UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

DEPARTAMENTO DE ARQUITECTURA DE COMPUTADORES Y AUTOMÁTICA

ALGORITMOS MEMÉTICOS PARALELOS PARA EL PROBLEMA DE DISTRIBUCIÓN DE ESFUERZO EN SISTEMAS MULTI-AGENTE DINÁMICOS

PARALLEL MEMETIC ALGORITHMS FOR THE PROBLEM OF WORKFORCE DISTRIBUTION IN DYNAMIS MULTI-AGENT SYSTEM

TESIS DOCTORAL DE:

DAVID MILLÁN RUIZ

DIRIGIDA POR:

JOSÉ IGNACIO HIDALGO PÉREZ

Madrid, 2013

Algoritmos Meméticos Paralelos para el Problema de Distribución de Esfuerzo en Sistemas Multi-Agente Dinámicos

(Parallel Memetic Algorithms for the Problem of Workforce Distribution in Dynamic Multi-Agent Systems)

by

David Millán Ruiz

Thesis submitted to the Complutense University of Madrid in partial fulfillment of the requirements for the degree of

Ph.D. in Computer Science

Architecture and Technology of Computing Systems

Facultad de Informática

Supervisor: Professor Dr. José Ignacio Hidalgo Pérez

Universidad Complutense de Madrid

June, 2013

Abstract

This thesis describes a novel approach to the problem of workforce distribution in dynamic multi-agent systems based on blackboard architectures, focusing especially on a real-world scenario: the multi-skill call centre. Traditionally, to address such highly-dynamic environments, diverse greedy heuristics have been applied to provide solutions in real-time. Basically, these heuristics perform a continuous re-planning on the system, taking into account its current state at all times. As decisions are greedily taken, the distribution of the workforce may be poor in the medium and/or long term. The usage of parallel memetic algorithms, which are more sophisticated than standard ad-hoc heuristics, can lead us towards much more accurate solutions. In order to effectively apply parallel memetic algorithms to such a dynamic environment, we introduce the concept of adaptive time window. Thus, the size of the time window depends upon the level of dynamism of the system at a given time. This thesis proposes a set of tools to automatically determine the dynamism of the system, as well as a novel and precise prediction module based on a neural network and a powerful search method based on parallel meta-memetic algorithms to cope with complex dynamic systems. To conclude, we compare our approach with other techniques from the state-of-the-art in a real-world production environment (Telefónica's call centre), performing better than other conventional techniques. We also provide a thorough study of each of the modules that compose our global solution.

Resumen

Esta tesis describe un novedoso enfoque para resolver el problema de distribución de carga de trabajo en sistemas multi-agente dinámicos basados en arquitecturas de pizarra, enfocándose especialmente en un escenario real: el call center multitarea. Para abordar este tipo de entornos dinámicos, tradicionalmente se han aplicado diversas heurísticas voraces que permiten dar una solución en tiempo real. Básicamente, dichas heurísticas realizan replanificaciones continuamente, considerando el estado del sistema en cada momento. Como las decisiones se toman de forma voraz sin hacer una planificación óptima, la distribución de la carga de trabajo puede ser pobre a medio y/o largo plazo. El uso de algoritmos meméticos paralelos nos puede permitir encontrar soluciones mucho más precisas. Para aplicar este tipo de algoritmos, introducimos el concepto de ventana temporal adaptativa. De esta forma, el tamaño de la ventana temporal depende del nivel de dinamismo del sistema en un instante dado. Este trabajo propone una serie de herramientas para determinar el dinamismo del sistema de forma automática, así como un novedoso módulo de predicción basado en una red neuronal y un potente método de búsqueda basado en meta-algoritmos meméticos paralelos para poder lidiar con entornos dinámicos complejos. Para concluir, comparamos nuestro enfoque con otras técnicas del estado del arte en un entorno de producción real (Telefónica) obteniendo mejores resultados que el resto de técnicas actuales. También se proporciona un estudio exhaustivo de cada uno de los módulos.

Declaration of Authorship

The author, David Millán Ruiz, hereby declares and confirms that this thesis is entirely the result of the work carried out in the Department of Architecture and Technology of Computing Systems of the School of Computer Science at the Complutense University of Madrid. This thesis contains original contribution by the author unless otherwise indicated.

David Millán Ruiz,

June 2013.

Acknowledgement

This thesis has been carried out in the Department of Architecture and Technology of Computing Systems of the School of Computer Science at the Complutense University of Madrid, under the supervision of Prof. Dr. José I. Hidalgo.
Firstly, I would like to thank Prof. Dr. José I. Hidalgo for giving me the opportunity to perform this research under his supervision. He supported me and guided me from the beginning of this work, giving me many valuable and useful comments to progress in my research. He has been much more than a mere supervisor...he has been a real friend in every sense of the word. Besides, I would like to extend my gratitude to the whole Group of Parallel Architectures and Bioinspired Algorithms (PABA) for their support...Nacho, Iván, Javi, Josele, Juan, Alfredo, Chema and Oscar, many thanks!
Secondly, I would also like to thank the Department of Customer Care of Telefónica Research & Development, headed by Mr. Javier Bonastre, for their support to this work and for the infrastructure provided (mainly for the Sun Fire E4900 server). Namely, Mr. Jorge Pacheco, Mr. José L. Vélez and Mr. Jesús Celis deserve a special mention for their effort in helping me out with the adaptation of some of my models and algorithms to the systems of Telefónica.
Thirdly, it is mandatory to highlight the priceless help of Prof. Dr. Severino F. Galán from the School of Informatics at the Spanish National University for Distance Education (UNED). He helped me to channel this thesis into the right direction.

Fourthly, I would like to thank Prof. Dr. Mark Jerrum from the University of Edinburgh because he was the "seed" of all this work. He gave me the first guidelines for some of my models when I did my master's dissertation at the University of Edinburgh.
Last but not least, I also want to thank the reading committee, for the time spent in carefully reviewing this thesis.

Dedication

In loving dedication to my girlfriend, Dóra Lajkó, who encouraged and gave me the required strength to finish this work.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Abbreviations

ACO	Ant colony optimisation
ANOVA	Analysis of variance
ANN	Artificial neural network
ARIMA	Autoregressive integrated moving average
BB	Branch and bound
BPA	Back-propagation
CC	Call centre
CG	Call group
COP	Constraint optimisation problem
DMAS	Dynamic multi-agent system
DP	Dynamic programming
DTTS	Dumped trend time series
EA	Evolutionary algorithm
ES	Exponential smoothing
GA	Genetic algorithm
GRASP	Greedy randomised adaptive search
GWB	Greedy workload balancing
ILS	Iterated local search
LogR	Logistic regression
LR	Lineal regression
LS	Local search
MA	Memetic algorithm
MAE	Mean absolute error
MAS	Multi-agent system
MH	Meta-heuristic
MSS	Multi-start search
MSCC	Multi-skill call centre
MSE	Mean square error
PCA	Principal component analysis
PD	Poisson distribution
PGA	Parallel genetic algorithm
PMA	Parallel memetic algorithm
RM	Regression model
Rprop	Resilient back-propagation
RWB	Random workload balancing
SA	Simulated annealing
SBR	Skill-based routing
SES	Simple time series
SS	Scatter search
STS	Stationary time series
TaS	Tabu search
TS	Time series
uRprop	Upgraded resilient back-propagation
VND	Variable neighbourhood descent
VNS	Variable neighbourhood search
WCOP	Weighted constraint optimisation problem

Chapter 1

Introduction

1.1 Overview

Over the last years, a gradually-growing interest in parallel and distributed computing has arisen in computer science. Specifically, this concern has recently guided most of the research activities towards areas such as parallel and distributed programming, distributed information systems, and parallel and distributed hardware architectures and infrastructure. As this tidal wave has swept across many industries, distributed computing has arisen as the new ubiquitous term. Truthfully, there exists a vast bibliography on parallel and distributed computing (see Chapter 3) although there are still paths to explore [(16)] [(143)] [(6)].

Furthermore, we perceive a tendency to tackle increasingly complex problems and application domains, which frequently entail the processing of extremely dynamic data flows. These demanding environments are usually hard to be efficiently handled by most of the existing, sequential techniques. In this context, parallel and distributed evolutionary algorithms do not only mitigate this drawback but also present several noteworthy characteristics such as robustness, traceability, problem simplification, adaptivity, scalability and speed-up.

Nevertheless, it is not always straightforward to synchronise and control the internal dynamics of parallel and distributed information systems because the representation of linear problems into sub-problems is not always feasible or a simple task.

Anyhow, parallel and distributed systems should somehow self-improve to attain high performance. In fact, nowadays, a wide range of studies on adaptive techniques in parallel and distributed information systems can be found [(37)] [(11)].

A classical, well-suited problem for studying dynamic systems is *the problem of workforce distribution in multi-agent systems*. Agents can work for a common goal, coordinate the plans or draw up a plan for others' tasks. Although there are many types of multi-agent systems, we will focus on the blackboard architectural model [(55)] [(81)] (see Figure 7.2). In other words, we will work on systems with a common repository of knowledge, the "blackboard", which is iteratively updated by a diverse group of agents.

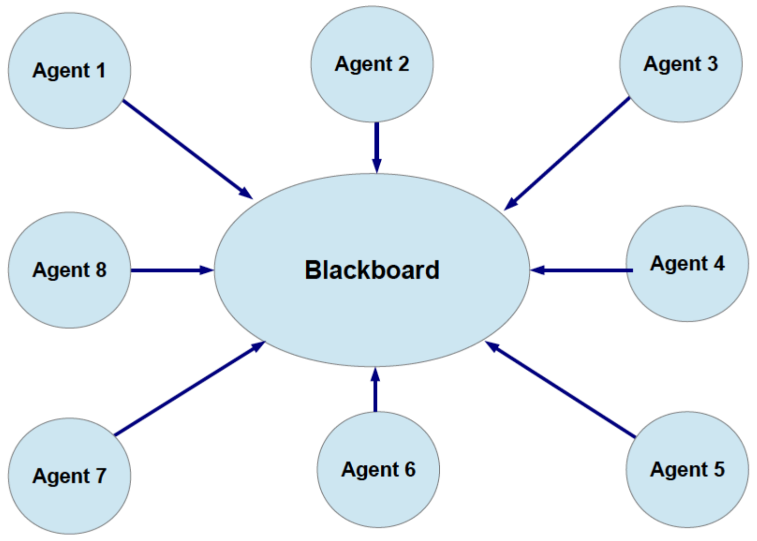

Figure 1.1: Example of the blackboard architectural model with *8* agents.

The *basic variant* of a workforce distribution problem, which is related to the *family of assignment problems*, requires the assignment of tasks to the agents who have the required skills to handle them over time, satisfying a given set of additional constraints and respecting the dependencies among individual tasks and the differences in the execution skills of the agents. This problem has multiple variants but, depending on the dynamism of the system, we can principally distinguish **two main scenarios**:

1. On the one hand, we can find **short-term** planning environments in which a continuous planning is needed due to the high dynamism of the system. These solutions attempt to distribute the workload among agents by applying "basic" ad-hoc heuristics, looking at the current situation (without predictions or predictions for a short time-frame). This feature can be effortlessly seen in workload allocation within a *dynamic multi-skill call centre* [(15)].

2. On the other hand, we can find **long-term** planning systems in which the list of tasks is predefined and known by all agents like in the *classic scheduling problem* [(28)]; or environments in which a single task type is assigned to each agent for a long period of time, similarly to the *job assignment problem* [(36)]. In other cases, agents are assigned to patterns of tasks, instead of specific tasks (such as in *pattern-based scheduling* [(28)]). Analogously, stable multi-skill call centres [(15)] can be also included in this group. These solutions consider stable behaviour over time, anchored in historical data and apply more complex algorithms to match agents and task types. However, when having a dynamic system, these approaches cannot be efficiently applied, since an adaptive method is required.

The problem is clearly a dynamic optimisation problem with constraints, so that our proposal is thus encapsulated in the first scenario: dynamic systems. We put forward an alternative approach to traditional solutions, which relies on an **adaptive time-frame**. This can be seen as a **middle-term** planning where the time-frame is adjusted to the changes of the system: when the dynamism is very low, it is analogous to having a long-term time-frame, whereas a very high dynamism leads to a short-term time-frame. In other words, we dynamically enlarge or diminish the time-frame considered to better adapt the algorithm to the current state of the system. Figure 7.1 illustrates where our approach is positioned.

Besides, we provide the required mechanisms to implement this more efficient, adaptive solution. Although this solution can be extended to countless domains and multi-agent systems, we will go over the call centre application in order to examine the idiosyncrasy and complexity of a real-world problem.

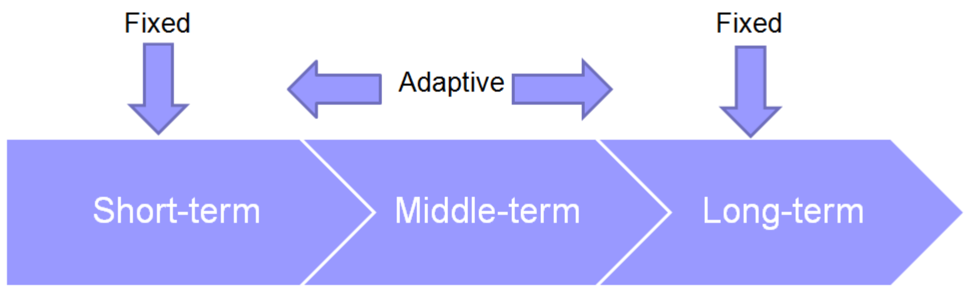

Figure 1.2: Adaptive time-frame mechanism.

Table 1 summarises some fundamental characteristics of the previously described scenarios in relation to the time-frame considered.

Table 1.1: Comparison of the time-frame considered for the workforce distribution problem.

Time-frame	*Complexity*	*Response time*	*Adaptability*	*Performance*	*CPU Utilisation*
Short-term	low	low	medium	medium	low
Middle-term	high	medium	high	high	high
Long-term	medium	high	low	low	high

To conclude this outline, we would like to stress that the present study has been applied to a real-world production environment, which is fed by real-time data. This production environment belongs to Telefónica (http://www.telefonica.com), the sponsor of this work.

Telefónica is one of the world's largest telecommunications companies by market capital and number of customers. Its activities are mainly centred on the fixed and mobile telephony businesses, while its broadband business is the key growth driver, underpinning both. It operates in *25* countries and its customer base exceeds *300* million people worldwide. Telefónica's growth strategy is focused on the markets in which it has a strong foothold: Spain, Europe and Latin America. Telefónica group stands in third position worldwide in the telecom sector in terms of market capitalisation, the 1^{st} as an European integrated operator and also the second company in the Eurostoxx *50* ranking, composed of the major companies in Europe (December 31^{st} 2011). All this makes Telefónica's call centre an extremely complex net of interconnected sub-call centres.

1.2 Motivation & Main Objectives

The problem of workforce distribution in multi-agent systems is an appealing and challenging subject of research, not only from a technical point of view but also from a business impact angle. The eminent complexity of this problem makes it even more interesting and a firm member of the class of NP-hard problems [(63)]. Besides, timing constraints complicate, even more, finding an accurate, feasible solution. Another reason to analyse this problem is that it is often omnipresent in our daily life and is highly relevant to many industrial application domains like trading and workflow organisation, entailing a huge business impact.

From a parallel computing angle, this problem is also tempting since it inherently allows for parallelism because the tasks to handle can be distributed over several computing nodes and also because the computing nodes can execute different tasks in parallel.

From an artificial intelligence point of view, this problem is also very motivating because it involves many fields, which range from forecasting techniques derived from machine learning theory to optimisation algorithms that use diversity maintenance techniques from evolutionary computation and other local search schemes like simulated annealing or tabu search.

The main purpose of this work is to provide a solution, which is fully described in Chapter 4, for dynamic multi-agent systems based on blackboard architectures. Thus, an efficient forecasting method must be provided in order to predict the real situation in next time-frame (future system state) and, therefore, an adaptive optimisation algorithm must be performed to determine the right assignment <task-agent>.

1.3 Problems with Current Solutions

Existing solutions present one or more of the following problems and limitations:

- *Lack of planning*: the allocation <job-agent> is performed according to the current state of the system, without taking into account forthcoming changes. Some authors [(37)] perform very short-term predictions so that the techniques to reallocate jobs and agents must be very simple like the skill-based routing.

- *Non-adaptive*: other authors (see Chapter 3) apply complex meta-heuristics (like simulated annealing, tabu search, scatter search or other complex local search techniques) to get a stable allocation of agents to job types considering a specific state of the system. This allocation must be stable as long as the techniques require long computing times.

- *Non-scalable*: some of the existing techniques of the state-of-the-art are not parallelisable (e.g. [(22)] [(37)]) and do not scale up properly.

- *Lack of robustness in the results*: most of the techniques consider stable allocations for a truly changing environment (e.g. [(106)]). In contrast, other techniques allow for dynamic allocations (e.g. [(37)]) but the distribution often sharply fluctuates with new patterns or load peaks.

- *Non-fault tolerant*: some techniques do not keep alternative solutions in case of system failure, stopping the system (e.g. [(22)]).

- *Lack of granularity*: most techniques (e.g. [(22)] [(106)]) cannot work with individual agents or individual calls. In order to reduce the complexity, those techniques group agents and types of calls (following a coarse-grain model). This way, the allocation between agents and tasks cannot be *100%* accurate.

- *Lack of preferences personalisation*: traditional process management systems rigidly distribute tasks to queues from which agents take and process work (e.g. [(37)]), regularly without having the chance to opt for the precise tasks they actually desire to cope with.

- *Hard to incorporate job/agent constraints and business rules*: most of the techniques (e.g. [(106)]) do not allow for injecting business rules and handling job/agent constraints.

1.4 Key Aspects to Overcome Existing Limitations

To overcome the limitations of current solutions, the right approach to the problem of workforce distribution should coalesce forecasting with optimisation, thereby considering an adaptive middle time-frame mechanism, so that the forecasting component can provide the optimisation piece with more time to find a better solution.

Of course, a perfect method should consist of an accurate forecasting module that minimises the error in the predictions for future system states (artificial neural networks can be of valuable help).

This way, the ideal approach should have a robust, self-adapted method to handle the changes of the system over time (for instance, time series for determining the dynamism of the system and a Markov model for performing the system transitions). With such an approach, we would be able to check the stability of the system and inform the forecasting module about the size of the prediction window.

To scale-up properly and find an optimal solution for forthcoming system states, the method should be based on a powerful, parallelisable approach like parallel memetic algorithms. In order to better configure the parallel memetic algorithm in dynamic environments, we may need a meta-genetic algorithm to automatically fine-tune all its internal parameters.

Typically, traditional process management systems distribute tasks to queues from which agents pull work, regularly without opting for the precise tasks they actually desire to deal with. Instead, the right approach should enhance workforce distribution by additionally injecting real-time knowledge of the task, individual skill sets, preferences, business rules and constraints, and availability and utilisation of the workforce, allowing for dynamic and active distribution of tasks over time.

Additionally, the ideal method should provide further clearness on customer service level agreements and offer outstanding customer service.

1.5 Market Relevance

The market relevance of the present work can be devised from three distinct angles: *customer satisfaction* (happy customers remain loyal to their telecommunications provider), *optimisation of resources* (monetary savings as a result of a better workforce distribution) and *employee satisfaction* (brand pride, self-esteemed fortification and fair workload allocation). Figure 1.3 shows the three main bases of success for any telecommunications operator.

Our research has revealed that the lack of a proper planning in inbound traffic management can reduce customer satisfaction by *10* to *15* percent, and employee satisfaction by *20* to *30* percent. Both figures are cause-for-concern as customer satisfaction is paramount in the present times, and soaring attrition rates give sleepless nights to call centre management. Consequently, it is very important for managers and supervisors to discern when to switch an agent, and how to achieve it effectively.

Other studies ([(4)] [(46)]) prove that users' key period to migrate to another telecommunications operator (also denoted as *churn*) after having a negative experience with call centre's contact service is, for the majority of individuals, about *10* days from the notification date. During these days, and even afterwards, these people negatively influence their social circles or communities, causing a cascade effect, which implies huge losses of money to telecommunications operators every year.

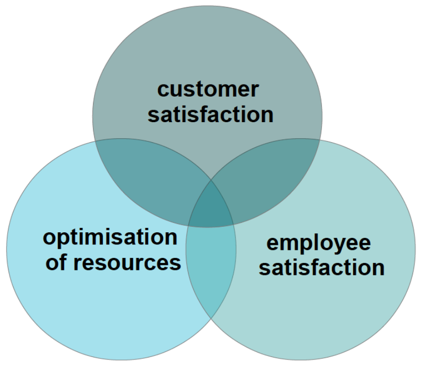

Figure 1.3: Key factors of the market relevance.

If an organisation is planning to link up with any existing outbound churn prediction models, based on likelihood to churn, then there appears the need of having a process for risk prioritisation (potential churners are prioritised) built, if the company has capacity constraints in its outbound call centre (when there are not enough resources to directly contact potential churners, an automatic risk prioritisation is needed in the inbound call centre).

However, we should highlight we cannot evade churn effect as far as it decidedly depends upon multiple individual reasons. There, call centres can play an important role in churn prevention as a consequence of customer satisfaction enhancement, which irrefutably leads us to customer loyalty.

By upgrading customer service with our approach, Telefónica has estimated savings up to €*2,000,000* per year only in Spain, as it enhances brand loyalty (customers are happier with their telecommunications operator) and other encouraging behaviours such as the word-of-mouth advocacy.

Nevertheless, customer satisfaction is not the unique edge from where we can profit. Another important aspect refers to the optimisation of resources we are actually doing because we increase the speaking level of each agent (more time on the phone). If we consider the mean upgrading percentage obtained by our approach in 2010 (*7%*), we can affirm that, only in Spain, it is possible to obtain savings up to *€3,000,000* per year.

Enhancing employee satisfaction can also be of extraordinary benefit to any telecommunications operator, as happy agents will be prone to reply to more incoming calls and stay loyal to the company. The workforce distribution is a key factor to keep agents much more satisfied and motivated because all the agents have a similar workload.

As a final point, the market relevance of this work can be extended to many other dynamic multi-agent systems in which the list of tasks is not predefined such as plane maintenance [(74)], online trading [(75)], disaster response [(125)], congestion in stations [(149)] or overloading in networking nodes [(167)].

1.6 Organisation of this Thesis

The rest of this document is organised as follows:

- Chapter 2 introduces the problem of workforce distribution in dynamic multi-agent systems from a generic point of view in Section 2.1 and from formal perspective in Section 2.2. In Section 2.3, we address the difficulty of handling hard and soft constraints. Section 2.4 describes the specific characteristics of our problem domain: the call centre.

- Chapter 3 kindly discusses the state-of-the-art and presents the required background for situating this work and making a proper understanding of it. A brief survey of DMAS algorithms is given in Section 3.1. Section 3.2 provides a study of related work on forecasting from diverse angles, considering commonalities with other problem domains. Section 3.3 reviews the most representative combinatorial optimisation techniques to the best of our knowledge.

- Chapter 4 puts forward a novel approach to the problem of workforce distribution in DMAS. Section 4.1 sets out the underlying idea of this new approach. In Section 4.2, we explain the methodology that we have applied. Section 4.3 describes how to create an adaptive time-frame method. Section 4.4 focuses on the forecasting component. Section 4.5 addresses the search component of the proposed approach.

- In Chapter 5, we adapt our approach to the multi-skill call centre. Section 5.1 presents some special adaptations for the forecast module. In contrast, Section 5.2 points out some particular adaptations for the search module.

