 2009.

[45] T. Klingberg and R. Manfredi. Gnutella 0.6 RFC, 2002.

[46] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, 1992.

[47] J.L.J. Laredo, P.A. Castillo, B. Paechter, A.M. Mora, E. Alfaro-Cid, A. Esparcia-Alcázar, and J.J. Merelo. Empirical validation of a gossiping communication mechanism for parallel EAs. In

EvoWorkshops, volume 4448 of *Lecture Notes in Computer Science*, pages 129–136. Springer, 2007.

[48] Wei-Po Lee. Parallelizing evolutionary computation: A mobile agent-based approach. *Expert Systems with Applications*, 32(2):318–328, 2007.

[49] Arnaud Legout, Guillaume Urvoy Keller, and Pietro Michiardi. Understanding BitTorrent: An Experimental Perspective. Technical Report, 2005.

[50] Feiyue Li, Bruce Golden, and Edward Wasil. Very large-scale vehicle routing: new test problems, algorithms, and results. *Comput. Oper. Res.*, 32(5):1165–1179, 2005.

[51] Gao Lisha and Luo Junzhou. Performance analysis of a P2P-based VoIP software. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 11–16, 2006.

[52] Hong Liu and P. Mouchtaris. Voice over ip signaling: H.323 and beyond. *Communications Magazine, IEEE*, 38(10):142–148, Oct 2000.

[53] Fernando G. Lobo and Claudio F. Lima. Adaptive population sizing schemes in genetic algorithms. In *Parameter Setting in Evolutionary Algorithms*, Studies in Computational Intelligence, pages 185–204. Springer Berlin / Heidelberg, 2007.

[54] J.J. Merelo, P.A. Castillo, J.L.J. Laredo, A.M. Mora, and A. Prieto. Asynchronous distributed genetic algorithms with Javascript and JSON. In *IEEE Congress on Evolutionary Computation (CEC2008), WCCI2008 Proceedings*, pages 1372–1379. IEEE Press, Hong Kong, June 2008.

[55] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK, 1996.

[56] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications. SIAM, 2000.

[57] Martin Pelikan, Kumara Sastry, and David E. Goldberg. Scalability of the bayesian optimization algorithm. *International Journal of Approximate Reasoning*, 31(3):221–258, 2002.

[58] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. In *ACM SIGCOMM*, pages 161–172, 2001.

[59] I. Rechenberg. Evolution strategy. In J.M. Zurada, R.J. Marks, and C. Robinson, editors, *Computational Intelligence: Imitating Life*. IEEE Press, Piscataway, NJ, 1994.

[60] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[61] Stefan Saroiu, Krishna P. Gummadi, and Steven D. Gribble. Measuring and analyzing the characteristics of napster and gnutella hosts. *Multimedia Systems*, 9(2):170–184, 2003.

[62] K. Sastry. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Technical Report 2002004, University of Illinois at Urbana-Champaign, Urbana, IL., 2001.

[63] Birgitt Schönfisch and André de Roos. Synchronous and asynchronous updating in cellular automata. *Biosystems*, 51(3):123 – 143, 1999.

[64] H. Schulzrinne and J. Rosenberg. The session initiation protocol: Internet-centric signaling. *Communications Magazine, IEEE*, 38(10):134–141, Oct 2000.

[65] Ian Scriven, Andrew Lewis, and Sanaz Mostaghim. Dynamic search initialisation strategies for multi-objective optimisation in peer-to-peer networks. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pages 1515–1522, Piscataway, NJ, USA, 2009. IEEE Press.

[66] Ralf Steinmetz and Klaus Wehrle, editors. *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*. Springer, 2005.

[67] Ralf Steinmetz and Klaus Wehrle. What is this peer-to-peer about? In *Peer-to-Peer Systems and Applications* [66], pages 9–16.

[68] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[69] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM on Internet measurement (IMC 2006)*, pages 189–202, New York, NY, USA, 2006. ACM Press.

[70] Dirk Thierens. Scalability problems of simple genetic algorithms. *Evolutionary Computation*, 7(4):331–352, 1999.

[71] Marco Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[72] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, 2009.

[73] Spyros Voulgaris, Márk Jelasity, and Maarten van Steen. *Agents and Peer-to-Peer Computing*, volume 2872 of *Lecture Notes in Computer Science (LNCS)*, chapter A Robust and Scalable Peer-to-Peer Gossiping Protocol, pages 47–58. Springer Berlin / Heidelberg, 2004.

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

[75] J.M. Whitacre, R.A. Sarker, and Q.T. Pham. The self-organization of interaction networks for nature-inspired optimization. *Evolutionary Computation, IEEE Transactions on*, 12(2):220–230, April 2008.

[76] W. R. M. U. K. Wickramasinghe, M. van Steen, and A. E. Eiben. Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In *GECCO '07*, pages 1460–1467, New York, NY, USA, 2007. ACM Press.

[77] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.