- Chapter 6 covers the experimental evaluation of our approach. Section 6.1 highlights the magnitude, in terms of volume, of our application domain. Section 6.2

describes the dataset employed. Section 6.3 points out the hardware descriptions of the SunFire sever in which the evaluations have been performed. Section 6.4 analyses the selected metrics for testing and comparing our approach. Section 6.5 examines the forecast module for five different CGs as there are too many CGs to accomplish an exhaustive study for all of them. Section 6.6 evaluates the search module by studying several time intervals from days with different complexity; this section also compares our search module with other acknowledged techniques. In Section 6.7, we analyse our complete approach (forecast module + search module) for one-day campaign. We also demonstrate how our complete approach outperforms other conventional call centre algorithms.

- Chapter 7 concludes our work with a summary of ideas exposed in this thesis (in Section 7.1). This chapter also presents the major contributions in Section 7.2 and points out prospects for future work in Section 7.3.

Chapter 2

The Problem of Workforce Distribution in Dynamic Multi-Agent Systems

The present chapter expounds the problem of workforce distribution in dynamic multi-agent systems from different perspectives. The main aim of this chapter is to introduce this problem to the reader as the pillars of this work rely on the concepts given throughout this section.

Section 2.1 presents the problem from a generic point of view. Section 2.2 formalises the problem definition in order to provide the present work with a higher level of scientific rigour. In Section 2.3, we tackle the difficulty of handling hard and soft constraints as it is the typical situation in real-world environments. Finally, Section 2.4 describes the specific characteristics of the call centre.

2.1 Generic Definition of the Problem

The term *intelligent agent* ([(59)]) describes an autonomous entity, which is able to observe and interact with its environment in order to accomplish a given set of tasks [(152)]. Intelligent agents may also learn from their environment or use previous knowledge of the domain to achieve their goals. Their complexity can range from very simple systems to very complex ones. Unlike an object, which is defined in terms of methods and attributes, an agent is defined in terms of its behaviour.

Different authors (e.g. [(59)] [(152)] [(97)]) have proposed diverse definitions of agents, which commonly include concepts such as *persistence* (code is not executed on demand and decides for itself when it should perform a given activity), *veracity* (an agent cannot communicate false information), *kindness* (agents do not have conflicting goals), *rationality* (agents will act in order to achieve their goals), learning (agents improve performance over time), *autonomy* (agents have capabilities of task selection, prioritisation and goal-oriented behaviour), *sociability* (agents are able to engage other components through some sort of communication and coordination, so that they may collaborate on a task) and *reactivity* (agents perceive the context in which they operate and react to it appropriately).

When several agents interact, these may compile a **Multi-Agent System** (MAS) [(181)]. Characteristically, such agents have a partial point of view of the problem and thus need to cooperate with other agents. Furthermore, there may be no global control and thus such systems are sometimes denoted as *swarm systems*. In those cases, data is decentralised and execution is asynchronous.

The real world is actually a multi-agent environment because we often need to co-operate with others in order to achieve our own goals. In fact, many goals can be only achieved with the cooperation of others. Social ability in agents is the ability to inter-act with other agents (and possibly humans) via some kind of agent-communication language.

Commonly, the basic variant of the workforce distribution problem in a **Dynamic Multi-Agent System** (DMAS) requires the assignment of tasks to the agents that have the required skills to handle them over time, satisfying a predefined set of additional constraints and respecting the dependencies among individual tasks and the differences in the execution skills of the agents.

In a common DMAS, there are n tasks or work items grouped in k types of tasks and m agents that may have up to l skills ($l \leq k$) to perform these works. In this manner, each agent can process different types of tasks and, given a type of task, it can be carried out by several agents that have that skill. The set of skills an agent has is frequently denoted as its *profile*. These profiles can be truly heterogeneous as there are massive potential skills. Although agents may have multiple skills, each agent can only process one operation at the same time. Given an operation, it requires an unknown amount of time to be accomplished. Furthermore, each agent must orderly process each operation during an uninterrupted period of time; in other words, the task cannot be divided or postponed once it has already started.

Constraints may be given by many factors that we cannot cover in this section as this issue is problem dependent. However, we will describe how we propose to deal with them in Section 2.3. The solution to the problem of workforce distribution in dynamic multi-agent systems consists in dynamically assigning every task (according to its type) to the right agent, so that this solution satisfies all hard constraints and respects, if possible, all soft constraints.

Eventually, we need a metric of quality to measure the rightness of each solution. Of course, the definition of the quality function is problem dependent too. In next chapters, we will show an example of quality function for the dynamic multi-skill call centre use case.

2.2 Formal Definition of the Problem

Formalising the definition given in Section 2.1, we can find the following parameters in a dynamic multi-agent system based on the blackboard architecture:

1. Let W be a finite set of $n \epsilon \mathbb{N}$ tasks or work items: $W = \{w_1, w_2, ..., w_n\}$.

2. Let T be a finite set of $k \epsilon \mathbb{N}$ task types $T = \{t_1, t_2, ..., t_k\}$, where $k \leq n$ when every task type has, at least, one task assigned.

3. Let A be a finite set of $m \epsilon \mathbb{N}$ agents $A = \{a_1, a_2, ..., a_m\}$.

4. Let S be a finite set of $k \epsilon \mathbb{N}$ agent-skills $S = \{s_1, s_2, ..., s_k\}$ in which each agent-skill, s_i, represents the capability to handle the corresponding type of task, t_i, with the equivalent sub-index in T. $T \sim S : s_1 \sim t_1, s_2 \sim t_2, ..., s_k \sim t_k$. In other words, there is a bijection between tasks and skills.

5. Let P be a finite set of $d \epsilon \mathbb{N}$ agent-skill profiles $P = \{P_1, P_2, ..., P_d\}$ in which each agent-skill profile $P_i \subset S$ can be any subset of $S = \{s_1, s_2, ..., s_k\}$.

6. Let O be a finite set of $n \epsilon \mathbb{N}$ operations (execution or processing of each task, w_i) $O = \{o_1, o_2, ..., o_n\}$ in which each operation, o_i, has associated a processing time, which depends on its type of task: $T = \{t_1, t_2, ..., t_k\}$.

Moreover, the solution must fulfil the following descriptions:

1. every agent, a_i, has associated a finite non-null subset of P, A_P, containing its skills to handle different types of tasks (individual skill-profile): $(a_i \sim A_P \subset P)$.

2. on O define R, a binary relation, which represents the precedence among operations. If $(o_i, o_j) \epsilon R$ then o_i has to be performed before o_j.

3. the same profile P_i can be assigned to several agents. In other words, several agents may have some skills in common (or even all of them): $(1...*)P_i \sim (1...*)a_j$.

4. although every agent, a_i, may have several profiles assigned, only one can be used at a given instant t, $< a_i, P_j >_t$.

The goal is to obtain, by means of the maximisation of a quality metric Q_M (problem dependent), the right assignment between every agent a_i and the required profile P_j for each time-frame considered (v seconds): $Max[Q_M(\Sigma a_i, \Sigma P_j, t)]$. In addition, the assignment $< a_i, P_j >_t$ must satisfy all hard constraints and handle the soft ones.

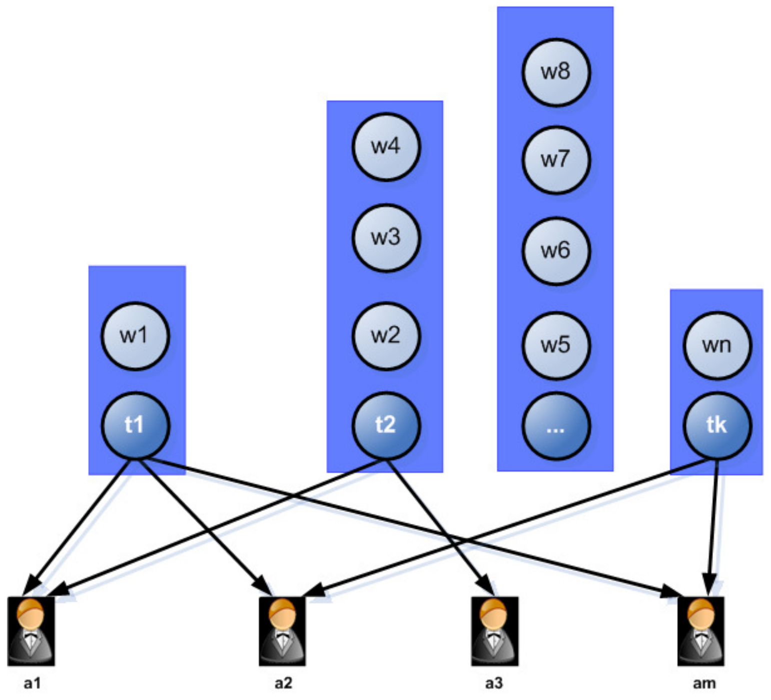

Figure 2.1: Multi-agent system configuration based on the potential skills of all agents.

Figure 7.3 illustrates a feasible solution for a given time-frame, supposing that agent a_1 has the skills to process the tasks t_1 and t_2 (through its skills s_1 and s_2), agent a_2 has the skills to process tasks t_1 and t_k (s_1 and s_k), agent a_3 has the skill to process the task t_2 (s_2) and agent a_m has the skills to process the tasks t_1 and t_k (s_1 and s_k).

We have presented an example in which each agent has certain potential skills (at least one) to attend some tasks types. The fact that a given agent has multiple skills does not mean he must attend all these types at the same time within a given interval (do not confuse potential skills with currently assigned skills).

2.3 Constrainted Optimisation Problem

The problem described in Section 2.2 can be viewed as a *Constrainted Optimisation Problem* (COP) [(47)]. A COP is characterised by a set of v variables, $X = \{x_1, x_2, ..., x_v\}$ and a set of c constraints $Co = \{co_1, co_2, ..., co_c\} \subset D_i$ for a nonempty domain D_i of feasible values.

A system state is defined by an assignment of values to some (or all) variables. An assignment that does not violate any constraints is denoted as *consistent* or *legal assignment.* A *complete assignment* is one in which every variable is mentioned, and the solution satisfies all the constraints. In our case, the constraints are associated to the tasks, the agents, timing, actions or desired/undesired situations.

Classic COPs treat every single constraint as hard, referring to the fact that each feasible solution must satisfy all constraints. In contrast, *flexible COPs* relax this assumption by partially relaxing constraints and allowing the solution not to comply with all them (soft constraints).

We consider the *Weighted Constrainted Optimisation Problem* (WCOP) in the present work, in which each violation of a soft constraint is weighted according to a predefined relevance (relevance is usually given by the business units of a company). Consequently, satisfying soft constraints with greater weight is preferred, whereas hard constraints cannot be violated in any case. The violation of soft constraints is penalised according to the degree of non-accomplishment of these constraints and their relevance.

Weights can be assigned by defining *level of constraints.* For each level, we can define a range for the weights (constraint relevance) and the gap between two levels follows a *logarithmic function* in order to soften the difference among levels. Different levels cannot have the same relevance (no overlapping constraints levels) and determining the difference among levels is frequently a business driven action according to the market relevance. The values for a given level should be proportionally assigned.

For a better understanding, we present an example of the relationship among constraints and levels of constraints for *4* levels and *7* constraints (see Figure 2.2). In our example, *Level 4* (the most relevant level) has two constraints (C1 and C2), *Level 3* and *Level 2* have just one constraint (C3 and C4 respectively) and *Level 1* (the less relevant level) has three constraints (C5, C6 and C7). The differences among levels follow a logarithmic function whose range limits are given in Figure 2.2 and also in the explanation below. As we have previously mentioned, we can distribute the constraints of the same level in two ways: (1) equally distributing weights (see Level 1 in Figure 2.2) or (2) assigning a higher weight for certain constraints (see Level 4 in Figure 2.2).

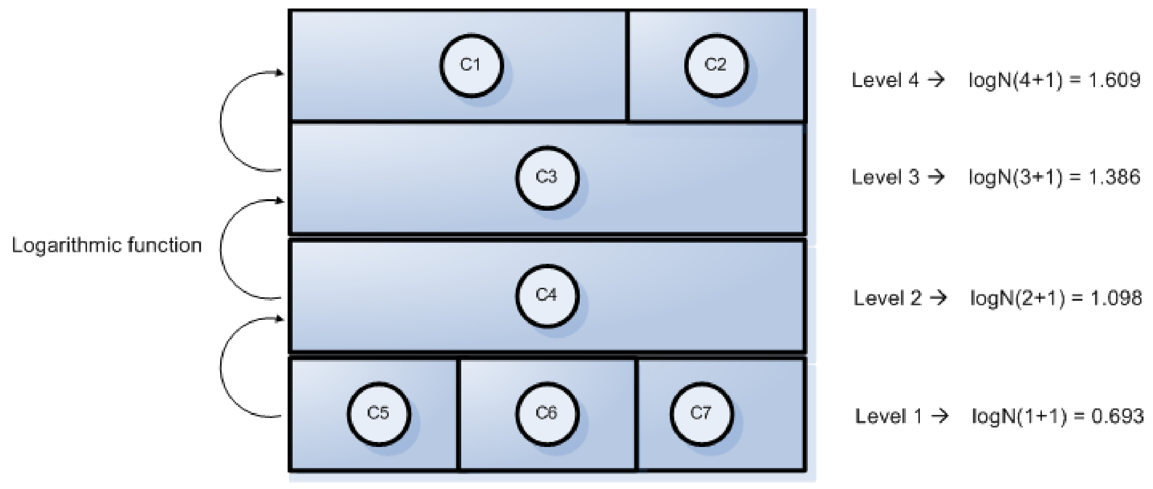
C1
C2
C3
C4
C5
C6
C7
Logarithmic function
Level 4 → logN(4+1) = 1.609
Level 3 → logN(3+1) = 1.386
Level 2 → logN(2+1) = 1.098
Level 1 → logN(1+1) = 0.693

Figure 2.2: Example of the relationship among constraints with *4* levels and *7* constraints.

Let us now explain in detail each level and its constraints. In our example, *Level 4* (the most relevant constraint level) has two constraints where *Constraint 1* (C_1) has a greater weight than *Constraint 2* (C_2). *Level 4*'s weights range from

$Ln(4) = 1.386$ to $Ln(4+1) = 1.609$, $(1.386, 1.609]$. Therefore, C_1 may have a weight of $1.550\epsilon(1.386, 1.609]$ while C_2 may have weight of $1.450\epsilon(1.386, 1.609]$ (these weights are fictitious, we just want to remark that C_1 has a greater weight than C_2 in Figure 2.2.

Level 3's weights range from $Ln(3) = 1.098$ to $Ln(3+1) = 1.386$, $(1.098, 1.386]$ and *Level 2*'s weights range from $Ln(2) = 0.693$ to $Ln(2+1) = 1.098$, $(0.693, 1.098]$. *Level 3* and *Level 2* have a unique constraint, which must belong to its respective constraint level's range. Of course, *Level 3* has more relevance than *Level 2* , which has more importance than *Level 1* at the same time.

Level 1's weights range from *0.1* (we will consider *0.1* as a minimum) to $Ln(1+1) = 0.693$, $(0.1, 0.693]$. In *Level 1*, all constraints have the same relevance as take the same "space" in the level (let's say *0.5*).

Note that if we need to set up higher differences among levels, we just need to assign a higher range of weights for each level but this is problem dependent (we have just shown an example).

Finally, we need to normalise all penalisations by dividing by the total sum of weights assigned to the soft constraints.

The aim is to find a solution to the problem whose cost, evaluated as the sum of the cost functions (penalisations of soft constraints), is minimised.

2.4 Particularisation of a Real-World DMAS: The Multi-Skill Call Centre

Let us now describe the real-world DMAS to be faced in this work: the *Call Centre* (CC). A CC is a centralised office where there are groups of agents, who are skilled to handle large volumes of customer phone calls [(22)]. Most large companies use CCs to assist with everything from information requests to the selling of products and/or services. Since CCs are usually the main point of contact between customers and companies, the service level provided to those clients must be high enough to assure extreme customer satisfaction. When a CC is committed to delivering exceptional customer experience, an effective workforce planning based on the needs of current and future inbound traffic is required as response delays irreversibly cause client frustration.

In a common CC, the stream of calls is often divided into *outbound* and *inbound* traffic. *Outgoing calls* are handled by agents, primarily, with commercial pretensions. This type of call is planned as agents know in advance which customers must be contacted on a daily basis. Conversely, *incoming calls* are made by customers to contract a service, ask for information or report a problem. These unplanned calls are initially modelled and thus classified into manifold *Call Groups* (CGs) in relation to the nature of each call (complaints, VIP clients, client loyalty, etc.). As soon as these calls have been modelled, each call is assigned to a unique CG (there is no overlap among CGs), following a “First-Come, First-Served” policy. The processing time of an incoming call can be broken down into *3* distinct parts as indicated below:

1. The time needed to assign a type to the call (*modelling*).

2. The time the call is queuing (*waiting*).

3. The time the agent needs to handle the call (*processing*).

A key component of any CC is the *automatic call distributor*, which is a system that models incoming calls and automatically distributes them over different queues from which certain agents can pull work. The routing scheme is a rule-based set of operations that enables the automatic call distributor to route a given incoming call within the system. Typically, once an incoming call has been assigned to a queue, a second algorithm is required to select the best available agent to reply to that call. The major problem with this second step is that conventional algorithms usually have no time to reach an optimal solution since CCs are extremely dynamic environments [(130)].

Habitually, the distribution of the incoming flow is carried out, considering the current state of the queues [(61)]. However, the challenge is to solve the uncertainties of next system state (by means of a forecast) in order to provide the algorithms with more time to efficiently reallocate the agents according to the predicted workload. The majority of traditional techniques is supported by a strong assumption that relies on the way that incoming traffic arrives. Most of the techniques suppose that incoming CC arrivals follow a Poissonian distribution. In this context, the main concern should be to forecast, for an upcoming state, the inbound traffic, drop calls rate and available agents having the required skills, in order to properly divvy up the workload among agents as our resources can be, at this point, optimised by a search algorithm. Remind that a fair allocation of workforce improves client satisfaction and, furthermore, reduces costs.

A specific type of CC, which is our case of study, is the *Multi-Skill Call Centre* (MSCC). In an MSCC, there are n incoming customer calls $C = \{c_1, c_2, ..., c_n\}$ grouped in k call groups $CG = \{cg_1, cg_2, ..., cg_k\}$ according to the call type, and m agents $A = \{a_1, a_2, ..., a_m\}$ that have a subset of all the possible skills ($S = \{s_1, s_2, ..., s_k\}$)

to attend the corresponding CGs (having the skill s_i enables you to attend the call group cg_i). Not all the agents have the same skill set and the number of skills per agent is different. Agents can only attend the CGs they have been trained for. This implies that each agent can attend different call types and, given a call type, it can be answered by several agents who have the associated skill. Note that agents cannot attend any kind of customer calls as they are usually specialised in concrete tasks (they do not have the complete skill set) or sometimes limited by law regulations. Although agents may have multiple skills, each agent can only process one call at the same time. Furthermore, given a call, it requires an unknown amount of time to be accomplished. Besides, each agent must orderly process each call during an uninterrupted period of time; in other words, the call cannot be divided or postponed once it has been started.

Obviously, the scenario can be simpler in some special CCs in which agents have a single skill. Those CCs can be modelled with q single queues working in parallel. In other cases, every agent has the whole skill set; hence all customers are queued in a single queue that can be handled by any agent. The system is noticeably easier to analyse in these two extreme cases. With all the agents having every skill, the system is also more efficient (shorter waiting times, fewer call drops) when the service time distribution for a given call type does not depend on the agent's skill set. However, this assumption turns out to be wrong in practice: agents are usually faster when they handle a smaller set of call types (even if their training or background provides them with more skills).

Agents with more skills are also more expensive as their salaries depend on their skill sets. Thus, for large volumes of call types, it makes sense to dedicate a number of single-skill agents (specialists) to handle most of the load. A small number of agents, proportional to the calls of each type, with two or more skills can cover potential fluctuations in the arriving load (this is one of the reasons for which we predict call arrivals). To address these fluctuations, the skills are grouped in skill profiles (subsets of skills) so that we can assign an agent to specific types of tasks during a given period of time, despite this agent has skills to process other types of work. These profiles are sets of related skills so that we can group similar skills (same relevance or close nature) without adding more dimensions. Of course, profiles may contain a unique skill if we need the maximum granularity for other skills.

Besides, the mean call arrival rate is not the same for each CG as well as the calls of these CGs require different processing times. Traditionally, the forecasting of incoming calls in CCs has been approximated according to a Poisson Distribution (PD) [(27)]. However, inbound traffic in CCs is usually not a stationary Poisson process [(105)] [(2)] and, the service times do not increase exponentially. Since calls randomly arrive according to a stochastic process, it would be desirable to have a well-balanced allocation of agents, who can be available or not, in order to attend the incoming calls as soon as possible (more reasons to forecast incoming calls).

Figure 2.3 illustrates the relationship among client calls, queues and agents. This figure describes an example for *9* client calls grouped in *4* CGs, *5* agents having different real skills and *7* different profiles.

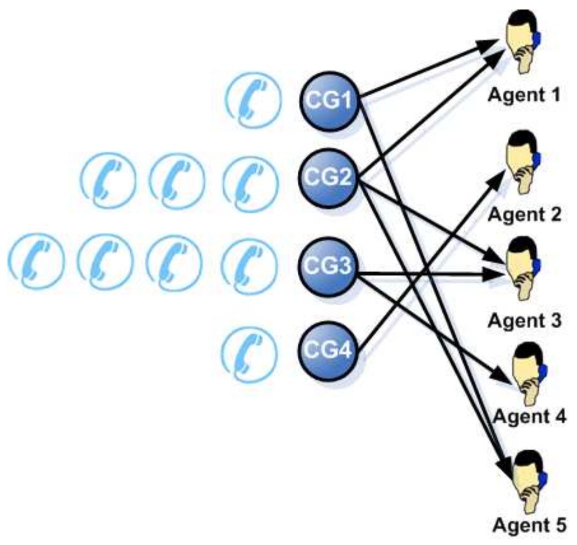

Figure 2.3: Inbound traffic scheme in MSCCs.

More formally speaking, the following parameters can be found in an MSCC:

1. Let C be a finite set of $n \epsilon \mathbb{N}$ customer calls: $C = \{c_1, c_2, ..., c_n\}$.
2. Let CG be a finite set of $k \epsilon \mathbb{N}$ call groups $CG = \{cg_1, cg_2, ..., cg_k\}$, where $k \leq n$ when every CG has, at least, one call queuing.
3. Let A be a finite set of $m \epsilon \mathbb{N}$ agents $A = \{a_1, a_2, ..., a_m\}$. Usually, $m >> k$.
4. Let S be a finite set of $k \epsilon \mathbb{N}$ agent-skills $S = \{s_1, s_2, ..., s_k\}$ in which each agent-skill, s_i, represents the capability to handle the corresponding CG, cg_i, with the equivalent sub-index in CG. $CG \sim S : s_1 \sim cg_1, s_2 \sim cg_2, ..., s_k \sim cg_k$.
5. Let P be a finite set of $d \epsilon \mathbb{N}$ agent-skill profiles $P = \{P_1, P_2, ..., P_d\}$ in which each agent-skill profile $P_i \subset S$ can be any subset of $S = \{s_1, s_2, ..., s_k\}$.
6. Let O be a finite set of $n \epsilon \mathbb{N}$ operations (execution or processing of each customer call, c_i) $O = \{o_1, o_2, ..., o_n\}$ in which each operation, o_i, has associated a processing time, which depends on its CG.

Moreover, the solution must fulfil the following descriptions:

1. every agent, a_i, has associated a finite non-null subset of P, A_P, containing its skills to handle different CGs (individual skill-profile): ($a_i \sim A_P \subset P$).

2. on O define R, a binary relation, which represents the precedence among operations. If $(o_i, o_j) \epsilon R$ then o_i has to be performed before o_j.

3. the same profile P_i can be assigned to several agents. In other words, several agents may have some skills in common (or even all of them): $(1...*)P_i \sim (1...*)a_j$.

4. although every agent, a_i, may have several profiles assigned, only one can be used at a given instant t, $< a_i, P_j >_t$.

The solution to the problem of the workforce distribution in MSCCs is defined as the right assignment, by means of the maximisation of a quality metric Q_M (problem dependent), between every agent a_i and the required profile P_j for each time-frame considered (v seconds): $Max[Q_M(\Sigma a_i, \Sigma P_j, t)]$. In addition, the assignment $< a_i, P_j >_t$ must satisfy all hard constraints and handle the soft ones given by the business units.

To determine whether (or not) a given solution is suitable, we need to define a quality metric to evaluate the rightness of each feasible solution. There are very significant metrics to measure the quality of a CC such as the drop call and service rates. These metrics somehow hinge on the (customer) *service level* [(106)], which is defined as the percentage of customer calls that have to queue shorter than a specified amount of time. Our work has been conducted by applying this metric.

To wrap up this section, it is important to highlight that an initial step to produce a workforce planning is to predict future system loads. Therefore, the challenge is to forecast the number of incoming calls that will be in each queue on that prospective system state. Bear in mind that for each forthcoming system state, we have to determine the total number of agents having certain skills and the total number of calls queued in each queue at $t + v$:

$$IC(t + v) = QC(t) + AC(t, t + v) - PC(t, t + v) - CD(t, t + v) \tag{2.1}$$

where

- v is the size of the prediction window,
- $IC(t + v)$ is the predicted number of Incoming Calls at time $t + v$,
- $QC(t)$ is the number of Queuing Calls at time t,
- $AC(t, t + v)$ represents the number of Call Arrivals during the interval $(t, t + v)$,
- $PC(t, t + v)$ is the number of Processed Calls during the interval $(t, t + v)$,
- $CD(t, t + v)$ is the number of Call Drops during the interval $(t, t + v)$.

As the number of available agents mainly depends on the agent timetable, the number of queuing calls is well-known, the number of processed calls between t and $t + 1$ depends upon the mean processing time of each CG and the number of call drops is proportional to the number of incoming calls, we will mainly focus on the prediction of call arrivals and will sum up/subtract the other terms of the equation.

The main objective of this real-world problem is to get, for each time-frame $(t+v)$, an automatic allocation of agents and call groups ($\{a_i, cg_j\}_t$ when a_i is related to s_j) that maximises the service level (see [(130)]). It stands to reason that we want to assign a larger number of agents to those CGs with a greater volume of traffic based on their priority or relevance. As a general rule, we should find the situation that respects the desired service levels for each CG (at least the most important ones). Having a much smaller number of available agents than incoming calls, it will be necessary to carry out an adequate redistribution of the agents at any time.

The problem of workforce distribution in MSCCs is a very complex, dynamic real-world problem. Usually, the number of incoming calls (n) is much larger than the number of agents (m) and the flow of calls is very dynamic over time, making this problem really hard. Intuitively, this problem is much more complicated than having a simple pool of incoming calls where agents take work from, since it requires the assignment of customer incoming calls to the agents having the right skills, satisfying a given set of additional constraints and respecting the dependencies among individual tasks and differences in the execution skills of the agents (see [(130)]). This problem is somehow related to other classic changing scenarios where staffing requirements are identified to insure that the organisation has the right number of agents at the right time. This is a highly difficult problem because we are not only dealing with an NP-hard problem like the job assignment problem ([(28)]), but the problem also considers rapidly varying conditions, massive incoming calls and a large number of agents having hard constraints to process certain tasks.

All these premises make us conclude that we need to employ non-deterministic techniques like evolutionary algorithms. In many scenarios, evolutionary algorithms have proved that they can solve very complex real-world and theoretical problems and frequently perform well approximating solutions to all types of problems because they do not make any assumption about the underlying fitness landscape.

Chapter 3

Literature Review

Throughout the present thesis, we will apply diverse fields from *artificial intelligence* since this work is founded on predictive and optimisation algorithms: artificial neural networks for the forecasting module, time series for the dynamism mechanism and parallel memetic algorithms for the optimisation module. Therefore, apart from reviewing workforce distribution techniques for DMAS, we will present a survey of forecasting and optimisation techniques as they constitute the basis of our novel approach.

In Section 3.1, we will present some key workforce distribution techniques for DMAS (in the first subsection) with special focus on the call centre (in the second one). Section 3.2 provides a study of related work on forecasting from diverse angles, considering commonalities with other problem domains, including the call centre. Section 3.3 reviews the most representative combinatorial optimisation techniques to the best of our knowledge. These techniques are not specifically conceived for the call centre, so that we will discuss whether (or not) they can be employed. At the end of each section, we present a brief discussion, highlighting current limitations and some improvement points.

3.1 Workforce Distribution Algorithms for DMAS

3.1.1 Generic Algorithms

There is a large number of workforce distribution techniques for conventional DMAS. A very common technique is the *Random Workload Balancing* (RWB) [(79)]. RWB purely assigns a random profile to each agent (among the available ones for that agent). In RWB, the neighbourhood covers the whole search space. After multiple iterations, the best solution found is chosen. Supposing there are s possible solutions, the probability of finding the global optimum is $1/s$ for each execution. This technique can be appropriate whether there is little communication overhead and numerous agents are available. As the number of agents decreases, the workload of the busiest agents increases in relation to the average agent workload, resulting in poor parallel efficiency. Since each task is assigned to an agent by selecting a random destination, RWB only needs to execute a single pass through the tasks list.

Other common approach is the *Greedy Workload Balancing* (GWB) [(34)]. GWB reallocates agents without considering the current assignment *task type-agent* (note that other techniques start out from a neighbour solution but GWB does not). An agent heap is built with the intention that the agent with the least assigned workload is on the top of that heap. At the beginning, no tasks are assigned to any agents, hence every agent in the heap has no workload, and the agent on the top of the heap is randomly chosen. A task heap is also built and organised, so that the most time-consuming task is on the top of the heap. For each agent, the most time-consuming unassigned task is allocated to the less loaded agent with the capability of handling that type of task. Afterwards, the agent's workload is updated and both heaps are readjusted. This process is carried out until every task has been assigned to an agent with the required skills.

Other authors have as well put forward different approaches. For instance, Baker (1983) [(17)] stated that a task flow allowance was a fixed amount for given task data and did not depend upon the status of the system when the task arrived. Then, Vaughan (1985) [(171)] provided a literature survey, describing various load balancing techniques using a tree structured classification. Eager et al. (1986) [(52)] concluded that simple workload distribution policies offered the greatest promise in practice in homogeneous MAS, because of their combination of nearly optimal performance and inherent stability. Afterwards, Shenker and Weinrib (1989) [(158)] investigated the performance of load-sharing and routing algorithms, proposing new heuristic policies (like the shortest-expected delay policy) that performed well over a wide range of system parameters. In 1990, Kuchen and Wagner (1990) [(109)] described several load balancing techniques that depended only on local knowledge. Their techniques were divided into three types: active, passive and mixed. Baker and Milner (1991) [(18)] used a load balancing strategy in process migration experiments. Applications were programmed, using a special harness. Then, Cutkosky et al. (1996) [(42)] provided rapid responsive and dynamic reconfigurable structures in order to facilitate flexible and efficient use of manufacturing resources in a dynamic environment. In 1997, Trehel et al. (1997) [(169)] presented an original model of dynamic workload balancing inside groups of agents, using queueing theory. Later, a dynamic load balancing scheme for structured adaptive mesh refinement applications was proposed in Lan et al. (2001) [(111)]. They presented two improvements to reduce overhead. Then a detailed sensitivity analysis was provided to identify an optimal value for the parameter threshold. Liu et al. (2005) [(119)] presented a macroscopic characterisation of agent-based workload balancing in homogeneous environments. The agent-based workload balancing was regarded as agent distribution from a macroscopic point of view. Finally, Dhakal et al. (2007) [(48)] defined an approach with a certain special random variable, called *regeneration time*, defined as the time to the first completion of a task by any agent or the first arrival of a communication, whichever came first.

3.1.2 Specific Algorithms for the Call Centre

Reviewing the state-of-the-art on CC algorithms, one can realise that many algorithms for workforce distribution in single-skill CCs are available (e.g. [(178)]) because, in the past, agents were commonly allocated to single customer call groups. Nevertheless, few works have been conducted to workforce distribution in MSCCs, which is the commonest scenario in nowadays CCs.

In the past, the first call-distribution techniques followed a very basic scheme, assigning the next incoming call to be processed (from a pool of calls) to the next available agent (from a pool of agents) who had the required skill. Afterwards, workforce distribution in MSCCs had broadly been faced by a *Skill-Based Routing* algorithm (SBR) [(64)]. SBR was a call-assignment strategy used in CCs to assign incoming customer calls to the most suitable agent, instead of simply choosing next existing agent. The need for SBR arose as CCs became larger and had to deal with a wider variety of call types. Habitually, the routing strategy was led by a simple heuristic (e.g. efficient driven SBR) as SBR claimed for quick movements rather than convoluted, time-consuming formulas. SBR usually relied on the Erlang-C formula [(126)], which had extensively been applied to the CC domain. Nevertheless, some researches [(91)] [(123)] claimed that the conventional Erlang-C formula was no longer applicable to settling on staff schedules as they were frequently inexact. The major handicap of this approach was that online (ad-hoc) routing heuristics could not be very complex in view of the fact that a very short response time was required. These fast, unplanned decisions could imply suboptimal task types assignments to existing agents.

Conversely, Thompson [(166)] proposed an integer programming model, which differentiated minimum acceptable service levels per time-frame from a constraint on the mean service level over the planning horizon. Notwithstanding this approach considered prospective situations, it was less dynamic to changes than SBR.

Other approaches considered dependent planning intervals (e.g. [(92)]). Most methods performed well enough within separate intervals but their performance decreased when moving to the next one, giving much trouble in prospective time-frames.

Other authors took into consideration overflow routing in multi-skill blocking systems with randomisation parameters by applying a branch-and-bound algorithm (e.g. [(112)]) or cutting planes (e.g. [(72)]). These techniques were only appropriate for stable environments because they needed long response times and their performance highly decreased in large instances.

One of the most representative algorithms of the state-of-the-art is the one proposed in Koole et al., 2008 ([(106)]). Koole presented a heuristic, which considered the costs of agents and a service-level condition, to optimise the distribution of agents among different CGs. This algorithm is faster than most of the aforementioned approaches but deals with specific types of MSCCs in which customer calls arrive according to a Poisson process with deterministic rate. However, note that inbound flow in MSCCs is not usually a stationary Poisson process [(105)] [(2)] and the service times do not increase exponentially. Since calls arrive randomly according to a stochastic process, agents must be well-distributed to handle the calls as soon as possible. Besides, the previous techniques often consider a high granularity and need to work at agent groups' level instead of an agent's profile level. This setback does not enable us to offer more accurate solutions.

3.1.3 Discussion

A quarter-century of blackboard-system experience and more than a decade of MAS development have produced a solid baseline of collaborating techniques. Truly, there is a number of workforce (agents are assigned to task types) and workload (tasks are routed to the agents) techniques for DMAS.

The power of any MAS can be realised by allowing its constituent agents to work cooperatively so that large workloads are distributed among them in a fair and effective manner. Any strategy for workload distribution among agents is also called workload balancing. An effective workload policy ensures optimal use of the distributed agents, whereby no agent remains in an idle state, while any other agent is overworking. Although the majority of workload distribution policies developed heretofore took into account such time delays, they were predicated on the assumption that delays are deterministic.

Besides, there is a commonality in all the reviewed algorithms, as they are all supported by basic optimisation heuristics in order to provide fast responses. Therefore, the predictive component of these approaches is very basic because it is intended for short-time predictions.

3.2 Forecasting in DMAS

Most people perceive the world as a place where there is a large number of alternatives. In this context, forecasting refers to the estimate of attributes under unknown situations to help decision, making and strategic planning. A significant but ignored aspect of forecasting is the close relationship it holds with strategic planning. Forecasting can be expressed as predicting how future will resemble, whereas strategic planning enlightens how future should look like. There is no universal, most suitable forecast method to use, as it depends on our objectives and preconditions. In this section, we will review some forecasting techniques for both general DMAS and call centres.

3.2.1 Generic Forecasting Techniques for DMAS

Some authors have addressed similar problems by means of **Regression Models** (RM) [(118)], [(135)], [(77)], [(102)]. An RM is a statistical method in which an unknown variable is predicted according to its relation with the rest of well-known variables (also named as predictors), using a formula called *regression equation* [(118)], [(108)], [(107)]. This equation deals with some constant parameters, which must be optimised to reduce the *Mean Square Error* (MSE) between the predicted output and its real value. These model types are sophisticated and make possible almost infinite variety in analysis, and can thus be complex and difficult to master.

We can find different variants of RM like the *Linear Regression* (LR), which is one of the commonest variants. LR fits all parameters by applying diverse policies. The commonest policies are the following ones: least squares approach, minimisation of the "lack of fit" and minimisation of least squares loss function as ridge regression assumes. Although not identical, least squares and linear model are intimately related to each other [(118)]. LR approximates the unknown variable with a straight line by using well-known variables as follows:

$$Y_i = \beta_0 + \sum \beta_p X_{ip} + \varepsilon_i \tag{3.1}$$

where parameter i is the pattern-position in the dataset, p indicates the n-th well-known variable, β_P represents the associated parameters to the n-th well-known variable, β_0 is a constant parameter, Y refers to a dependent variable and ε denotes the associated error. β_P and β_0 are calculated in order to reduce $\Sigma\varepsilon_i$, using predefined patterns.

Another important variant is the *Logistic Regression* (LogR) (Nemes, 2009) [(138)], which is used for predicting the occurrence probability of an event by fitting data to a logistic function. It is indeed a generalised linear model used for binomial regression. Like many forms of regression analysis, it makes use of several predictor variables that can be either numerical or categorical [(138)]. LogR measures the relationship between a categorical dependent variable and usually a continuous independent variable (or several), by converting the dependent variable to probability scores. The regression coefficients are usually estimated using maximum likelihood estimate. Unlike LR with normally distributed residuals, it is not possible to find a closed-form expression for the coefficient values that maximises the likelihood function, so an iterative process must be applied instead (i.e. Newton's method). This process begins with a tentative solution, revises it slightly to see whether it can be improved, and repeats this revision until improvement is minimal, at which point the process is said to have converged. The model may not converge in some instances. When a model does not converge, this indicates that the coefficients are not meaningful because the iterative process was unable to find appropriate solutions. Like many forms of regression analysis, it makes use of several predictor variables that can be either numerical or categorical. LogR has also been applied to intra-day forecasting successfully.

Much of the previous work on call arrivals forecasting is focused on the application of standard **Time Series** (TS) (Wei, 1989) [(173)], such as *Autoregressive Integrated Moving Average* or *Exponential Smoothing* like *Single Exponential Smoothing*, *Damped Trend Time Series* and *Stationary Time Series* (Garcia, 2009) [(62)].

A TS is a sequence of observed variables, taken in regular time-slices. This sequence is used for understanding and forecasting the behaviour of a given variable over time based on previous states [(173)]. A TS applies a (more or less complex) regression over the n-previous variables to estimate forthcoming values. This way, TS can be divided into two major groups: 1) *Exponential Smoothing* (ES) and 2) *Autoregressive Integrated Moving Average* (ARIMA).

At the same time, ES methods, which assign decreasing weights to each previous observation, are split into: *Single Exponential Smoothing* (SES), *Damped Trend Time Series* (DTTS) and *Stationary Time Series* (STS) [(62)]:

1) *SES* [(184)] is a method for forecasting whether the mean is stationary or slowly changes over time. The name is frankly ambiguous, given that this is a moving average method in which weights decline as the interval between the current time increases. The smoothed value lags the current value as long as this method depends upon previous values. When the smoothing value is small, the oscillations are seriously damped and the smoothed value tends in the direction of the mean. Nevertheless, when the smoothing value is large, the oscillations noticeably fluctuate and, as a result,

the smoothed value tends to the current value. SES can be obtained as follows:

$$\begin{aligned} S_t &= \alpha y_{t-1} + (1-\alpha)S_{t-1}; \\ 0 &< \alpha \leq 1, t \geq 3; \\ F_{t+1} &= \alpha y_t + (1-\alpha)S_t \end{aligned} \tag{3.2}$$

where S stands for the smoothed observation, t refers to an index, which denotes a time-period, α is a constant, which must be estimated with the purpose of minimising the MSE, y is the observation and F is the forecast.

2) Instead, *DTTS* (also known as *Double Exponential Smoothing*) [(25)] extends exponential smoothing by incorporating a term for linear trends. DTTS is a suitable technique to deploy when data show trend and seasonality [(122)]. This technique introduces a third equation to cope with seasonality. Suppose that at time t, y_t is observed, the level L_t is estimated and the slope b_t is known in the series. Afterwards, a k-step ahead forecast is $F_{t+k} = L_t + b_{tk}$. DTTS allows us to adjust the slope with each new observation. DTTS can be formalised as:

$$\begin{aligned} S_t &= \alpha y_{t-1} + (1-\alpha)(S_{t-1} + b_{t-1}); 0 \leq \alpha \leq 1; \\ b_t &= \gamma(S_t - S_{t-1}) + (1-\alpha)b_{t-1}; 0 \leq \gamma \leq 1; \\ F_{t+m} &= S_t + mb_t; \end{aligned} \tag{3.3}$$

3) A *STS* is one whose statistical properties (e.g. mean, variance, autocorrelation, etc.) are all constant over time. Most statistical forecasting methods are based on the assumption that the TS can be rendered approximately stationary through the use of mathematical transformations. A stationarised series is relatively trivial to predict: you simply predict that its statistical properties will be the same in the future as they have been in the past. The predictions for the stationarised series can then be untransformed, by reversing whatever mathematical transformations were previously

used, to obtain predictions for the original series. Therefore, finding the sequence of transformations needed to stationarise a TS often provides important clues in the search for an appropriate forecasting model.

STS is a suitable technique to deploy when data show trend and seasonality. This technique introduces a third equation to cope with seasonality. STS can be formulated as follows:

$$\begin{aligned} S_t = \alpha \frac{y_t}{I_{t-L}} + (1-\alpha)(S_{t-1} + b_{t-1}); 0 \leq \alpha \leq 1; \\ b_t = \gamma(S_t - S_{t-1}) + (1-\gamma)b_{t-1}; 0 \leq \gamma \leq 1; \\ I_t = \beta \frac{y_t}{S_t} + (1-\beta)I_{t-L}; \\ F_{t+m} = (S_t + mb_t)I_{t-L+m}; \end{aligned} \tag{3.4}$$

where y is the observation, S stands for the smoothed observation, b is the trend factor, I indicates the seasonal index, F denotes the forecast at m periods ahead, t refers to an index that denotes a time period and α, β and γ are constants, which must be estimated with the purpose of minimising the MSE.

Differently to smoothing approaches, *ARIMA* [(122)] is determined by three parameters (p, d, q), where p is the autoregressive term, d is the number of previous values and q is the average moving parameter. *ARIMA (p, d, q)* can be calculated for a TS sequence $Y_t(t = 1, 2, \ldots, n)$, as follows:

$$\begin{aligned} \phi(B)(1-B)^d Y_t = \theta(B)Z_t; \\ where \phi(B) = (1 - \alpha_1 B^1 - \alpha_2 B^2 - \ldots - \alpha_p B^p); \\ and \theta(B) = (1 - \beta_1 B^1 - \beta_2 B^2 - \ldots - \beta_q B^q); \end{aligned} \tag{3.5}$$

Z_t is a white noise sequence and B is the backshift operator.

ARIMA $(p, q, d) \cdot (P, D, Q)$ represents a multiplication of two ARIMAs to inject seasonality into the model. This method requires that new seasonal and non-seasonal parameters are estimated; analogously to simple ARIMA. The involved parameters are the following ones: p is the autoregressive order, which indicates the number of parameters of ϕ, d is the number of times that data series must be distinguished to induce a stationary series, q is the moving average order that designates the number of parameters of θ, P is the seasonal autoregressive order that specifies the amount of parameters of ϕ, D is the seasonal moving average order, which points out the quantity of parameters of θ, and Q is the number of times that a data series needs to be differenced to induce a seasonal stationary series.

Finally, the problem of predicting task arrivals can be faced by an **Artificial Neural Network** (ANN), which is a mathematical model founded on the functioning of biological neural networks [(124)]. In this manner, an artificial neuron is a computational model inspired in biological neurons and also the simplest processing element of an ANN. Natural neurons receive signals through synapses placed on the dendrites. When the arriving signals surpass a certain threshold, the neuron is activated and emits another signal through the axon. This signal can be sent to another synapse to activate other neurons.

In order to emulate biological neurons, the artificial ones (see Figure 3.1) are organised into two units: the first one is a non-linear weighted sum of weight coefficients and input signals, *F(x)*, whilst the second one follows a non-linear function, widely known as *neuron activation function*, *K*. The function *F(X)* accumulates weights w_i and maps results to an output as given below:

$$F(x) = K(\Sigma w_i input_i) \tag{3.6}$$

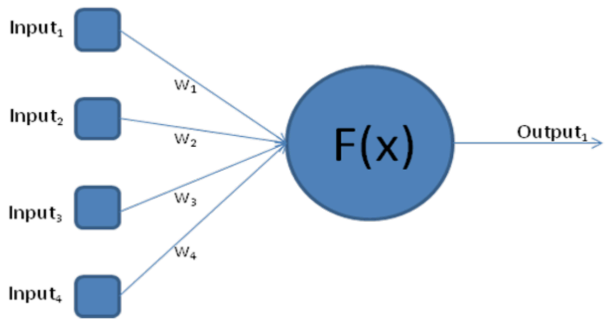

Figure 3.1: Scheme of a basic artificial neuron.

The weights, w_i, are randomly initialised and then updated during the training process.

There are numerous functions to approximate K, but the most widespread ones include the Gaussian function, the hyperbolic function and the *sigmoid function*. We will employ the sigmoid function as this is the most appropriate one for our dynamic environment. The sigmoid function and its derivative are defined as indicated below:

$$\begin{aligned} \sigma(x) &= \frac{1}{1+e^{-x}} \\ \delta\sigma(x)/\delta x &= \sigma(x)(1-\sigma(x)) \end{aligned} \tag{3.7}$$

Figure 3.2 plots the sigmoid function to facilitate reader's understanding.

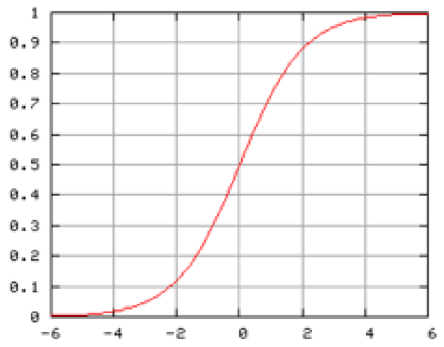

Figure 3.2: Sigmoid function.

As a remark, the sigmoid function can never return "*0*" or "*1*" due to its asymptotic nature so that values over *0.9* should be treated as *1* and those under *0.1* should be considered as *0*.

Neurons can be grouped into three types of layers: *input*, *hidden* and *output*. The input layer is composed by neurons that represent the data input variables and "feed" next layers of neurons. Next layers, which are sometimes optional, are denominated hidden layers and there may be several of them. The last layer is called output layer, in which each neuron represents an output variable. Each layer is fully connected to the succeeding layer as Figure 3.3 illustrates.

For linearly separable problems, a sole neuron can categorise the output, but when having more than one class or multimodal spaces at least one hidden layer is needed [(24)].

Many authors apply regression methods in which data is best-fitted to a specified relationship that is usually linear. However, these methods have several handicaps. For instance, relationships must be chosen in advance and these must be distinguished as linear or non-linear when defining the equation. ANNs enable us to overcome all these problems.

In regression, the objective is to forecast the value for a continuous variable, which is the incoming flow rate in our case. The output required is a single numeric variable that has been normalised between *0* and *1*. ANNs can actually perform a number of regression tasks at once, although each network commonly performs only one.

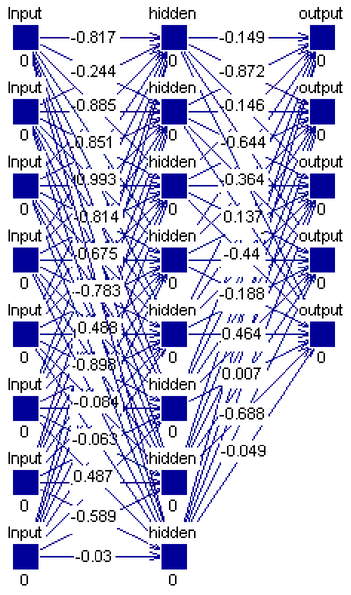

Figure 3.3: Example of a simple ANN with *8* input neurons, *8* hidden neurons and *5* output neurons, forming *3* fully connected layers.

There exists a number of learning algorithms for training ANNs. Most of them can be viewed as a clear-cut application of optimisation theory and statistical estimation. This way, we can find learning algorithms such as back-propagation by gradient descent [(129)], back-propagation with momentum [(146)], resilient propagation [(148)], quick-propagation [(9)], Broyden-Fletcher-Goldfarb-Shanno [(121)], radial basis function [(29)], cascade correlation [(56)], Hopfield [(85)], etc.

3.2.2 Call Centre Forecasting

There are numerous works on forecasting although very few of them address the problem of predicting call arrivals in CCs. But, what does forecasting stand for in a CC domain? Forecasting consists of the estimate of values at certain specific future points in time.

Forecasting is the pivotal point in CCs'operations where efficiency, effectiveness and service quality intersect. In this manner, there are many things that would be worth predicting in a common CC, such as call arrival rates, call drop rates, available agents having a certain skill, speaking levels (this is the time the agent is truly processing calls), service rates (given by a quality metric) and mean delay times, although we will just focus on call arrival forecasting.

Why is forecasting necessary in a typical CC? Since CCs are usually the main point of contact between customers and companies, the service level provided to those clients must be high enough to assure extreme customer satisfaction. When a CC is committed to delivering exceptional customer experience, an effective workforce planning based on the needs of current and future inbound traffic is required as incessant response delays irreversibly cause client frustration. Where CC forecasting is repeatedly inaccurate, it is most likely that the CC will very shortly cease to exist. Even though CCs have been broadly studied, there is still some lack of optimisation which sparks a wrong allocation between agents and tasks. This inaccurate allocation of resources may cause huge losses of money every year and client dissatisfaction due to the derived never-ending delays.

We can distinguish two types of call arrivals forecasts in a common CC: (a) inter-day forecasts for several days or weeks ahead; and (b) intra-day forecasts to dynamically update the prediction using newly available information as one of the biggest forecasting challenges is related to call volume fluctuations. In general, call arrival data is grouped in delimited time periods which may range from *15*- to *30*-minute intervals and the target of forecasting is future call volumes over such periods (Jongbloed and Koole, 2001 [(96)]).

On the one hand, underestimating incoming call volume involves unacceptable call drop rates. A large number of call drops indicates that the service level is not met and, consequently, the client is most likely to be unhappy. Obviously, customers with excessive queuing times get cranky. On the other hand, overestimating inbound traffic has a direct impact on the bottom line. Less than fully occupied, customer service agents feel demotivated, get bored and lose interest. Service level goals may effortlessly be attained but idle staff sends the cost per call through the roof.

Particularly, a precise prediction enables us to be prepared for what future may have in store for us, so that we can correctly balance the workload among agents, presenting higher service levels and, eventually, optimising our resources. To achieve this, we can compile arriving tasks, task failures and queuing tasks in a unique value, which designates the total number of pending tasks to handle (grouped by task types).

Unfortunately, there is no accurate mechanism to foresee what future will bring along with complete sureness. *Risk* (wrong workload predictions generally entail losses of money or even major hazards) and *uncertainty* (ambiguity or indecision to reach our predefined goals) are omnipresent in forecasting to the degree that it is customarily considered appropriate practice to specify the level of uncertainty associated to each prediction.

Existing techniques usually distribute workload according to daily, weekly or even monthly forecasts [(130)], so that specific anomalous peaks cannot be forecast. As an example, many Spanish CC professionals may remember the chaotic morning of May 20^{th}, 2010 when some building works in Valence caused an Internet cut-off in Castellón and its surrounding area. This obviously entailed an important peak of incoming calls - from people asking for information - which could not probably be predicted. This peak of calls involved an unusual low service level and many people ported out to other competitor broadband providers due to the poor service offered on that day. To mitigate this drawback, any CC should inject both real-time and historical information into its predictive models so that workforce plannings meet their real needs to achieve the desired service levels.

The prediction of call-centre arrivals has traditionally been addressed by means of a Poisson Distribution (PD) [(27)]. PD, which was first introduced by Siméon Denis Poisson, expresses the probability of a number of events occurring within a defined time-interval when those events are independent of the previous one and take place with a known rate. Under these conditions, PD is a reasonable approximation of the exact binomial distribution of events. Additionally, PD provides a useful mechanism to assessing the percentage of time when a given range of results is expected.

In the calculation of the distribution function, the values for the mean and standard deviation are given by the binomial distribution. Thus, "conventional" approaches assume that the number of incoming calls at any given point in time follows a PD. For this reason, pure-chance traffic is also named as Poisson traffic. Assuming pure-chance arrivals and pure-chance terminations leads to the following probability distribution:

$$P(n) = \frac{\lambda^n}{n!} \cdot e^{-\lambda} \tag{3.8}$$

where n denotes the number of arriving tasks in an interval of duration d, λ stands for the mean of arriving tasks at time t and e refers to the base of the natural logarithm ($e \simeq 2.7183$). The parameter μ does not only correspond to the mean number of event occurrences, but also its variance. This way, the number of observed occurrences fluctuates around its mean λ with a standard deviation $\sigma_k = \sqrt{\lambda}$. These fluctuations are designated as Poisson noise.

The correlation of the mean and standard deviation in counting independent discrete occurrences is helpful to for instance estimate the contribution of a single occurrence even when the contribution is very small.

Table 3.1 shows the values returned by a PD when varying λ between *0.1* and

n / λ	*0.1*	*0.5*	*1*	*1.5*
0	0.905	0.607	0.368	0.223
1	0.090	0.303	0.368	0.335
2	0.005	0.076	0.184	0.251
3	0.000	0.013	0.061	0.126
4	0.000	0.002	0.015	0.047
5	0.000	0.000	0.003	0.014
6	0.000	0.000	0.001	0.004
7	0.000	0.000	0.000	0.001
8	0.000	0.000	0.000	0.000
9	0.000	0.000	0.000	0.000
10	0.000	0.000	0.000	0.000

Table 3.1: Poisson distribution by varying λ between *0.1* and *1.5*.

1.5.

In the same way, Figure 3.4 plots the points of Table 3.1 to better understand PD's nature.

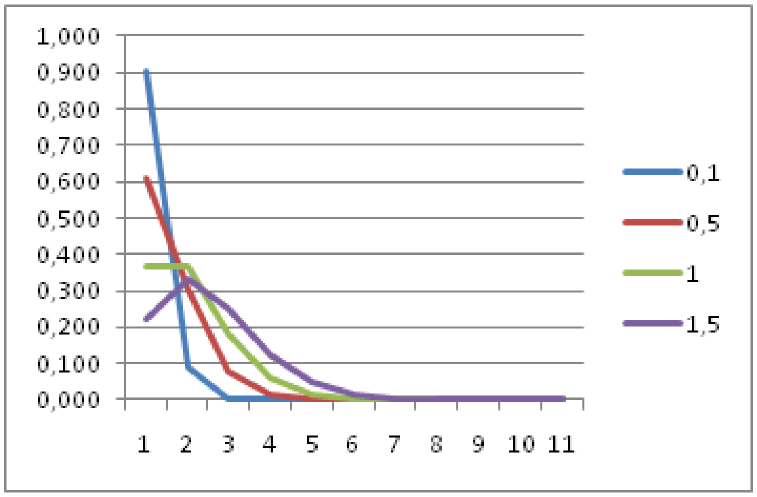

Figure 3.4: Poisson distribution by varying λ.

PD has inspired diverse authors to extend its philosophy to other problem domains. Reviewing the existing literature, we can bump into numerous algorithms founded on (or at least supported somehow by) a PD. As an example, we should highlight Erlang-based algorithms.

The Erlang distribution, first pioneered by A. K. Erlang [(126)], is a continuous probability distribution with extensive applicability. This distribution, which has a positive value for every real number greater than zero, is given by two terms: the shape k (a non-negative integer) and the rate l (a non-negative real number). The distribution is sometimes defined using the inverse of l, which is denoted as the scale m. Erlang distribution was conceived as a mechanism to inspect the number of calls arriving simultaneously to the agents of any MSCC. This work, which was originally designed for the CC domain, has afterwards been applied to other queuing environments by other authors [(57)], [(51)]. In the Erlang distribution, events are modelled in accordance with a Poisson process and independently take place with certain average rate. The waiting times between k event occurrences are Erlang distributed.

Figure 3.5 plots Erlang distribution for k = *(2, 3)* and λ = *(3, 1)*.

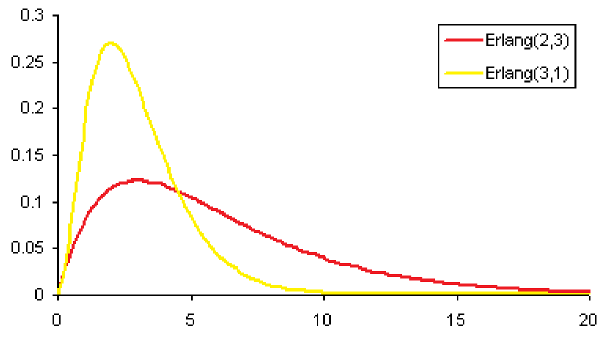

Figure 3.5: Erlang distribution for k = *(2, 3)* and λ = *(3, 1)*.

However, the prediction of inbound traffic in an MSCC cannot sometimes be adjusted to a PD with deterministic rate. In all studies (e.g. [(60)] and [(105)]), the arrival scheme agrees with a Poisson process only if the arrival rate of the Poisson process is itself a stochastic process.

Characteristically, the variance of calls arriving within a given interval is much larger than the mean. However, PD's hypothesis states it should be equal to the mean for PDs. The mean arrival rate also strongly depends upon the day-time and often on the week-day, but Poisson processes comply with the memoryless property of the exponential distribution [(60)], which is unable to detect this kind of features. Besides, in some MSCC's like ours, there is positive stochastic dependence between arrival rates in successive periods within a day and arrival volumes during successive days. Taking into account all these premises, we can realise how pertinent it is to find a more effective method to forecast incoming calls, which does not merely rely on the hypothesis of a simple PD.

There are other numerous works on forecasting although very few of them address the problem of predicting call arrivals in call centres. A very good compilation of forecast techniques for call centres was provided by Van den Bergh (2006) [(20)]. His work mainly focused on the forecast methods that were expounded and utilised for predicting call arrivals in call centres on a daily basis. Van den Bergh (2006) [(20)] gathered a wide selection of forecast models which ranged from ARIMA models and regression models to a variety of time series.

Like in other DMAS, some authors have addressed similar problems by means of Regression Models (RM). For instance, Klungle (1998) [(102)] used a Dynamic RM to forecast the number of incoming calls for the Emergency Road Service. An important finding was that the number of call arrivals noticeably varied at different times of the day during winter and spring seasons. In this work, it is claimed that RM performed better than the Holt-Winter's method and other basic neural networks. Ibrahim et al. (2012) [(88)] also considered a simple linear regression model with independent residuals to predict CC arrivals. The model was equivalent to a historical average approach as it basically used past averages as forecasts of future call volumes which was also related to the works of Weinberg (2007) [(174)] and Shen and Huang (2008)

[(157)]. Weinberg et al. (2007) [(174)] developed a Bayesian model for forecasting call centre arrivals that specifically assumed different weekday has its own smooth arrival rate profile. They extended the work of Avramidis et al. (2004) [(14)] and Brown et al.(2002) [(27)] to model both inter-day and intra-day dependences. Shen and Huang (2008) [(157)] proposed a dynamic factor model for *15*-minute call arrivals to a bank call centre. They applied univariate time series forecasting techniques to forecast them separately. Logistic Regression (LogR) (Nemes, 2009) [(138)] has also been applied to intra-day forecasting successfully.

Much of previous work on call arrivals forecasting is focused on the application of standard Time Series methods (TS) (Wei, 1989) [(173)], such as Autoregressive Integrated Moving Average (ARIMA) or Exponential Smoothing (ES) like Single Exponential Smoothing (SES), Damped Trend Time Series (DTTS) and Stationary Time Series (STS) (Garcia, 2009) [(62)]. This way, Andrews (1995) [(5)] used the ARIMA model to predict arrivals to L. L. Bean's call centre, and highlighted the impact of holidays and marketing campaigns on the arrival process. Instead, Bianchi (1998) [(23)] also used ARIMA models and found that they outperformed simple Holt-Winters smoothing. Another look at the literature reveals ARIMA and exponential smoothing forecasting methods specifically designed for intra-day CC arrivals data like Taylor (2012) [(164)] or Tych (2002) [(170)], where the authors proposed a new seasonal component model, based on modulated periodic components, that was capable of replicating multiplicative periodic components in an efficient manner.

Other authors have addressed related problems. Thus, Thompson (1996) [(165)] investigated different options for adjusting intra-day schedules in the context of hospitality workforce management. Hur et al. (2004) [(87)] proposed schedule updating techniques in quick service restaurants. Easton et al. (2005) [(53)] faced absence recovery problems for service operations.

Finally, the problem of predicting call arrivals can be faced by an Artificial Neural Network (ANN). Pacheco et al. (2009) [(144)] proposed an Improved Backpropagation Neural Network which was compared to some basic learning algorithms for neural networks (classic Backpropagation and Backpropagation with Momentum) as well as some of the existing forecast methods covered by Van den Bergh (2006) [(20)]. The authors claimed to slightly outperform the accuracy of the existing forecasting techniques although these apparently tiny differences implied huge cost savings at the end of the day. Afterwards, Millán et al. (2010) [(131)] provided a more exhaustive comparison of the Improved Backpropagation Neural Network with the forecasting techniques given by Van den Bergh (2006). Other authors like Darbellay and Slama [(45)] also used a simple neural network for data of a similar style, reporting outstanding results. Setzler et al. (2009) [(154)] also provided a comparative study of call volume predictions using an ANN. In this work, the call volume is predicted at different temporal and spatial granularities. The approach offered an improvement at low spatial granularity and at hourly level.

3.2.3 Discussion

Now, we present a brief discussion about the positive points and existing limitations of the above-described predictive techniques.

The main advantage of RM is the clearness to understand and track the model. Nevertheless, it is hard to choose the variables to generate the model considering seasonality and trend, which are crucial to better understand the behaviour of a DMAS. Besides, the model may not reach convergence in some cases.

An important point for stationarising a TS is to be able to acquire meaningful sample statistics such as means, variances, and correlations with other variables. Such statistics are useful as descriptors of future behaviour only when the series is stationary. This way, if the series is consistently increasing over time, the sample mean and variance will consequently grow with the size of the sample, and they will always underestimate the mean and variance in future intervals. If the mean and variance of a series are not properly defined, then neither are its correlations with other variables. For this reason, it is important to be prudent about trying to extrapolate regression models fitted to non-stationary data.

The main advantage of *ES* methods is that they require short computing times [(7)], [(20)]. Nevertheless, the model cannot accurately predict for a long prediction window [(122)]. Another setback of this technique is its low performance when there is a trend as the single coefficient α is not enough to fit the outcome.

The principal advantage of ARIMA TS is that it usually suites better than ES methods, although this model requires longer computing times [(41)] and also poorly forecast for large time-horizons [(183)]. Given these premises, we can realise how promising exponential smoothing is to forecast data with no trend or seasonal patterns. Instead, DTTS should be applied whether there is linear trend. For shifting data, ES is remarkably well-adjustable, although its speed depends upon α.

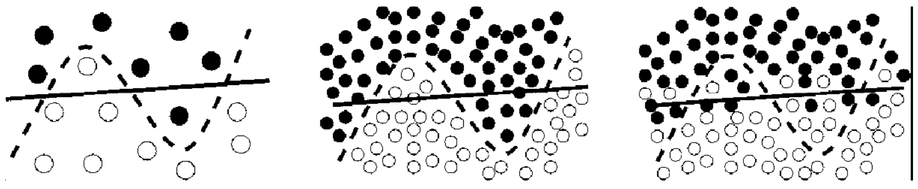

Figure 3.6: Overtraining risk - Volume of data.

On the one hand, the main advantage of ANNs is their flexibility to make patterns, being suitable for large and complex datasets as well as long-term forecasting [(41)], [(183)], [(124)], [(155)]. On the other hand, we can also find some disadvantages: longer training times, risk of overfitting, need of a feature selection process and difficulty to fine-tune all the required parameters [(183)]. The overfitting, also called overtraining, is the consequence of reducing the error in a specific dataset. When an ANN is trained during a large number of epochs (an epoch is the presentation of the entire training set to the ANN), the function determined by the weights of the ANN may take the particular characteristics of the examples. If this happens, the results will be optimal for the training dataset but no guarantee is given for any other one (see Figure 3.6 [(7)]).

3.3 Combinatorial Optimisation Techniques for DMAS

Chapter 1 presented the two types of algorithms for DMAS we can find in the state-of-the-art. There exists a kind of ad-hoc algorithms conceived for short-term planning environments in which a permanent planning is required because of the high variability of the system. Instead, there are other techniques devised for more stable (long-term planning) environments. Nevertheless, when facing a dynamic system, the second approaches cannot be efficiently applied, since an adaptive method is needed.

In this section, we describe different techniques that could be applied to the problem of workforce distribution in dynamic multi-agent systems (stable environments are out of the scope of this dissertation). Note that the purpose of this section is to briefly describe these techniques rather than to deeply detail them as the reader can carefully peruse the references provided in the following subsections, if desired.

3.3.1 Generic Optimisation Techniques

In the related literature, we can find numerous references to memetic algorithms that were applied to different assignment problems. A very nice compilation can be found in [(43)]. That work examined the application of MAs to different problems (scheduling, planning and timetabling). The article also described the basic architecture of an MA, and presented some guidelines for the design of a successful MA for those applications.

There are other alternatives in the state-of-the-art solutions. For instance, in computer science, **Local Search** (LS) [(84)] is a *Meta-Heuristic* (MH) for solving computationally hard optimisation problems. LS can be pertained to problems that can be formulated as finding a solution by maximising or minimising a criterion within a set of candidate solutions.

Frequently, the neighbourhood is composed by more than one solution where the choice of which one to move to is taken by only considering information concerning the solutions within the neighbourhood of the current one. When we select a neighbour solution taking the one, which maximises the criterion, then the MH is named *hill climbing*.

LS based algorithms "navigate" the search space, jumping from a solution to another one until a solution deemed optimal is reached or a given computing time has been elapsed. Another common choice is to terminate when the best solution found by the algorithm has not been improved in a given number of steps. LS algorithms are typically incomplete algorithms, as the search may stop even if the best solution found by the algorithm is not optimal. This can happen even if termination is due to the impossibility of improving the solution, as the optimal solution can lie far from the neighbourhood of the solutions crossed by the algorithms.

LS algorithms have been extensively applied to numerous hard computational problems, including problems from computer science, mathematics, operations research, engineering and bioinformatics [(179)].

To conclude, we provide the pseudo-code adapted to the problem of the workforce distribution in dynamic multi-agent systems, which illustrates the LS algorithm in its basic form:

```
void **Local_Search** (Chromosome & candidate_solution)
begin
    Chromosome best_solution = candidate_solution;
    Chromosome neighbour = candidate_solution;
    for $i \leftarrow 0$ to $candidate\_solution.size() - 1$ do
        Agent a = neighbour.getAgent(i);
        for $j \leftarrow 0$ to $a.get\_number\_profiles() - 1$ do
            neighbour.change_profile(i,j);//profile $j$ for agent $i$
            if $(neighbour.fitness() > best\_solution.fitness())$ then
                best_solution = neighbour;
            end
            neighbour = best_solution;
            j++;
        end
        candidate_solution = best_solution;
        i++;
    end
end
```

Algorithm 1: Basic LS pseudo-code.

Another option consists of employing a **Genetic Algorithm** (GA) [(83)], [(71)]. GA is a class of adaptive stochastic optimisation techniques, which attempts to find exact or approximate solutions for optimisation and search problems. GAs were proposed by John Holland in 1975 [(83)]. GAs are also a particular class of *Evolutionary Algorithms* (EA) that use techniques derived from evolutionary ideas of natural selection and genetics such as inheritance, mutation, selection, and crossover or recombination. It is important to highlight that GAs always work with a set of candidate solutions.

GAs are implemented in a computer simulation in which a population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals or phenotypes) to an optimisation problem evolves toward more accurate solutions.

The evolution typically begins with an initial population of randomly generated individuals and occurs over time by means of generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on either their fitness or composition), and modified (recombined and randomly mutated) to compose a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm ends up when a given number of generations has been produced, after a period of time, or after x generations without evolution, or a satisfactory fitness level has been accomplished for the population. If the algorithm has ended up due to a maximum number of generations, a satisfactory solution may or not have been reached.

```
Procedure **Basic__Genetic__Algorithm**
begin
    Generate an initial population of individuals;
    Evaluate each individual from the population;
    while !stopping_condition do
        Pick the best individuals for reproduction;
        Breed new individuals by means of the crossover;
        Apply a small perturbation over these new individuals;
        Evaluate their individual fitness;
        Replace the worst individuals;
    end
end
```

Algorithm 2: Basic GA pseudo-code.

The application of a **Memetic Algorithm** (MA) [(133)] is another good candidate for solving our problem. MA represents one of the current growing areas of research in EC. MAs are a population-based technique for heuristic search in optimisation problems [(139)] [(140)] [(80)] . These are much faster than traditional GAs for many problem domains. Fundamentally, these combine GA's operators with LS heuristics (an LS algorithm typically refines the solution obtained by the GA's operators).

Conversely, the continuous application of LS as a refinement mechanism does not guarantee a better performance. The frequency and the intensity characterise the level of progression (exploration) in opposition to the refinement achieved (exploitation) in the MA search. Thus, a more intense exploitation implies having more chances of convergence to the local optima. Evidently, it highly depends on the stage where the algorithm is, so it is broadly agreed that exploration should be more important at the beginning of the process and exploitation should be performed at the end [(110)], [(13)], [(12)]. However, we will see in this work that success can be achieved by dynamically adapting exploration and exploitation, depending on the circumstances found in our search.

For these reasons, some researchers have successfully denoted MAs as Hybrid GAs while others consider them as class of MHs. Frequently, MAs are also referred to in the literature as Baldwinian EAs, Lamarckian EAs, cultural algorithms, or genetic LS.

```
Procedure **Basic_Memetic_Algorithm**
begin
    Generate an initial population of individuals;
    Evaluate each individual from the population;
    while !stopping_condition do
        Pick the best individuals for reproduction;
        Breed new individuals by means of the crossover;
        Apply a small perturbation over these new individuals;
        Evaluate their individual fitness;
        Replace the worst individuals;
        Each $g$ generations, refine the $k$ best individuals with a LS procedure;
    end
end
```

Algorithm 3: Basic MA pseudo-code.

Other authors have applied **Dynamic Programming** (DP) [(54)] for similar optimisation problems. DP is a technique, which basically breaks problems down into smaller overlapping sub-problems. The philosophy of DP relies on solving problems where we need to find the best decisions serially. DP takes less time than other methods when it is applicable, because the results of certain calculations are stored and can be re-used by succeeding operations.

We can also find approximations with **Branch and Bound** (BB) [(113)] in related works. BB is a broad-spectrum algorithm devised for discrete and combinatorial optimisation problems. It systematically itemises all candidate solutions, from the uppermost one to the lowest one, discarding unproductive candidates. Every node (candidate solution) at a level l in the search tree corresponds to a partial sequence of p operations.

Differently to LS, **Random Neighbour Search** (RNS) consists in jumping from a candidate solution to a random neighbour (note that basic LS sequentially explores the neighbourhood). If the hop implies an improvement of the candidate solution, the best solution is updated and then considered as new candidate solution. This process is carried out until a given computing time has been elapsed or a fixed number of random neighbours has been generated.

Other possibility would be **Variable Neighbourhood Search** (VNS) [(8)]. VNS is an MH whose fundamental idea is to cause a systematic, stochastic change of neighbourhood within an LS. VNS escapes from local optima by changing of neighbourhood. To achieve it, VNS increases the size of the neighbourhood until a local optimum, better than the current one, is reached [(78)] . In this line, we can also find **Variable Neighbourhood Descent** (VND) [(21)]. VND is an MH where the search is not restricted to only one neighbourhood as in the LS but, instead, it deterministically changes at the same time as the algorithm advances (predefined sizes for the neighbourhoods).

One of the most used approximations is **Simulated Annealing** (SA) [(101)]. SA is an MH of variable search environment, which generalises Monte Carlo's method. SA proposes that the current state of a thermodynamic system is equivalent to the candidate solution in optimisation, the energy equation for a thermodynamic system is analogous to a target function and the ground state corresponds to the global minimum. This technique has the ability to hinder getting trapped in local optima since the algorithm allows for changes that decrease the values returned by the target function with a given probability. This probability depends on the current temperature value, which varies according to the cooling scheme. The main complexity is to determine the right value for the initial temperature and the cooling scheme.

Other authors have used *Tabu Search* (TaS). The meaning of the word **tabu** (also known as *taboo*) refers to a prohibition imposed by social customs as a protective measure [(69)]. In particular, TaS is based on the principle that search techniques should incorporate adaptive memories and guiding exploration mechanisms. The adaptive memory [(66)], [(67)], [(44)] allows for the implementation of procedures that are capable of economically and effectively navigating the search space. These memories introduce complexities that often confound alternative approaches as they allow for restriction of the search environment and the introduction of intensification mechanisms in zones of the search space that have been already visited, or diversification in possible zones of the search space, which are rarely visited [(44)] .

In contrast, **Scatter Search** (SS) [(68)] works over a set of solutions (reference points) by merging them in order to produce new feasible ones. The combination of solutions is commonly accomplished in a linear way. These combinations can be devised as a feasible generalisation of the existing solutions.

However, the basic idea of **Iterated Local Search** (ILS) [(98)] is to concentrate the search on a smaller subspace defined by the solutions, which are locally optimal to the current one. ILS consists in the iterative application of an LS method. To avoid getting trapped in local optimums, a perturbation is applied before executing each LS.

Other alternative to apply is **Multi-Start Search** (MSS) [(103)]. There are two phases in MSS: initially, a feasible solution is generated and, afterwards, is normally improved by means of an LS procedure. MSS is relatively simple because it merely executes several LS's from different initial solutions. The stopping condition for each LS is then taken as a restarting criterion. The most imperative disadvantage of improving each solution by means of an LS procedure is the possibility of getting ensnared in a non-optimal local optimum. MSS heuristics are earmarked to obtain limited solutions as far as the LS procedure cannot avoid escaping of non-promising environments. A key issue for the performance of MSS is whether (or not) the information about the topology of the neighbourhood (corresponding to the distance among neighbour solutions) is used.

The **Greedy Randomised Adaptive Search** (GRASP) [(58)] is one among those MSS methods whose first phase (constructive phase) randomly generates a greedy solution. The second phase (refinement) iteratively improves every solution by applying an LS procedure. Greedy randomised solutions are generated by injecting new elements to the problem's solution set from a list of elements ranked by a greedy function according to the quality of the solution (problem dependent). This method provides an appropriate and simple framework to develop algorithms for hard optimisation problems. The goal of this methodology is to combine the diversification strategy given by the construction phase with the intensification given in the improvement phase.

Another famous technique in this line is **Ant Colony Optimisation** (ACO) [(50)]. ACO is a stochastic method, which can be applied to problems that can be simplified to finding the right paths within a graph (usually, the shortest ones). Pheromone is a chemical substance secreted by a living organism that transmits a message inducing other members of the same species to react in a certain way. In our case, virtual ants deposit pheromones once they have built their solutions. The release of such a chemical signal, although systematic, is not constant. It is, instead, dependent upon the heuristic desirability of transition. This pheromone release is carried out once the solution is complete and is only updated when the loop ends. In order to refine the ants' generated solutions an LS procedure can be added to this algorithm. An ant *a* chooses to go forward to the following node with a determined probability.

Finally, **Particle Swarm Optimization** (PSO) [(99)] is another technique, which does not require any knowledge of the gradient of the problem to optimise. PSO emulates the behaviour of a group of birds that are flocking. PSO keeps a population of candidate solutions (particles) and then shifts them around in the search space in accordance with a more or less straightforward formula.

3.3.2 Discussion

With the purpose of summarising the previous descriptions, Table 2 shows the efficacy of each strategy to find a nearly optimal solution on a real-world DMAS in relation to the time-frame considered.

We can distinguish methods based on LS from those rooted in *global search*. Global search takes into account the whole search space whereas LS approaches can be applied to problems, which can be devised as finding a solution maximising (or minimising) a criterion among a number of candidate solutions. An LS algorithm starts out from a candidate solution and, thus, iteratively moves to a neighbour solution, generating the neighbourhood until a solution deemed optimal is reached or a predefined amount of time has been elapsed. The main problem with LS methods is that these usually get stuck in local optimums, which are often far from the global optimum. This setback can be mainly mitigated in five distinct ways:

1. The first possible solution, exemplified by VND, is to *modify the environment* (also known as neighbourhood). In VND, the search is not only restricted to

Table 3.2: Comparison of strategies' efficacy on a real-world DMAS in relation to the time-frame considered. MA is highlighted because it is our choice for this problem.

Algorithm	*Short-Term Efficacy*	*Middle-Term Efficacy*	*Long-Term Efficacy*
MA	**low**	**high**	**high**
ACO	low	low	medium
Basic LS	low	medium	low
BB	low	low	high
DP	low	low	high
GA	low	medium	medium
GRASP	low	medium	medium
GWB	medium	low	low
ILS	low	medium	medium
MSS	low	medium	medium
PSO	low	low	medium
RNS	low	low	low
RWB	low	low	low
SA	low	medium	high
SBR	medium	low	low
SS	low	medium	high
TaS	low	medium	high
VND	low	medium	medium
VNS	low	medium	medium

one environment as LS imposes; instead, the size of the environment deterministically changes as the algorithm progresses. The change of environment is a technique that is dependent upon the stage at which the algorithm is currently working.

2. The second possible solution is to *permit deterioration movements*, such as in SA or TaS. In the SA method, each point of the search space is equivalent to a state of some physical systems, and the function *E(s)* to be minimised is similar to the internal energy of the system in that state. The aim is to bring the system, from a random initial state, to a state with the smallest amount of energy. TaS increases the performance of an LS method by employing memory structures.

Once a potential solution has been reached, it is marked as *tabu* so that, the algorithm does not "visit" that possibility recurrently.

3. The third possible solution is to *restart from another initial solution* as MSS, GRASP, ILS or VNS do. In the case of the MSS, initial solutions are randomly generated and, afterwards, the algorithm applies an LS over them as a fine-tuning mechanism. This is equivalent to executing several LS in parallel. Therefore, the accuracy of the results will depend upon the number of executions that are launched. However, this is an inefficient method because a conscious stopping condition has to be provided. Conversely, ILS applies a mutation operator before each execution to attain an intermediate solution, which is refined by an LS. VNS (very similar to VND) is an ILS method, which changes of environment when the solution obtained is worse than the current one. Finally, GRASP relies on the use of a randomised greedy in its basic version.

4. Another way to find a good solution involves using methods based on *populations*, such as GAs and MAs. If the diversity of the population is low, then the GA converges to the closest neighbour. In contrast, if the selective pressure is high, which makes the diversity low, individuals will be alike or even identical. GAs are a powerful global search technique that slowly converges to the global optimum for a set of relevant real-world problems. MAs emerge as an improvement of this mechanism in which an LS is applied over a subset of individuals each n-generations.

5. Finally, there are other strategies to obtain a feasible solution such as *constructive methods* (e.g. ACO), which develop a solution instead of refining a given solution.

The MHs presented above provide diverse methods to escape from local optima.

The empirical impact of these MHs has been immense. Diverse tendencies on MH schemes have been explored by many researchers. The most relevant issue, provided by the incorporation of such techniques, is to know whether the benefit of the performance enhancement compensates for the effort of its implementation.

Frequently, trendy appealing heuristics are skilfully figured out. Also, great effort and inventiveness has been deployed in the adjustment of numerous parameters, but as yet the reasons that make them work still remain unknown. When facing a dynamic real-world production environment, some techniques (we will present an empirical study in Chapter 6) cannot perform well enough.

Intuitively, although RWB and RNS require low computing times, these will not be appropriate for a real production environment as they do not guarantee accurate solutions and their robustness is weak. A randomly generated solution can be acceptable as an initial solution, but not as a proper search mechanism. With luck on our side, we might find a good solution, but we would rapidly notice that these methods do not always perform properly. In fact, the probability of obtaining the global optimum is *1/nsl* where *nsl* stands for the number of possible solutions in the search space. Imagine a die with *nsl* faces (a very large number) with the added problem that we can only throw that die *nt* times in each time-frame (where *nsl » nt*).

GWB and SBR “route” work items to the available agents by applying basic heuristics, considering the current state of the system. Obviously, these techniques can be perfectly employed in DMAS but these fast, unplanned decisions may guide the algorithm to congestion states (evident need of a better planning, which takes into account future states).

In basic LS, a neighbour is generated every iteration. Theoretically, due to its local character, it is difficult to reach a high-quality solution because, when a local minimum is found, the algorithm will often stagnate as deterioration movements are not allowed.

In VND and VNS, the search is not simply restricted to a single search environment, but instead, the environment changes as the algorithm advances (deterministically in VND and stochastically in VNS). Therefore, the improvement of basic LS is remarkable as we will verify in Chapter 6.

TaS and SA introduce a very sophisticated mechanism of deterioration movements. However, these techniques only perform better when the time-frame is not very reduced: SA takes time even when we apply a Cauchy's scheme, which is the fastest one and TaS requires many iterations to take advantage of using the memory structures.

MSS increases the probability of finding an accurate solution as compared to basic LS as many LS's are run in parallel. In contrast, GRASP improves this philosophy by means of a probabilistic greedy procedure. This greedy process reassures us that, on one hand, that initial solution will be more or less promising and, on the other hand, that other local minima may be found, since the algorithm can start from different initial solutions.

Constructive techniques (e.g. ACO), although they are a very promising growing area, are not fast enough to be applied to real-world DMAS as [(49)] demonstrates.

Finally, GAs offer a different mechanism to finding precise solutions based on a population schema. Generally, GAs converge very slowly to the global optimum (sometimes there may be multiple optima) but, when these are combined with LS procedures (e.g. MAs), GAs are an astonishingly powerful search technique. For this reason, we will propose, in next Chapter, a search module based on a parallel MA.

Chapter 4

A Novel Approach to the Problem of Workforce Distribution in DMAS

This chapter proposes a novel approach to the problem of workforce allocation in DMAS, using ideas from different fields of artificial intelligence. In order not to dilute (nor blur) the scientific contributions exposed in this chapter, we will just focus on the innovative general approach and its original proposals applied to each specific component. The rest of details to fully develop these components is given in Chapter 5 and Chapter 6. Section 4.1 sets out the underlying idea of this novel approach. In Section 4.2, we describe the methodology that we have followed. Section 4.3 explains how to create a self-adaptive time-frame mechanism. Section 4.4 focuses on the predictive component. Section 4.5 addresses the optimisation component of our approach.

4.1 Underlying Idea

We have illustrated in Chapter 1 how short-term planning techniques distribute tasks to the next available agents having the required skills to process them by employing greedy heuristics, while long-term planning techniques contend with a stable incoming task flow over time, which is not the archetypal scenario in nowadays DMAS. This way, short-term planning strategies distribute workforce without considering future system states (just the current global picture), causing inapt allocations *task-agent* in the middle-term. In contrast, long-term planning strategies find optimal solutions for a given system state, requiring long computing times. However, if the system is not very stable, we might have serious problems in the future, because an optimal configuration for the current system state may not be the best option for a forthcoming point in time.

The underlying hypothesis of this work, which will be proved and confirmed in Chapter 6, is that most DMAS require accurate allocations of incoming tasks to the right available agents, considering a self-adaptive time-frame, rather than continuous naive/greedy assignments in real-time or static long-term configurations for remote future system states. In highly dynamic systems, it is preferable to make strategic planning for a middle-term time-frame (assuming certain noise as predictions are not 100% accurate) than to make poor planning for a short-term time-frame repeatedly.

We basically need to enlarge (or reduce) the observed time-frame and then forecast the real system state in a future time point in order to apply more sophisticated search algorithms, which can outperform short-term planning techniques. There, a need of obtaining an exact prediction of a middle-term system state comes out. Subsequently, a powerful search algorithm must find a feasible solution for the predicted system state, thereby assuring a fair balance between diversity (exploration) and intensity (exploitation) in order to meet with success [(161)].

The forecasting module will be founded on a ANN with a novel learning algorithm, while the optimisation module will be based on a parallel MA with an external meta-GA to automatically recalibrate its internal parameters depending on the dynamism of the system.

4.2 Methodology

Prior to explaining each component of our approach in detail, we will present the overall process in order to clarify the steps to take.

Firstly, we need to determine the size of the prediction window, also denoted as *time-frame* (v), where the system will be stable enough. During this time, we can employ the system configuration given by an optimisation mechanism in the previous time-frame. In the first iteration, we can use a configuration built from historical data.

Thanks to the adaptive time-frame, we can have knowledge of the time a determined configuration is adequate, so that after that time point, this configuration is no longer valid. Therefore, we can infer that the size of the time-frame will not be fixed over time and will depend upon the changes in the behaviour of the system (variability in the arriving task flow). In order to study this variability, we will use dispersion metrics and time series that may trigger the system's transition function. This function will cause transitions between system states, which have associated a different dynamism level (e.g. *very low, low, medium, high or very high*).

This way, we need to employ different system configurations for each time-frame and, while a given configuration is being applied to the current state, a search module must find the right setting-up for the next time-frame (note that it takes some time to find a nearly-optimal solution). In order to perform this optimisation, we need to predict all the variables involved in that future state by means of a forecasting module.

The algorithm below also describes the steps of our approach to better understand the underlying idea.

```
Procedure Approach to the problem of workforce distribution in
DMAS
begin
    Analyse the dynamism of the system with time series;
    Set up the size for the time-frame;
    Forecast all variables for next state (ANN) ;
    Optimise the assignment among tasks and agents for that future state
    (Parallel MA);
    Go to next state;
end
```

Algorithm 4: Pseudo-code of our global approach.

Figure 4.1 illustrates the complete mechanism graphically, reflecting the points in time where each component is running. This figure shows an example for *2* time-frames of different sizes and illustrates the relationship between modules for these time-frames.

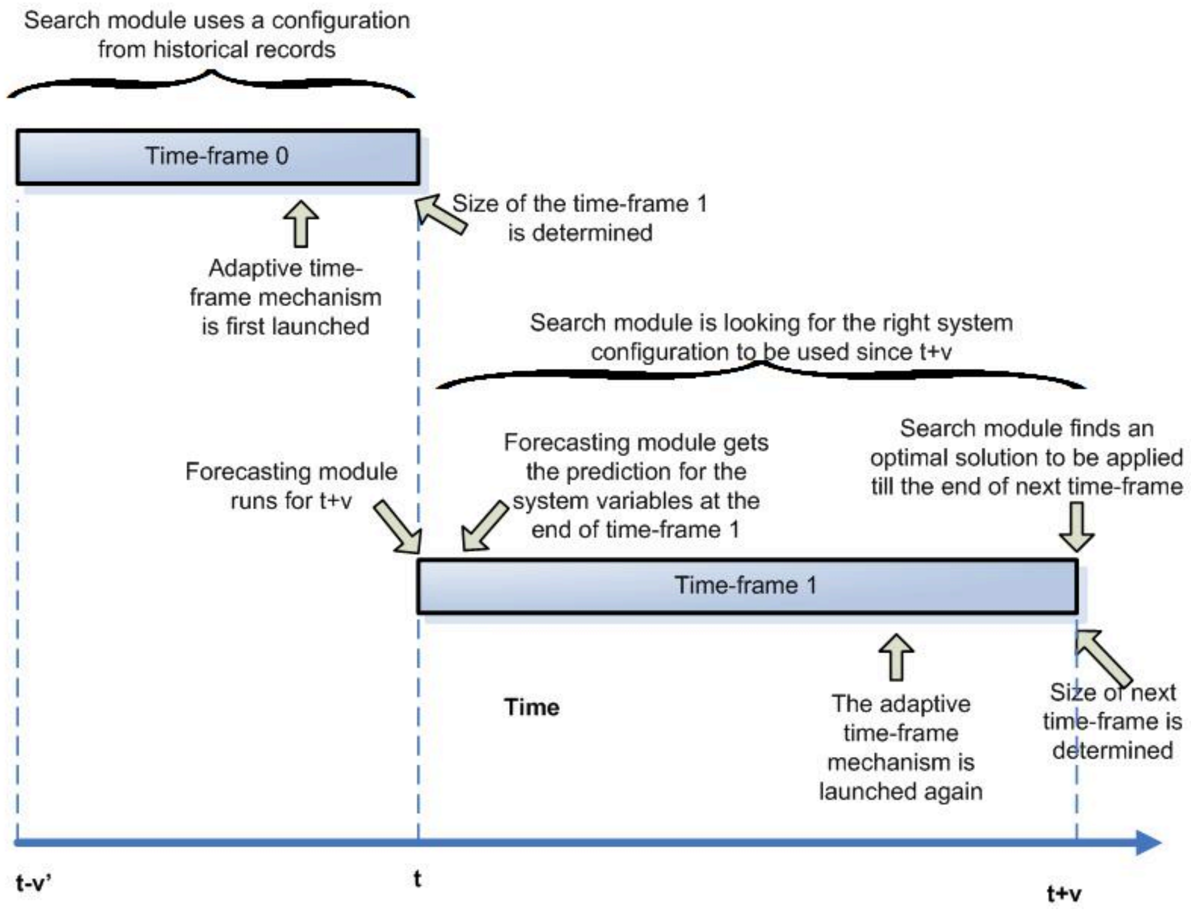
Search module uses a configuration from historical records
Time-frame 0
Size of the time-frame 1 is determined
Adaptive time-frame mechanism is first launched
Search module is looking for the right system configuration to be used since t+v
Forecasting module runs for t+v
Forecasting module gets the prediction for the system variables at the end of time-frame 1
Search module finds an optimal solution to be applied till the end of next time-frame
Time-frame 1
Time
The adaptive time-frame mechanism is launched again
Size of next time-frame is determined
t-v'
t
t+v

Figure 4.1: Relationship between the components of our approach from a timeline perspective.

Now, let us provide some more detail. If we start at time t_0 and v_0 is the size of the current time-frame, the forecasting module predicts all the system's variables at time $t_1 = t_0 + v_0$. At time t_1, the system is about to suffer a transition to new system state. From time t_0 to time t_1, we employ the allocation of tasks and agents that the search module produced in the previous time-frame (from time $t_{-1} = t_0 - v_{-1}$ to time t_0, where v_{-1} is the size of the previous time-frame) or information from historical records when we launch the first iteration. Once we have predicted those future variables for t_1 (by means of an upgraded resilient back-propagation neural network), we launch the search module again to get the optimal allocation between agents and tasks for that future state at t_1, which will be applied in the following time-frame $t_2 = t_1 + v_1$ (See Figure 4.2 for further information about the complete mechanism).

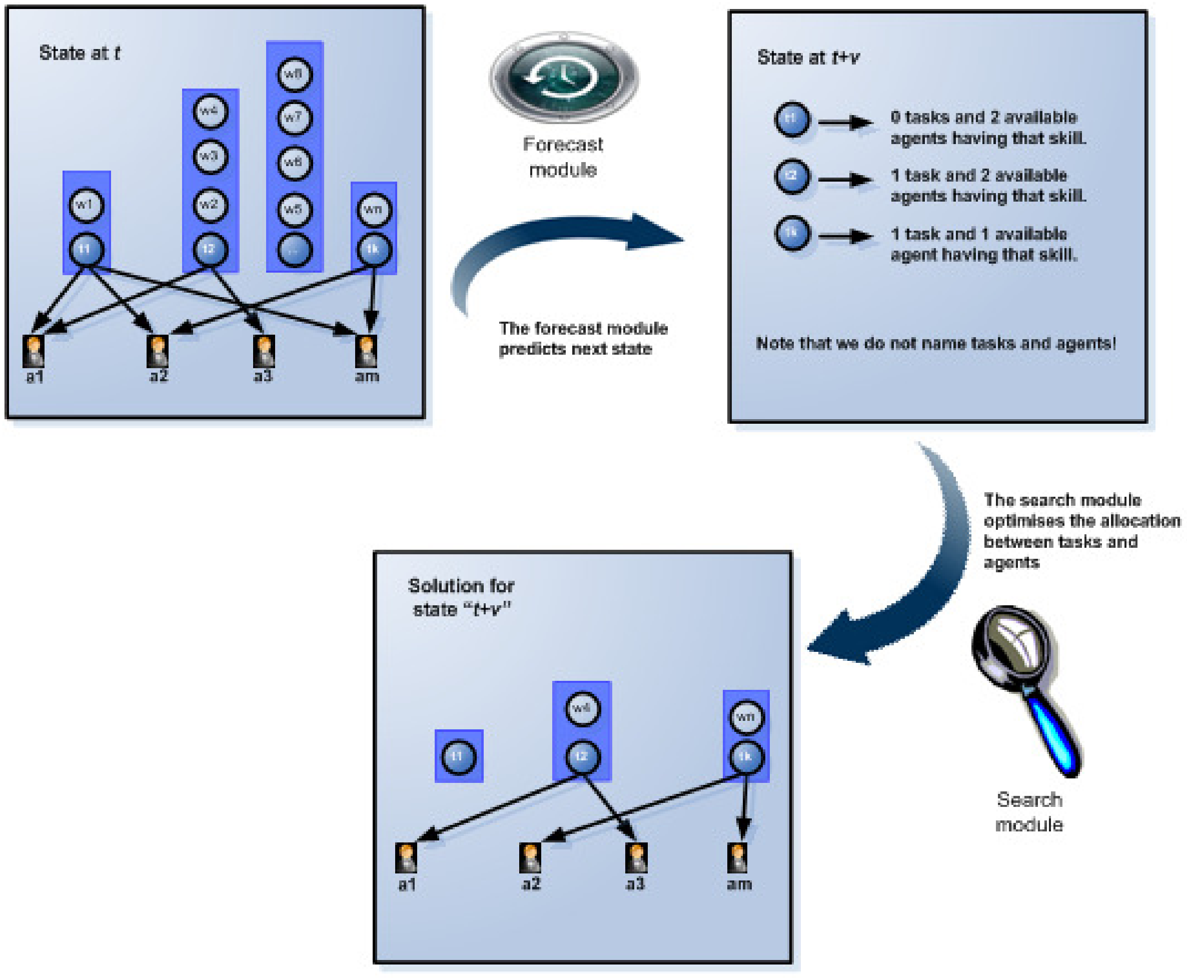
State at t
w8
w7
w6
w5
w4
w3
w2
w1
wn
t1
t2
tk
a1
a2
a3
am
Forecast module
The forecast module predicts next state
State at t+v
0 tasks and 2 available agents having that skill.
1 task and 2 available agents having that skill.
1 task and 1 available agent having that skill.
Note that we do not name tasks and agents!
The search module optimises the allocation between tasks and agents
Search module
Solution for state "t+v"

Figure 4.2: Overall process of the forecast module + search module.

The search module is implemented as a parallel MA with an external meta-genetic algorithm to automatically self-tune all the internal parameters in accordance with the variability of the system (preserving a fair balance between exploration and exploitation). Thus, the meta-GA is launched over historical data in order to determine the best configuration for the internal parallel MA for different levels of dynamism. Note that we will apply different parameter configurations for each level of dynamism because the internal parallel MA will have more or less time to find a solution as the size of the time-frame is variable. For the internal parallel MA, we propose an island topology and migration operators for individuals exchanging. Each island corresponds to a single MA. Each MA maintains a set (population) of abstract representations (chromosomes) of candidate solutions (phenotypes) according to the problem described in Chapter 2.

4.3 Adaptive Time-Frame

The mechanism of the adaptive time-frame in turn consists of two components: the *determination of the size of the time-frame* (we continuously analyse the dynamism of the system to determine the right size of the time-frame) and the *state-transition function* (when the dynamism of the system changes significantly).

To address the first difficulty (size of the time-frame), we can set up predefined time-frame sizes (see Figure 4.3), depending on the dynamism level of the system (e.g. *3* levels: *low*, *medium* and *high*). Obviously, this choice must be done according to a previous, exhaustive statistical analysis. In our experiments, we have considered *5* levels depending on system variability: *very low* (*v=3000* seconds), *low* (*v=1500* seconds), *medium* (*v=300* seconds), *high* (*v=120* seconds) and *very high* (*v=60* seconds). The reasoning for selecting predefined time-frame sizes is given by the requirement of robustness that any real-world DMAS habitually imposes. If we enable the system to automatically assign any size for the time-frame, we may crash down the system (for instance, we may have *2*-second time-frames or *20*-hour time-frames, which might seize up the system).

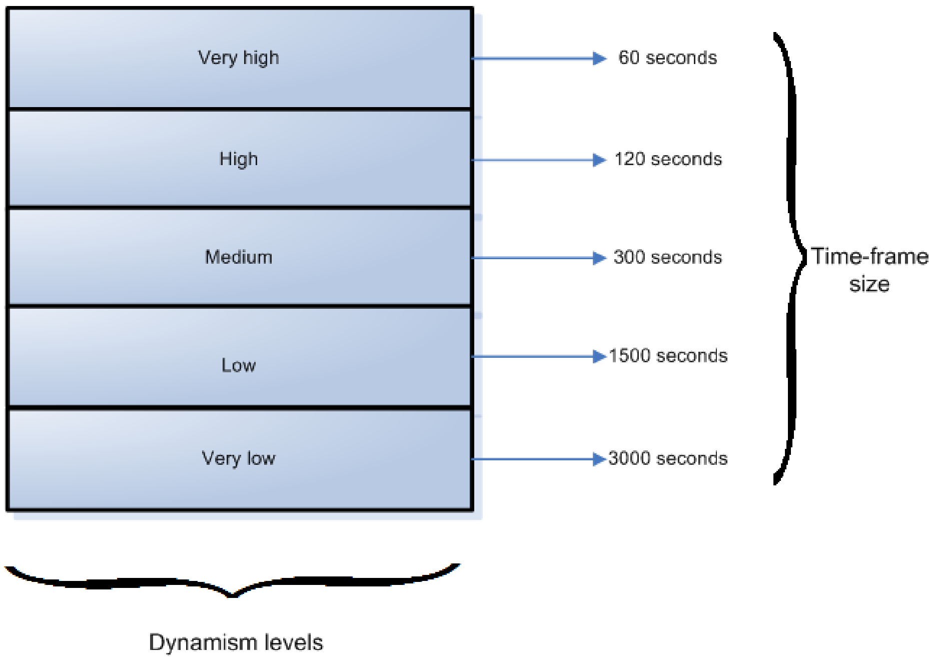

Figure 4.3: Time-frame sizes depending on the dynamism level.

To deal with the second complexity (when to change of dynamism level), we must determine the right state-transition function. This is a problem-dependent task and we cannot claim any universal rule of thumb. Instead, we propose some guidelines to accomplish with this arduous task. During the statistical analysis, we encourage the reader to analyse smaller intervals than the time-frame (let us say *30* seconds). Then, we should break down this interval into subintervals (e.g. *5* subintervals of *6* seconds) to describe a time series. If a given point highly differs from the previous one, we should not activate the state-transition function as peaks may crash down the system. But, when the trend of the time series drawn by these consecutive points shows important oscillations, we should switch to another dynamism level (e.g. if the tendency shows an important dynamism decrement, we should then enlarge the time-frame by switching to a less dynamic level).

In summary, we take into account the *trend* and *dispersion* of these consecutive time points with a double exponential smoothing time series as described in the state-of-the-art section. With this input, we can cluster any time series into any of the *5* dynamism levels that we have defined, according to a similarity metric (we have employed the Euclidean distance [(116)]).

Figure 4.4 shows an example of a *300*-second time-frame. For this time-frame, we analyse a smaller (shifting) time-frame of *30* seconds with *5* equidistant consecutive points (each *6* seconds). If we plot these points, we would have a time series (number of arriving tasks at each time point). We may discover numerous possible situations but, in this figure, we have only illustrated *6* different cases (bear in mind that a rigorous statistical study must be performed to achieve it). Figure 4.4.A shows a very changing time series (high dispersion without well-defined trend); therefore, the dynamism level would be *very high*. Figure 4.4.B exemplifies a quite dynamic time series but with fewer changes than Figure 4.4.A, so the dynamism level would be *high*. Figure 4.4.C and Figure 4.4.D point up *medium* dynamism level because there is a clear trend (increasing in Figure 4.4.C and decreasing in Figure 4.4.D). Figure 4.4.E illustrates a time series without changes, therefore the dynamism level should be *very-low*. Finally, Figure 4.4.F presents a time-series with few changes, thus, the dynamism level would be *low*.

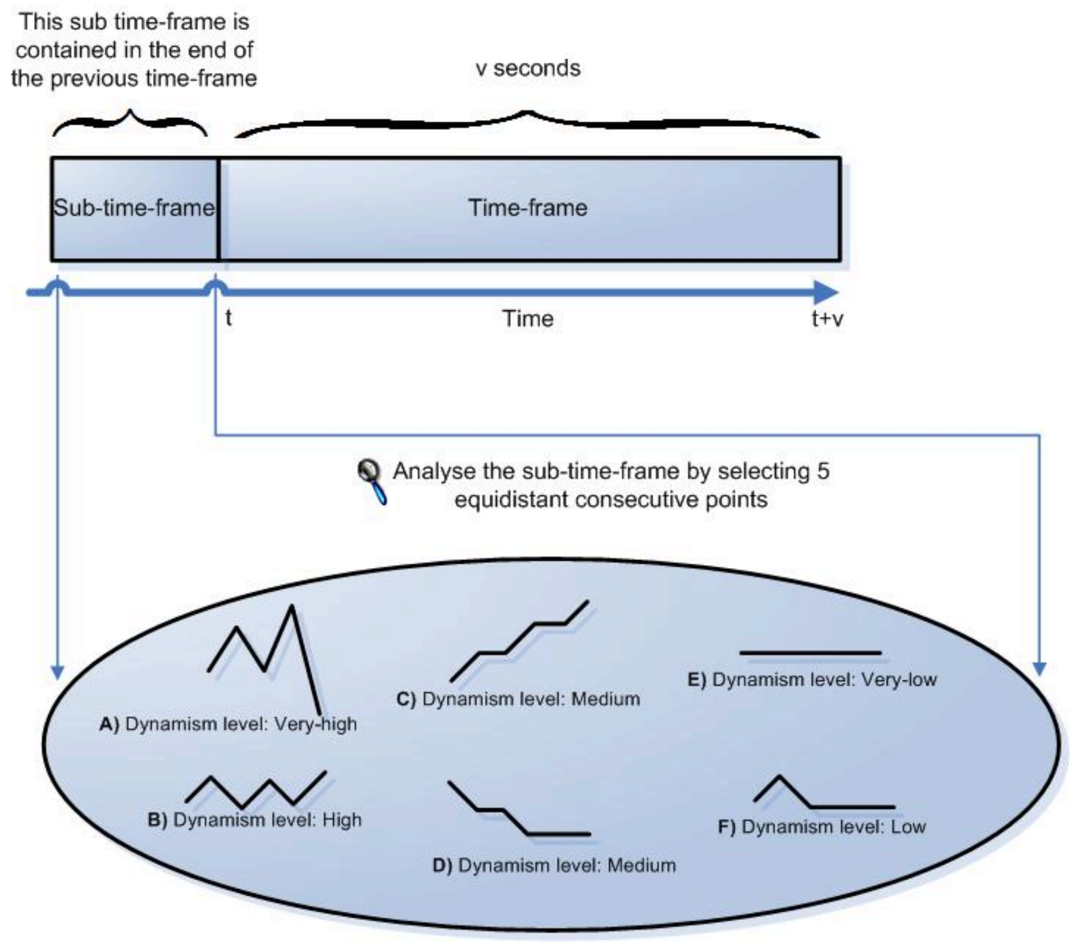
This sub time-frame is contained in the end of the previous time-frame
v seconds
Sub-time-frame
Time-frame
t
Time
t+v
Analyse the sub-time-frame by selecting 5 equidistant consecutive points
A) Dynamism level: Very-high
B) Dynamism level: High
C) Dynamism level: Medium
D) Dynamism level: Medium
E) Dynamism level: Very-low
F) Dynamism level: Low

Figure 4.4: Example of different time-series within a shifting sub-time-frame.

Naturally, the sub-time-frame is a rolling time window that must be shifted in relation to the time. Finally, note that we do not impose "sequentiality" when changing of dynamism level as Figure 4.5 exemplifies. Arrows symbolise that we can reach every state from any other state. Circles represent the dynamism levels. Note that we have not plotted self-pointing arrows as no transition is needed but, of course, as any regular Markov model [(127)], there is certain likelihood of remaining in the same state in next time-frame.

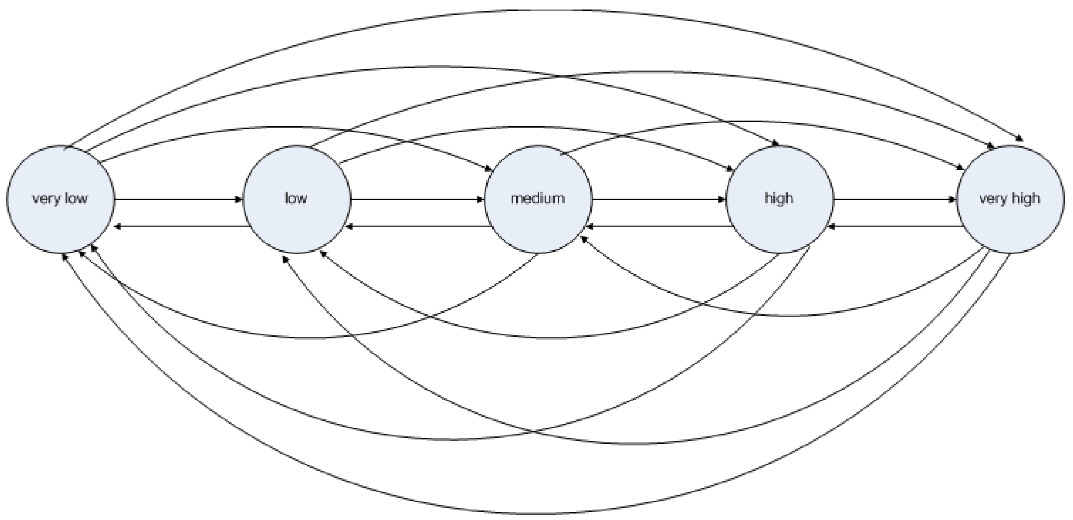

Figure 4.5: Potential dynamism level transitions.

Now, let us formalise the definitions exposed along this section. Denote the initial state at time t_0 as ξ_0 where we know all system variables $(W_0, T_0, A_0, S_0, P_0)$. We just mean that at the beginning (time t_0), we know the number of pending tasks, their types, the number of available agents, the potential skills and the prospective profiles (we highly encourage the reader to briefly review Section 2.2 to refresh the meaning of each variable). Also, denote the current state at time t as $\xi_t(W_t, T_t, A_t, S_t, P_t)$ and designate next future state at time $t+v$ as $\xi_{t+v}(W_{t+v}, T_{t+v}, A_{t+v}, S_{t+v}, P_{t+v})$, where v is the size of the current time-frame.

Finally, denote the state-transition function as $\delta : \xi_t / v_{dynamism\_level} \rightarrow \xi_{t+v}$. This just means that every state ξ_{t+v}, depends on the previous state ξ_t and the transition occurs each v seconds (size of the time-frame, which depends on the dynamism level). The variables of a given state are directly visible to the observer.

Note that short-term planning strategies consider intermediate systems states $(\xi_t < \xi_{intermediate} < \xi_{t+v})$, whereas long-term planning strategies take into account posterior system states $(\xi_t < \xi_{t+v} < \xi_{posterior})$. Therefore, short-term planning strategies rely on smaller transition-state steps (δ) than our approach, while long-term planning strategies hinge on longer transition-state steps (δ) than our approach. Graphically, this characteristic can be seen in Figure 4.6. Notice that ξ_i stands for $\xi_{intermediate}$ and ξ_p represents $\xi_{posterior}$. Also, note that we have not plotted ξ_0. In a real-world production environment, we can set up an initial system configuration for ξ_0 that considers historical records. From this first configuration for the initial state ξ_0, we should employ the mechanism we propose (prediction window determination based on system dynamism + current system state prediction + optimisation).

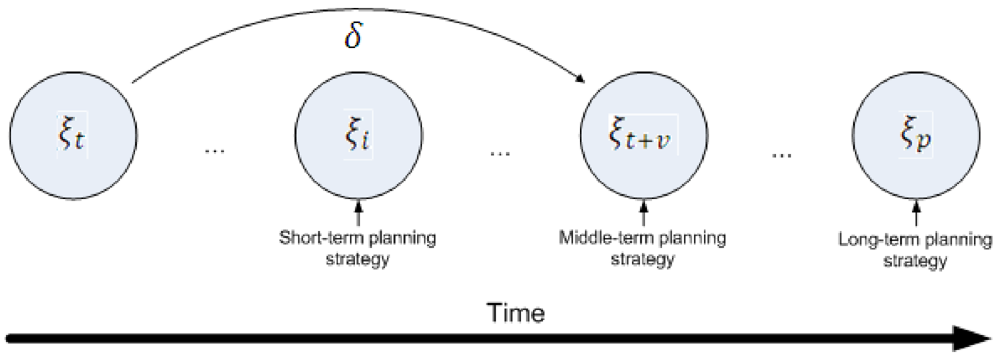
δ
ξ_t
...
ξ_i
...
ξ_{t+v}
...
ξ_p
Short-term planning strategy
Middle-term planning strategy
Long-term planning strategy
Time

Figure 4.6: System states depending on the time-frame considered.

4.4 Forecasting module

This section describes how to model a forecasting module founded upon an enhanced, self-adaptive gradient-descent based learning algorithm (an upgraded resilient back-propagation) to predict the volume of task arrivals at a forthcoming point in time. Section 4.4.1 provides the required background to understand our learning algorithm for neural networks. In Section 4.4.2, we formulate the mathematical basis of our learning algorithm.

Note that our centre of attention will be the innovative features of the forecasting module, not on the generic aspects of the inner ANN. The details of the ANN and the particularisation of the application are given in Chapter 5 and Chapter 6.

We have presented a mechanism to determine the size of the time-frame as well as a method to track the dynamism level. From this information, we need to forecast all the system variables at time t+v. Obviously, even when the forecast is pretty accurate, we are introducing some noise to the system as we are searching for nearly optimal solutions for a predicted system state, which may slightly differ from the real future state. In order to mitigate the impact of that noise, it is crucial to design a powerful, accurate forecasting module, which can provide us with the best possible approximation of next future system state.

Until now, we have presented copious numbers of variables from any ordinary DMAS (see Section 2.2). Although we have to consider all these variables in order to attain a feasible solution, uncertainty chiefly comes from the number of pending tasks grouped by task types and the number of existing available agents having each skill. As tasks continuously appear and require of a certain processing time to be executed, and given that the size of the time-frame v is variable (a smaller number when there is great dynamism or a larger number when there are few oscillations); we can assume that the number of pending tasks and available agents at time $t_1 = t_0 + v_0$

(apart from the new incoming tasks) depends upon the number of pending tasks and available agents at time t_0 as some tasks may not be accomplished during these v seconds.

4.4.1 Background

BPAs can be categorised as multilayer perceptrons [(146)], which have non-linear activation functions [(82)] such as the logistic function [(100)], the softmax function [(30)], the Gaussian function [(117)], among others [(142)], [(153)]. BPA denotes that any error made by the ANN when returning an output during the training process, is sent backwards with the purpose of learning and distinguishing what is right from what is not. Errors are propagated backwards from the external nodes to the internal ones. Therefore, BPA is employed to calculate the gradient error of the ANN with respect to its adjustable weights. This gradient is often used in a simple stochastic gradient-descent algorithm to find the weights that minimise that error. BPA simply takes the derivative of the cost function with respect to the ANN parameters and then changes those parameters in a gradient-related direction.

The most notorious problem with gradient-descent methods is the premature convergence to local optima, which might be far from the global optimum. This problem can be solved by applying global optimisation techniques. However, these techniques normally require high training times, which are not always compatible with highly dynamic environments like DMAS. Nonetheless, other improved gradient-based learning algorithms such as Resilient Backpropagation (Rprop) (Riedmiller, 1992) [(148)] with more global information can be more appropriate.

Rprop is a robust ingrained modification of classical gradient-descent method. This scheme tends to fine-tune an individual step-size to optimise each parameter. The mechanism to perform this action entails doing adaptations of these step-sizes by applying a more or less complex heuristic, instead of considering proportional step-sizes to the partial derivatives. Note that classical gradient-descent algorithms calculate the steepest descent direction by means of an Euclidean metric.

Classic Rprop just takes into account the sign of the partial derivative $\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}}$ (partial derivative of the error measure with respect the weight between two neurons i and j) in order to resolve the direction of the weight updating. When there is a change of sign of the partial derivative between two consecutive time points, we can state that a local minimum has been surpassed as there is a change of direction in the search space surface. Besides, we have to update the weights and automatically adapt the step-size, considering the sign the partial derivative.

Although we will see how to achieve this weight updating with classic Rprop and our modification later on, we recommend reading the full description of Rprop algorithm given in [(148)].

4.4.2 An Innovative Adaptive Learning Rate Algorithm for Resilient Back-Propagation Neural Networks

Specifically, this thesis proposes an upgraded, adaptive modification of the standard Rprop with weights backtracking (uRprop) learning algorithm in order to make the learning phase more adaptive to environmental circumstances.

The main purpose is to properly determine the right weights of our ANN. A great challenge is to find out how big step-sizes (learning rate speed) should be. Note that selecting the right learning rate is always a laborious task.

Rprop produces an expected output that is compared to the well-known output (real value) and this signal is then propagated, calculating the differentials among errors in accordance with the weights (gradients). To update the weights between each pair of neurons i and j (w_{ij}) after each pattern (epoch; [(24)]), we inspect the previous weights as follows [(24)]:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{4.1}$$

where $\Delta w_{ij}(t)$ can be formalised like so [(148)] (classical Rprop):

$$\Delta w_{ij}(t) = \begin{cases} \Delta_{ij}(t) & \text{if } \frac{\partial \Lambda\epsilon(t)}{\partial w_{ij}} < 0 \\ \\ -\Delta_{ij}(t) & \text{if } \frac{\partial \Lambda\epsilon(t)}{\partial w_{ij}} > 0 \\ \\ 0 & \text{other case} \end{cases} \tag{4.2}$$

where $\Delta_{ij}(t)$ is the step-size and $\Lambda\epsilon$ is the error measure (it can be defined as a normalised mean absolute error for the generalisation dataset between two consecutive epochs). Weights updating $\Delta w_{ij}(t)$ is carried out until the stopping condition is met. In our experimental analysis, we will employ a stopping criterion based on a fixed number of epochs as we have hard computing constraints in our environment, but we could otherwise have considered a fixed amount of elapsed time, a given error metric value, a percentage of error, a number of epochs without improvement, etc.

Classical Rprop [(148)] (without weights backtracking) just takes into account the change of sign of the partial derivative $\frac{\partial \Lambda\epsilon(t)}{\partial w_{ij}}$ (change of direction in the search space surface). This precisely means that a local minimum has been surpassed because the previous step-size taken has been too long. An important improvement to classical Rprop was to include weight updates, thereby enabling backtracking movements [(159)]. Weight updating $\Delta w_{ij}(t)$ leads to adjusting $\Delta_{ij}(t)$ by applying the following formula [(89)]:

$$\Delta_{ij}(t) = \begin{cases} \min(\gamma \cdot \Delta_{ij}(t-1), \Delta_{max}) & \text{if } \frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda \epsilon(t-1)}{\partial w_{ij}} > 0 \\ \\ \max(\frac{\Delta_{ij}(t-1)}{\mu}, \Delta_{min}) & \text{if } \frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda \epsilon(t-1)}{\partial w_{ij}} < 0 \\ \\ \Delta_{ij}(t-1) & \text{other case} \end{cases} \tag{4.3}$$

where γ is a multiplying coefficient greater than *1* ($\gamma > 1$) to enlarge the step-size and μ is a dividing coefficient greater than *1* ($\mu > 1$) to shorten it, depending on the surface of the search space. After an exploratory analysis, we have chosen $\gamma = 1.4$ and $\mu = 2$ for our experiments.

When $\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda \epsilon(t-1)}{\partial w_{ij}} > 0$, the signs of the derivatives do not change ("+" by "+" or "-" by "-" is always positive). This means that we have not reached the local minimum yet. Therefore, we increase the step-size until we have surpassed a local minimum ($\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda \epsilon(t-1)}{\partial w_{ij}} < 0$). When ($\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda \epsilon(t-1)}{\partial w_{ij}} < 0$), there is a change of sign in $\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}}$. This implies that we have already jumped over the local minimum. When the local minimum is surpassed, we change the sign of the gradient.

Authors like Jacobs (1988) [(94)] or LeCun (1998) [(114)] typically limit the step-size with $\Delta_{min} = 0$ and $\Delta_{max} = 50$. Instead, we propose $\Delta_{min} = 0.001$ and $\Delta_{max} = 30$ as limits ($\Delta_{min} = 0.001$ for the number precision needed and $\Delta_{max} = 30$ as this already implies a long movement).

Up until now, we have defined how to adjust the step-size $\Delta_{ij}(t)$ at time t, depending on the sign of the partial derivative. But we still have to update $\Delta w_{ij}(t)$. Tollenaere (1990) [(168)] proposed an important improvement to classic Backpropagation, which laid in weight updates with backtracking (reverting a wrong movement or step). Following the same reasoning, Igel and Husken (2003) [(90)] incorporated these ideas into the classic Rprop, reverting only those movements that entailed a higher error measure. When the error measure was lower, the step was not reverted but the direction of the gradient was changed.

Let us now elucidate our contribution. We distinguish three cases in Equation (4.6) when there is a change of sign in the partial derivative ($\frac{\partial \Lambda\epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda\epsilon(t-1)}{\partial w_{ij}} < 0$):

- the error measure is lower than the previous one so that we proceed as Igel and Husken (2003) proposes (not reverting the step).
- the error measure is a small percentage larger than the previous one, so that we interpolate the previous step-size, staying halfway.
- the error measure is much larger so that we revert the last step taken.

Our modification is based on weight backtracking movements but we also consider local information of the search space surface by means of the previous error measure $\Lambda\epsilon(t-1)$. When there is a change of sign in the partial derivative, we compute Δ_{ij} (t) as defined in Equation 4.3. This way, we verify whether the current error measure $\Lambda\epsilon(t)$ is a percentage α larger than the previous error measure $\Lambda\epsilon(t-1)$ (i.e. $\alpha = 15\%$; note that this threshold should not be greater than a $15-20\%$). When this occurs, we undo the previous movement $\Delta_{ij}(t-1)$ as we have not only surpassed the local minimum but also gotten a much higher error measure. However, when the deviation is lower than a percentage α but still larger than the previous error measure (we are further from the local optimum but not extremely far away), we go back halfway as fully reverting a movement leads us to wasting too many iterations. If we stay halfway between the previous point of the search space and the current one ($\frac{\Delta_{ij}(t-1)}{2}$), the probability of getting closer to the local minimum increases 2 times in each step and the total number of iterations needed to reach the local optimum decreases in accordance therewith. Finally, when the current error measure is lower than the previous one, we do not revert the movement but update the direction of the gradient. Hence, we propose to apply the following expressions for each w_{ij} according to the three cases we have previously proposed:

$$\text{if } \left(\frac{\partial\Lambda\epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial\Lambda\epsilon(t-1)}{\partial w_{ij}} > 0\right) \Rightarrow$$

$$\Rightarrow \begin{cases} \Delta_{ij}(t) = \min(\gamma \cdot \Delta_{ij}(t-1), \Delta_{max}) \\ \\ \Delta w_{ij}(t) = \begin{cases} \Delta_{ij}(t) & \text{if } \left(\frac{\partial\Lambda\epsilon(t)}{\partial w_{ij}} < 0\right) \\ -\Delta_{ij}(t) & \text{if } \left(\frac{\partial\Lambda\epsilon(t)}{\partial w_{ij}} > 0\right) \\ 0 & \text{other case} \end{cases} \\ \\ w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \\ \\ \frac{\partial\Lambda\epsilon(t-1)}{\partial w_{ij}} = \frac{\partial\Lambda\epsilon(t)}{\partial w_{ij}} \end{cases} \tag{4.4}$$

$$\text{else if } \left(\frac{\partial\Lambda\epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial\Lambda\epsilon(t-1)}{\partial w_{ij}} < 0\right) \Rightarrow$$

$$\Rightarrow \begin{cases} \Delta_{ij}(t) = \max(\frac{\Delta_{ij}(t-1)}{\mu}, \Delta_{min}) \\ \\ \Delta w_{ij}(t) = \begin{cases} -\frac{\Delta w_{ij}(t-1)}{2} & \text{if} \begin{cases} (\Lambda\epsilon(t) > \Lambda\epsilon(t-1)) \\ \&\& \\ (\Lambda\epsilon(t) < \alpha \cdot \Lambda\epsilon(t-1)) \end{cases} \\ -\Delta w_{ij}(t-1) & \text{if } \Lambda\epsilon(t) > \alpha \cdot \Lambda\epsilon(t-1) \\ 0 & \text{other case} \end{cases} \\ \\ w_{ij}(t+1) = \begin{cases} w_{ij}(t) - \frac{\Delta w_{ij}(t-1)}{2} & \text{if} \begin{cases} (\Lambda\epsilon(t) > \Lambda\epsilon(t-1)) \\ \&\& \\ (\Lambda\epsilon(t) < \alpha \cdot \Lambda\epsilon(t-1)) \end{cases} \\ w_{ij}(t) - \Delta w_{ij}(t-1) & \text{if } \Lambda\epsilon(t) > \alpha \cdot \Lambda\epsilon(t-1) \\ w_{ij}(t) & \text{other case} \end{cases} \\ \\ \frac{\partial \Lambda\epsilon(t)}{\partial w_{ij}} = 0 \end{cases} \tag{4.5}$$

else if $(\frac{\partial \Lambda\epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda\epsilon(t-1)}{\partial w_{ij}} = 0) \Rightarrow$

$$\Rightarrow \begin{cases} \Delta_{ij}(t) = \Delta_{ij}(t-1) \\ \\ \Delta w_{ij}(t) = \begin{cases} \Delta_{ij}(t) & \text{if } (\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} < 0) \\ -\Delta_{ij}(t) & \text{if } (\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} > 0) \\ \Delta_{ij}(t-1) & \text{other case} \end{cases} \\ \\ w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \\ \\ \frac{\partial \Lambda \epsilon(t-1)}{\partial w_{ij}} = \frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} \end{cases} \tag{4.6}$$

Note that our learning algorithm might sometimes get trapped in local minima because we are looking at the error measure, which may be misleading when the search surface has many oscillations (two close points may have very different error measures associated). But, as compared to Rprop, our approach is faster and usually obtains better results for reduced amounts of training time.

We have defined a generic modification of Rprop with weights backtracking algorithm for ANNs but we still have to formalise the rest of problem-dependent parameters of the ANN (topology activation function, inputs, outputs, hidden units, etc.). Section 5.1 describes how to fine-tune our ANN and specifies how to achieve this on a real-world environment, which is the vast call centre of Telefonica.

4.5 Search module

This section describes the key features of the search module (only the novel contributions as details are given in Chapter 5 and Chapter 6), which is the last component of the architecture proposed in Figure 4.2 (second arrow). This block is implemented as a parallel MA.

Once the forecasting module has provided us with all the predictions, the search module optimises the assignment among task types and agents. The parallel steady-state MA is devised as an island topology with migration operators for individuals exchanging, where each island corresponds to a fully-functional steady-state MA.

For the configuration of the parallel MA, we employ a novel meta-GA algorithm that determines the right setting for each level of dynamism. Of course, the meta-GA only works with historical data and provides different outcomes depending the size of the time-frame. When there is a system transition, we use the corresponding parameters configuration for the parallel MA, so that the algorithm can explore or exploit the search space, taking into account the time the system will be stable in the next time-frame. This mechanism provides a more flexible and automatic parameter selection.

Section 4.5.1 provides the required background to understand our meta-GA and, in Section 4.5.2, we explain its details.

4.5.1 Background

Some authors have already developed meta-GAs in the past. [(182)] was one of the pioneers in using GAs to optimise problems over several real parameters. [(115)] proposed an automatic fuzzy system design method that used a GA and integrated three design stages. Their method determined membership functions, the number of fuzzy rules and the rule-consequent parameters at the same time. [(38)] used a Meta-GA to investigate the evolution of parameter settings (genetic operators) for genetic and evolutionary algorithms in the hope of creating a self-adaptive algorithm. [(136)] presented and evaluated a method for estimating the relevance and calibrating the values of the parameters of an evolutionary algorithm. The method provided an information theoretic measure on how sensitive a parameter was to the choice of its value. In [(137)], the same authors proposed an advanced method that helped to calibrate the parameters of an evolutionary algorithm in a systematic and semi-automated manner. The method for relevance estimation and value calibration of evolutionary algorithm parameters was empirically evaluated in two different ways. More recently, [(26)] made use of a Meta-GA to optimise the parameters of a simple GA through an evolutionary process. They addressed the problem of determining the electronic structure of long chain molecules. The same year, [(156)] proposed a methodology for both optimal pattern selection and tuning. They employed a robust GA to solve a project scheduling problem.

All these meta-GAs were focused on single GAs and MAs but we now propose a meta-GA for parameter calibration that automatically tests out different islands, migration and replacement configurations. It entails a number of independently evolving populations to determine the right setting-up of the internal parallel MA.

4.5.2 Meta-Genetic Algorithm for Parallel Memetic Algorithms

The chromosome of our meta-GA, which has *6* genes, follows an integer encoding scheme. These genes refer to diverse parameters that affect the final performance of the internal parallel MA (see Figure 4.7).

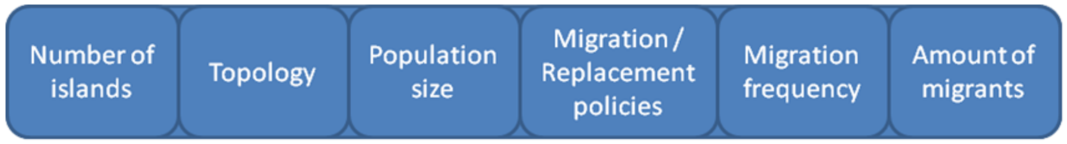

Figure 4.7: Encoding of the Meta Genetic Algorithm.

Let us present the pseudo-code before going on with the explanation (see below).

```
Meta-GA for Parallel Memetic Algorithm

    Generate a random population;
    Evaluate the individuals' fitness by running
    ...the internal parallel MA with current configuration;
    Store configurations and associated fitness to
    ...avoid running those configurations again in the future;
    $generations \leftarrow 0$;

While($generations \leq MAX_G generations$)

    Select a subset of individuals;
    Apply crossover;
    Apply Mutation;
    Check those configurations previously
    ...calculated and get their fitness values;
    Run the internal parallel MA for new configurations;
    $generations \leftarrow generations + 1$;


End While
```

Algorithm 5: Pseudocode of the Meta-GA for Parallel Memetic Algorithm

In our meta-GA, we encode each solution as an array of integers whose indexes represent each parameter and the array contents refer to the values that these parameters can take. These genes can take the following values:

- *Number of islands* (from *1* to *12* populations).
- *Topology* (star, bidirectional ring, all-to-all).

- *Population size* (from *4* to *100* individuals per island).
- *Migration and replacement policies*: Best-Fitted Individuals by Worst-Fitted Individuals (BFI-WFI), Best-Fitted Individuals by Random Individuals (BFI-RI), Best-Fitted Individuals by Best-Fitted Individuals (BFI-BFI), Best-Fitted Individuals by Most Different Individuals (BFI-MDI), Best-Fitted Individual + "Annealing" by Worst-Fitted Individuals (BFIA-WFI).
- *Migration frequency* (*30* or *60* seconds).
- *Amount of migrants* (percentage from 10% to 30%).

The evolutionary operators of the meta-GA has been set up as follows:

- *Fitness function*: We measure the service level resulting from each configuration.
- *Population size*: The population contains *20* different individuals encoded as hinted above.
- *Initialisation*: The initial population is randomly generated.
- *Selection*: Since the population needs to be bred each successive generation, we have chosen a binary tournament selection.
- *Crossover*: The offspring inherits the common points in their parents and randomly receives the rest of genes from them.
- *Mutation*: We apply a perturbation over each gene of the chromosome with a probability of *0.1*.

Chapter 5

Particularisation of the Problem and Details of the Solution

In Chapter 5, we describe the peculiarities of our specific environment (the multi-skill call centre), showing the concrete adaptations we have carried out to deal with our problem instances. In Section 5.1, we present some special adaptations for the forecast module. In contrast, Section 5.2 points out some particular modifications for the search module.

5.1 Forecasting Module Adaptations and Details

This section elucidates the required tuning to adapt the forecasting module described in the previous chapter to the MSCC's environment. Note that our forecasting module relies on a novel learning algorithm (for ANNs) that does not actually need any specific adaptation to our environment but the architecture of the ANN demands some additional tuning. Our focal control over this architecture lies in the number of hidden layers as well as the number of neurons in these layers because the number of input/output neurons is given by the number of inputs and outputs we have.

As the number of available agents mainly depends on the agent timetable and the number of drop calls is proportional to the number of incoming calls, we will mainly focus on the prediction of incoming calls. The following sections justify the pertinent configuration of our ANN for the MSCC's domain.

5.1.1 Number of Layers

The number of layers of an ANN must be, at least, two (*1* input, h hidden where $h \geq 0$ and *1* output). Sometimes, the hidden layer is not needed (e.g. simple linearly divisible problems). In our case, we propose three layers (or *4* layers at most): *1* input layer, *1* hidden layer and *1* output layer. Bear in mind that we necessitate a hidden layer at least because our problem is nonlinear. Nevertheless, we do not mandatorily need more than a hidden layer because we can well approximate every function by utilising a single hidden layer with an arbitrarily large number of hidden units (universal approximation property [(128)]). Of course, the more hidden layers we have, the more accurate our prediction may be (more coefficients in the global regression formula of the ANN). But, this increases the computing time to (re)train the network (more loops to update the weights) and we have very limited time to accomplish this task in dynamic environments. Besides, adding more than a hidden layer aggravates the problem of getting trapped in local minima [(24)].

5.1.2 Input Layer

The number of neurons of the input layer is given by the number of input attributes we have. But, what attributes or features do we have in our environment? At a typical MSCC, we can stumble on a wide variety of variables, which may range from information of previous calls (number of calls, tendencies, mean processing times, etc.) to contextual information (campaigns, peak hours, night shift timetable, etc.). We can directly take this raw information but, sometimes, we have too many variables or features to take them into account.

As we have much information and many dimensions (given by the input variables and the values they can take), it makes sense to reduce the number of input variables. Moreover, sometimes, some input variables may even contribute to the ANN with mutual (even identical) information because of the dependency among variables.

To mitigate the curse of dimensionality, feature selection appears as a promising solution. Feature selection is the technique that consists of selecting a subset of relevant features or variables in order to build robust learning models.

Choosing the right inputs from all information we have (e.g. *122* different input variables in our case) is not trivial and is very important for obtaining higher performance as having more predictors has the effect of adding new dimensions to the model (more complexity). Since variable selection should not be defined ad-hoc, *Principal Component Analysis* (PCA) [(95)] has been employed as Table 4 illustrates (first column stands for the relevance, second column refers to the component number and third column is the component itself).

PCA is a statistical technique that converts a set of potentially correlated pre-

Relevance	♯	Component
0.8514	1	0.182i_TENDENCIA_9 + 0.181i_TENDENCIA_18 + 0.179i_TENDENCIA_8 + 0.176i_TENDENCIA_17 + 0.167i_TENDENCIA_7...
0.7635	2	0.293i_LL_TOTALES_5 + 0.293i_LL_TOTALES_6 + 0.289i_LL_TOTALES_7 + 0.289i_LL_TOTALES_4 + 0.284i_LL_TOTALES_3...
0.6887	3	−0.213i_TENDENCIA_13 − 0.191i_TENDENCIA_12 − 0.19i_TENDENCIA_14 − 0.178i_TENDENCIA_4 − 0.176i_TENDENCIA_10...
0.6193	4	−0.245i_TENDENCIA_21 + 0.221i_TENDENCIA_11 + 0.221i_TENDENCIA_10 − 0.219i_TENDENCIA_22 − 0.205i_FLAG_21...
0.5563	5	−0.245i_TENDENCIA_33 − 0.207i_TENDENCIA_28 − 0.206i_FLAG_33 + 0.2i_TENDENCIA_43 + 0.199i_TENDENCIA_20...
0.4979	6	0.265i_TENDENCIA_38 − 0.26i_TENDENCIA_27 + 0.227i_FLAG_38 − 0.223i_FLAG_27 − 0.196i_TENDENCIA_45...
0.4413	7	0.288i_TENDENCIA_26 + 0.244i_FLAG_26 − 0.207i_TENDENCIA_32 − 0.196i_TENDENCIA_33 − 0.195i_TENDENCIA_3...
0.3861	8	0.257i_TENDENCIA_41 + 0.217i_FLAG_41 − 0.208i_TENDENCIA_45 − 0.205i_TENDENCIA_37 + 0.191i_TENDENCIA_7...
0.3323	9	−0.271i_TENDENCIA_1 − 0.233i_FLAG_1 − 0.202i_TENDENCIA_2 − 0.197i_TENDENCIA_6 − 0.19i_TENDENCIA_4...
0.2805	10	−0.34i_TENDENCIA_46 − 0.291i_FLAG_46 + 0.283i_TENDENCIA_44 + 0.243i_FLAG_44 + 0.202i_TENDENCIA_35...
0.2471	11	−0.45i_INTERVALO_8_2 − 0.376i_MINUTOS_DIA + 0.328i_INTERVALO_8_1 − 0.323i_INTERVALO_4_4 − 0.304i_HORA_PUNTA2...
0.2232	12	0.416i_INTERVALO_8_0 + 0.416i_NOCTURNO − 0.313i_INTERVALO_8_1 + 0.308i_INTERVALO_4_0 + 0.265i_INTERVALO_4_1...
0.2098	13	0.571i_INTERVALO_4_5 − 0.451i_INTERVALO_4_4 − 0.432i_HORA_PUNTA2 − 0.184i_DIA_SEMANA_6 + 0.181i_MINUTOS_DIA...
0.1995	14	0.432i_INTERVALO_4_3 − 0.414i_INTERVALO_4_2 + 0.244i_DIA_SEMANA_6 + 0.204i_DIA_SEMANA_5 − 0.186i_DIA_SEMANA_2...
0.1898	15	0.821i_DIA_SEMANA_2 − 0.447i_DIA_SEMANA_3 − 0.197i_DIA_SEMANA_1 + 0.166i_DIA_SEMANA_6 − 0.15i_DIA_SEMANA_4...
0.1802	16	−0.724i_DIA_SEMANA_4 + 0.524i_DIA_SEMANA_3 + 0.349i_DIA_SEMANA_6 − 0.164i_DIA_SEMANA_5 + 0.112i_INTERVALO_4_1...
0.1706	17	0.725i_DIA_SEMANA_5 − 0.478i_DIA_SEMANA_4 − 0.423i_DIA_SEMANA_6 + 0.159i_DIA_SEMANA_0 − 0.151i_DIA_SEMANA_3...
0.1611	18	0.685i_DIA_SEMANA_1 − 0.512i_DIA_SEMANA_3 + 0.313i_DIA_SEMANA_6 − 0.222i_DIA_SEMANA_4 − 0.207i_DIA_SEMANA_5...
0.1516	19	0.828i_DIA_SEMANA_0 − 0.487i_DIA_SEMANA_1 − 0.188i_DIA_SEMANA_3 + 0.104i_DIA_SEMANA_6 − 0.101i_DIA_SEMANA_2...
0.1428	20	−0.689i_INTERVALO_4_1 + 0.564i_INTERVALO_4_0 + 0.191i_INTERVALO_4_2 − 0.181i_INTERVALO_4_3 − 0.158i_DIA_SEMANA_4...
0.1351	21	0.443i_INTERVALO_4_2 − 0.364i_INTERVALO_4_3 − 0.32i_INTERVALO_4_0 + 0.26i_INTERVALO_4_1 + 0.182i_HORA_PUNTA...
0.1289	22	0.332i_FLAG_10 + 0.332i_FLAG_11 − 0.169i_TENDENCIA_11 − 0.169i_TENDENCIA_10 − 0.16i_HORA_PUNTA...
0.1233	23	0.41i_FLAG_11 + 0.41i_FLAG_10 − 0.186i_TENDENCIA_11 − 0.186i_TENDENCIA_10 + 0.175i_TENDENCIA_20...
0.118	24	0.597i_HORA_PUNTA + 0.339i_DIA_SEMANA_6 − 0.286i_INTERVALO_4_3 + 0.252i_DIA_SEMANA_5 − 0.196i_INTERVALO_8_1...
0.1132	25	−0.218i_FLAG_28 + 0.209i_FLAG_9 + 0.191i_FLAG_3 + 0.191i_FLAG_4 − 0.188i_FLAG_33...
0.1088	26	0.264i_FLAG_38 + 0.217i_TENDENCIA_45 − 0.214i_FLAG_45 − 0.209i_TENDENCIA_38 + 0.204i_FLAG_7...
0.1045	27	0.296i_FLAG_27 − 0.224i_TENDENCIA_27 + 0.22i_FLAG_30 − 0.205i_FLAG_37 + 0.203i_FLAG_26...
0.1002	28	0.275i_FLAG_20 + 0.224i_FLAG_13 − 0.219i_FLAG_33 + 0.201i_TENDENCIA_1 + 0.191i_TENDENCIA_2...
0.0961	29	0.249i_FLAG_26 − 0.239i_FLAG_35 − 0.214i_TENDENCIA_26 − 0.207i_FLAG_32 + 0.207i_FLAG_28...
0.0922	30	−0.258i_FLAG_41 + 0.234i_FLAG_37 + 0.23i_FLAG_39 − 0.224i_TENDENCIA_44 + 0.221i_TENDENCIA_41...
0.0884	31	0.282i_FLAG_11 + 0.282i_FLAG_10 − 0.249i_FLAG_16 + 0.245i_FLAG_24 + 0.241i_FLAG_21...
0.0852	32	−0.458i_HORA_PUNTA + 0.349i_DIA_SEMANA_6 + 0.228i_DIA_SEMANA_5 − 0.203i_FLAG_9 − 0.18i_DIA_SEMANA_0...
0.0822	33	0.274i_FLAG_36 − 0.266i_FLAG_45 + 0.239i_FLAG_4 − 0.234i_FLAG_15 + 0.228i_FLAG_35...
0.0792	34	0.276i_HORA_PUNTA − 0.271i_FLAG_30 + 0.243i_FLAG_42 + 0.219i_FLAG_38 − 0.218i_FLAG_29...
0.0763	35	−0.337i_HORA_PUNTA − 0.325i_FLAG_45 − 0.299i_FLAG_12 − 0.262i_FLAG_3 − 0.257i_FLAG_29...
0.0733	36	−0.314i_FLAG_14 − 0.312i_FLAG_39 − 0.276i_FLAG_21 − 0.265i_FLAG_40 − 0.247i_FLAG_38...
0.0705	37	0.328i_FLAG_41 − 0.301i_FLAG_7 − 0.278i_FLAG_45 + 0.258i_FLAG_42 + 0.245i_FLAG_1...
0.0676	38	0.433i_FLAG_1 − 0.36i_FLAG_46 + 0.312i_FLAG_2 − 0.29i_FLAG_8 + 0.261i_FLAG_44...
0.0648	39	0.515i_FLAG_46 − 0.303i_FLAG_44 + 0.291i_FLAG_1 + 0.259i_FLAG_17 + 0.244i_FLAG_24...
0.0622	0	0.568i_FLAG_2 − 0.503i_FLAG_1 − 0.277i_FLAG_19 + 0.257i_FLAG_12 + 0.199i_FLAG_25...
0.0597	41	−0.393i_FLAG_26 + 0.364i_FLAG_45 − 0.348i_FLAG_19 + 0.293i_FLAG_37 − 0.261i_FLAG_2...
0.0572	42	0.584i_FLAG_32 − 0.366i_FLAG_41 − 0.344i_FLAG_44 + 0.218i_FLAG_43 + 0.206i_FLAG_40...
0.0548	43	−0.583i_FLAG_37 − 0.404i_FLAG_43 − 0.324i_FLAG_19 + 0.281i_FLAG_38 + 0.259i_FLAG_39...
0.0524	44	−0.508i_FLAG_26 − 0.44i_FLAG_12 + 0.314i_FLAG_20 − 0.237i_FLAG_1 + 0.227i_FLAG_2...
0.05	45	0.467i_FLAG_32 + 0.432i_FLAG_41 + 0.414i_FLAG_44 − 0.289i_FLAG_42 + 0.216i_FLAG_46...

Table 5.1: Ranked attributes.

dictors into a smaller subset of uncorrelated predictors designated as principal components. The main advantage of PCA is the capability to compress data by reducing the number of dimensions without significant loss of information. However, PCA assumes that data dimensionality can be properly reduced by linear transformations. This problem can be removed by means of complementary dummy variables that are non-linear functions of the input data set elements. Another disadvantage of PCA is that the directions that maximise variance do not always maximise information.

To select the right generic inputs for any CC, we have compiled real data from *3* months and have used the R-Weka [(175)] framework in order to determine the right inputs to our ANN (see Table 5.1).

Among all variables (we have created trends, flags, rates, etc.), the volume of incoming calls in previous intervals, size of the time-frame, night shift timetable, week of the month, time, intervals of hours (broken down in blocks of *2*, *4* or *8* hours) and intervals of peak hours must be highlighted and analysed for separate.

Figure 5.1 shows the behaviour when factoring in the previous time intervals. For almost all CGs, the optimum number of previous intervals required is usually around *5-6* intervals. Considering more previous intervals does not enable us to obtain better results and makes the learning process slower.

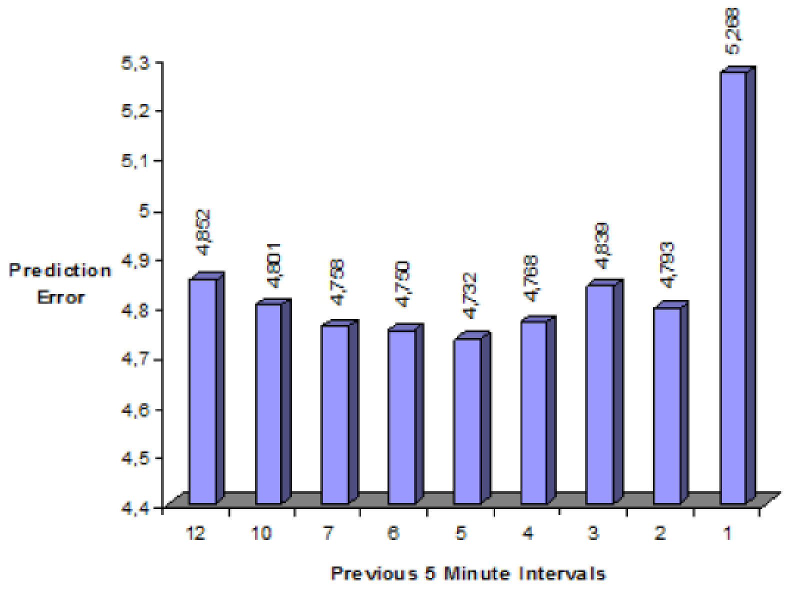

Figure 5.1: Mean absolute error returned by the ANN when considering previous 5-minute intervals.

Figure 5.2 illustrates a summary of the most relevant variables that have been studied in terms of error caused separately (not linearly combined with PCA). The night shift timetable offers an upgrading of the results for every CG. When splitting days up into intervals of hours, predictions are also improved. The improvement coming from adding these hourly intervals might guide us to a wrong decision because these variables are correlated with the current number of incoming calls (our target) but the causality comes from the night shift timetable and peak time variables. It is more valuable to know peak hours rather than have the whole hourly information catalogue. Note that the correlation among variables does not necessarily imply causality. In other words, the improvement is just obtained because these variables are correlated but only peak time intervals and night shift are truly useful to forecast the current number of incoming calls. Of course, PCA moderates the impact of these deceptive correlations.

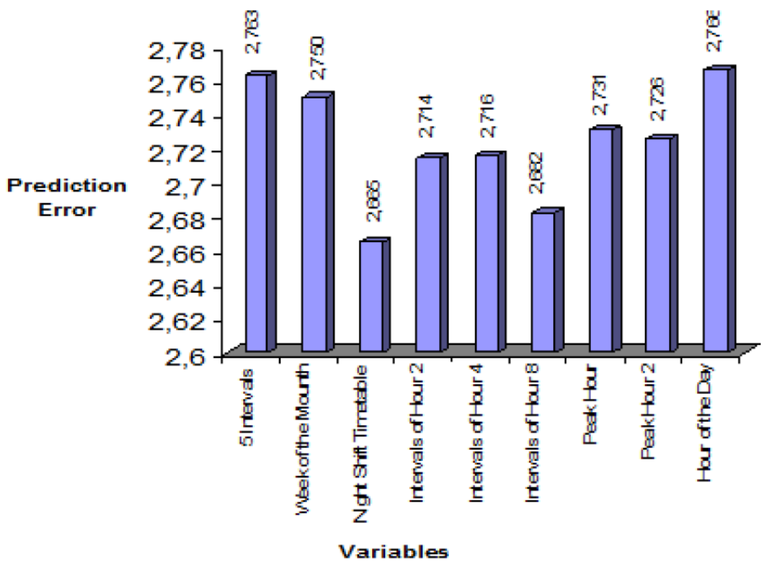

Figure 5.2: Mean absolute error returned by the ANN when adding different variables.

Intervals of peak hours are interesting to take into account because these divisions clearly outperform the results for almost all CGs. However, the improvement is intuitively a bit lower for those CGs without many oscillations as there are fewer differences among day partitions.

The rest of variables influence the results in some CGs (like the week of the month) but not significantly enough for most of them (they slightly contribute to

Individual relevant variables
Size of the time-frame
Calls in Previous 0-5 Minutes
Calls in Previous 5-10 Minutes
Night Shift Timetable
Week day
Calls in Previous 10-15 Minutes
Calls in Previous 15-20 Minutes
Calls in Previous 20-25 Minutes
Minutes of the Day
Peak Time
Second Peak Time

Table 5.2: Summary of the most relevant individual variables.

the target of the prediction). Since a quick response time is required and selecting more features involves a higher dimensionality, these variables have not been expressly included in our final implementation.

As showing all the components obtained when applying PCA may be hard to track, Table 5.2 summarises the most relevant individual features extracted from our dataset.

5.1.3 Hidden Layer

We have already revealed that our ANN has a single hidden layer. Now, we have to determine the number of existing hidden units (hidden neurons) located in this layer. There are some rule-of-thumb mechanisms to establish the right number of neurons of the hidden layer(s) but, in our case, this number has been empirically determined.

Figure 5.3 shows the results obtained for a variable number of hidden units. We can appreciate that the optimum value seems to be around *20* hidden units as MAE gets minimised.

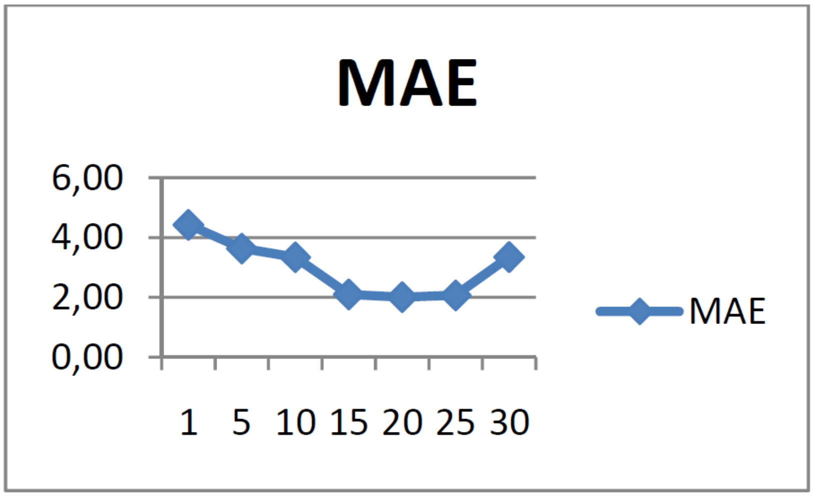

Figure 5.3: Mean absolute error, depending on the number of hidden units.

5.1.4 Output Layer

The last layer is named output layer and is used for unveiling and returning the result of the prediction. The number of neurons forming this layer is given by the number of output variables. In our case, only one variable is predicted (total number of incoming calls per CG) so that only one neuron forms the output layer. The output is a floating number, which indicates the number of incoming calls of a given CG (we need as many ANNs as CGs we have). As we may need multi-step forecast, we work over a rolling window so that predictions are then used as inputs for next iterations.

5.1.5 Parameter Initialisation

The neuron activation function that we have employed is the sigmoid function as it is very appropriate for dynamic environments. As for the weights between neurons, these are randomly initialised and then updated during the training process. In particular, we set them up to small values ranged in $[-0.5, 0.5]$. The idea of initialising the

Call Group	*MAE With Sets*	*MAE Without Sets*
CG 1	2.84524	2.87196
CG 2	2.34671	2.39941
CG 3	4.32158	4.44656
CG 4	1.40664	1.41888
CG 5	0.83214	0.94906

Table 5.3: MAE obtained for *5* different CGs with/without sets for *50* executions.

weights in this way is to reduce the number of epochs during the training process. Starting from weights that are closer to the required ones will perceptibly entail fewer changes in the training process.

Additionally, we can still outperform the results by including some specific knowledge of each CG. The large number of CGs (*1035*) and their miscellaneous behaviour make necessary to appropriately determine the initial parameters of the models (ANNs for each CG). To fulfil this requirement, the CGs have been divided into sets according to the mean number of incoming calls per day. This criterion has been taken as a consequence of the behaviour similarities of those CGs having similar volume of incoming calls. Therefore, we need to define different initial configurations for the step-sizes for these sets as well as the lower and upper bounds of the uRprop proposed in Section 4.4.

Table 5.3 demonstrates that we can still outperform the results a little by starting from different initial parameters depending on the CG behaviour. This table summarises the mean absolute error (MAE) gotten for *5* different CGs after *50* executions of their specific ANN.

The change proposed above does not vary our computing times but (slightly) improves the results, especially in those CGs that have more fluctuations in the arriving load.

5.1.6 Stopping Criterion

We consider the following measures to decide when to stop the training process:

1. *Maximum epochs reached*: the ANN will stop once a set number of epochs have elapsed (*1200*).

2. *Generalisation set mean squared error*: this is the average of the sum of the squared errors (real vs predicted values) for each pattern in the generalisation set (*MSE* $<$ *1 incoming call*).

5.2 Search Module Adaptations and Details

5.2.1 Encoding

Our solution consists in an integer representation. We just need an array of integers whose indexes represent the available agents, $A_t \subseteq A$, at a given instant, t, and the array contents refer to the profile, P_j, assigned to each agent a_i ($<P_1, ..., P_i, ..., P_l>$). Then, tasks are "routed" to the agents, according to the profiles assigned. Of course, we can also encode the solution as an array of integers whose indexes symbolise the task types and its respective contents represent the number of agents assigned to each task type. This option is recommended whether there are too many agents and hardware capacity is very limited (with respect the total number of available agents). In contrast, we are missing the capability of working at agent's profile level. As we have not this capacity constraint, we will employ the first codification proposed.

Figure 7.4 shows a fictitious example of encoding for *10* work items ($w_0, ..., w_9$) grouped in *3* different tasks types ($t_0, ..., t_2$) depending on the nature of the tasks, *5* agents ($a_0, ..., a_4$) and *4* skill profiles ($P_0, ..., P_3$), where $P_0 = \{s_0, s_1\}, P_1 = \{s_1\}$, $P_2 = \{s_2\}$ and $P_3 = \{s_1, s_2\}$. Now, suppose that $a_0 \sim \{P_0, P_1\}$, $a_1 \sim \{P_0, P_2\}$, $a_2 \sim \{P_1, P_3\}$, $a_3 \sim \{P_2, P_3\}$ and $a_4 \sim \{P_0, P_1\}$. We have seen the potential profiles for every agent but only one profile can be assigned to each agent at a given instant t; therefore, a feasible solution would be Figure 7.4 (a_0 and a_4 have been assigned to the profile P_0, a_1 and a_3 have been assigned to the profile P_2, while a_2 has been assigned to the profile P_1). This way, a_0 and a_4 can process task types t_0 and t_1; a_1 and a_3 can process the task type t_2; and a_2 can process the task type t_1.

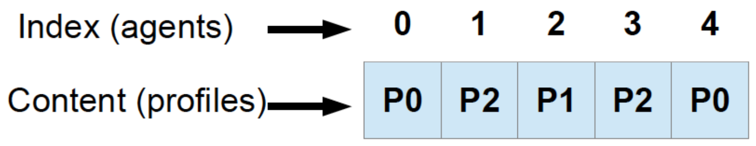

Figure 5.4: Example of encoding for *5* agents.

The population of our MA is a compilation of chromosomes encoded as hinted above. In our case, we propose to start from a random initial population, including the best solution found in the previous time-frame because the configuration of agents' profiles should not change too much over two successive time-frames.

5.2.2 Fitness Function

Now, we present the fitness function that is defined over the proposed encoding to measure the quality of a given solution. Our fitness function is inspired in the estimation of the *total service level* provided in [(106)] although we also consider the priority of each CG weighted as follows:

$$TotalServiceLevel = \sum_{i=0}^{k}(Pr_i \cdot SL_i(\gamma_i, \alpha_i)) \cdot \mu \quad \{SL : \Re X[0,1]X[0,1] \rightarrow [0,1]\} \quad (5.1)$$

where k refers to the number of CGs, μ is a normalising factor $(\frac{1}{\sum_{i=0}^{k} Pr_i})$, Pr_i is the priority of the CG_i whose service level is defined as:

$$SL_i(\gamma_i, m_i) = 1 - P(Agents\_are\_busy) \cdot e^{-(\gamma_i - m_i)\frac{\tau_i}{\beta}} \quad (5.2)$$

given that:

$$P(Agents\_are\_busy) = \left[1 + \frac{\gamma_i - m_i}{m_i} \cdot \sum_{\zeta=0}^{\gamma_i - 1} \frac{(\gamma_i - 1)...(\zeta + 1)}{m_i^{\gamma_i - \zeta - 1}} \right]^{-1} \tag{5.3}$$

where γ_i is the load of CG_i (number of incoming calls of CG_i by the mean processing time: $n_i \cdot \tau_i$), m_i is the number of agents of CG_i (based on the profiles assigned in the chromosome), τ_i is the number of agents of CG_i and β is the duration of the time-frame expressed in seconds.

Additionally, we handle some hard and soft constraints derived from the business rules given by our business units. In our case, these constraints are associated to tasks, agents, timing, actions or desired/undesired scenarios. Thus, the algorithm cannot violate hard constraints (e.g. we cannot change agents' profiles continuously due to certain laws and regulations); although we allow certain movements, which may imply the violation of some soft constraints (e.g. we should not take agents from CGs in which the service level is below a given threshold). Undoubtedly, this type of movements is penalised according to the degree of non-accomplishment of these constraints and their relevance as described in Chapter 2. Therefore, the *fitness function* can be formalised as follows:

$$f = (total\_service\_level - penalisations\_constraints) \quad f : [0,1]x[0,1] \rightarrow [-1,1] \tag{5.4}$$

where *penalisation_constraints* is the value obtained after applying our business rules (e.g. agents from CG-i should not move to CG-j).

Finally, we can speed-up the evaluations by introducing a *partial fitness function*. The first time, we need to employ (5.4) but the rest of the time; we just need to evaluate those groups affected by a mutation or, in the case of the LS, when generating a new neighbour. Hence, we only process the affected CGs in (5.1) and update their original values. With this information, we then recalculate (5.4).

5.2.3 Evolutionary Operators (Classic and New Operators)

In this subsection, we explain potentially appropriate evolutionary operators, which may be applied to the problem described in Chapter 2, given the encoding we are proposing in Subsection 5.2.1. This section does not attempt to cover all feasible evolutionary operators, just those that we consider, to the best of our knowledge, the most relevant ones for DMAS. Some of these evolutionary operators are innovative but others are not (this is specified for each one).

5.2.3.1 Selection Operator

Since the population must be bred each generation, several individuals are chosen for recombination. In the state-of-the-art, one can find the following ones:

- *Random Selection*: consists in randomly selecting a configurable percentage of individuals for potentially recombining them.
- *Tournament selection* [(132)]: implies executing t tournaments among some randomly chosen individuals from the population. The individual who has the best fitness is selected for recombination. When t is larger, individuals with worse fitness have fewer chances to be selected (t indirectly determines the selective pressure).
- *Roulette-wheel selection* [(83)]: associates a probability of selection with each individual chromosome. The probability of selecting a chromosome is proportional to its fitness or rank (survival of the fittest).
- *Truncation selection* [(134)]: removes a predetermined percentage of the candidates with worst fitness.

- *Ordered selection* [(180)]: randomly picks a chromosome from the top N percent of the population.
- *Best*: merely selects the best chromosome in terms of fitness. When there are more than two chromosomes with the same fitness, one of them is randomly chosen.

5.2.3.2 Mating Operator

The purpose of this operator is to mate individuals (which individual should reproduce with another one). We can hit upon the following techniques:

- *Random mating* [(65)]: randomly mates individuals for posterior crossover.
- *Fitness-based mating* [(70)]: selects pairs of individuals with the highest difference in terms of fitness (best fitness individual will be mated with the worst fitness one). The idea is to potentially provide the EA with a fast diversity mechanism.
- *Similarity mating* [(93)]: selects pairs of individuals having more differences in terms of genes in their chromosomes. This mechanism provides real diversity to the EA but it is time-consuming.

5.2.3.3 Crossover

This operator combines individuals to produce several children (offspring). The key idea behind the recombination of individuals is to potentially obtain other better fitted individuals:

- *One-point crossover* [(76)]: chooses a random point on both parents' chromosomes (the same point for both parents). All the genes previous to this point are copied from one of the parents (randomly chosen) to one of the children (also randomly chosen). The genes beyond this point from the first father are arbitrarily copied in the other child and the ones from the second father are arbitrarily copied to the other child as Figure 5.5 illustrates.

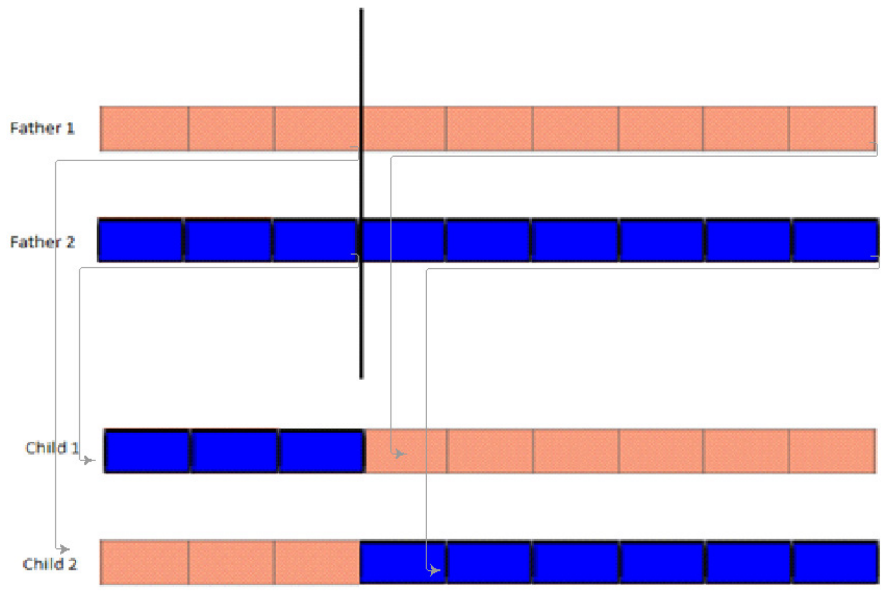

Figure 5.5: One-point crossover.

- *Multi-point crossover* [(160)]: selects N random points on both parents' chromosomes (the same points for both parents). Each piece of chromosome from the parents is alternatively copied to each child as Figure 5.6 shows.

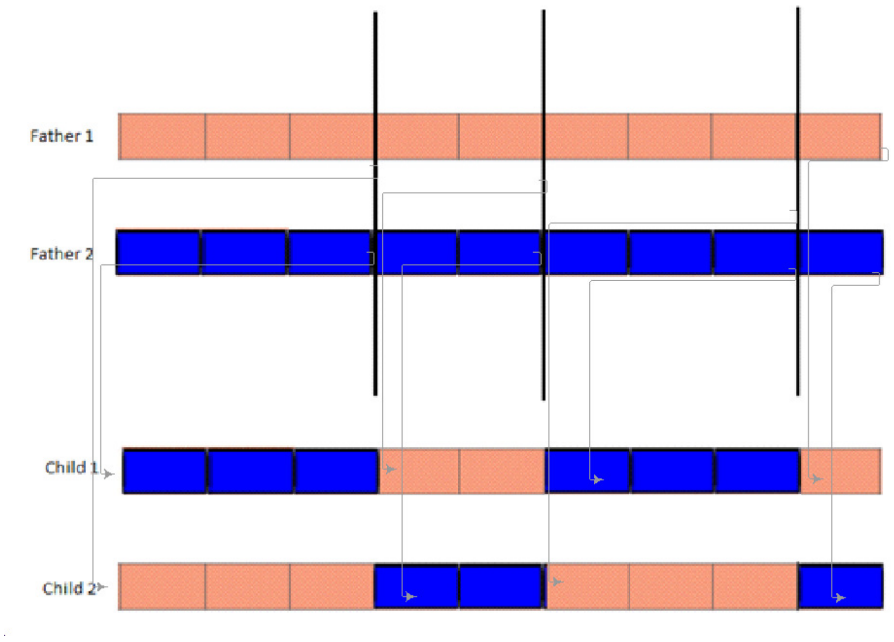

Figure 5.6: Multi-point crossover.

- *Cut-and-splice crossover* [(83)]: consists of selecting *2* different random points (one in each parent). One piece of father-*1*'s chromosome is then copied to a randomly chosen child. The same action is accomplished for father-*2* in the opposite child. The rest of genes are randomly copied (see Figure 5.7).

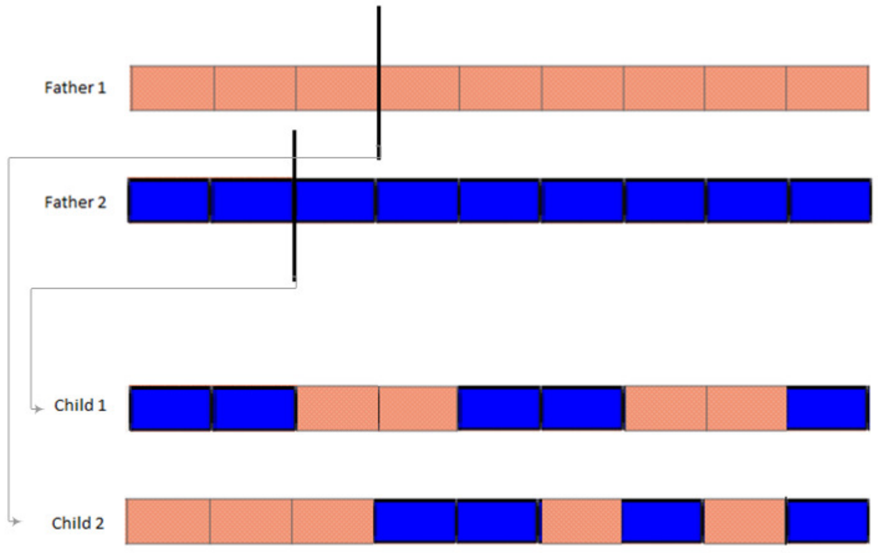

Figure 5.7: Cut-and-splice crossover.

- *Probability crossover:* considers that children will inherit the common points in their parents (potentially, the best genes) and randomly receive the rest of genes from them. This probability can be the *0.5* (uniform crossover [(162)]) or proportional to the fitness.

  We also propose to assign this probability in a more complex way such as simulated annealing does [(101)]. At the beginning of the process, when the temperature is higher, we can explore more by applying a probability of *0.5* and, when the temperature starts cooling off, we can give more probability to the best fitted individual as follows:

$$Pr_0(bestfitted) = 0.5(initialprobabilityforbestfittedparent).$$
$$Pr_i(bestfitted) = 0.5 + v/T_i(probabilityforthebestfittedparentatgenerationi). \quad (5.5)$$

where T_i is temperature at iteration i (the probability of giving a higher weight to the best fitted individual increases when the temperature decreases) and v is a factor to return values between *0* and *0.5*.

Simulated annealing has different schemes to decrease the temperature but they all decrease non-linearly. Another option is to increase the probability according to the number of generation generated as follows:

$$Pr_i(bestfitted) = 0.5 + v/G_i \tag{5.6}$$

where G_i stands for the generation number i and v is a factor to return values between *0* and *0.5*.

All in all, the idea is to choose a probability for recombination and we have several mechanisms to achieve this task as Figure 5.8 confirms.

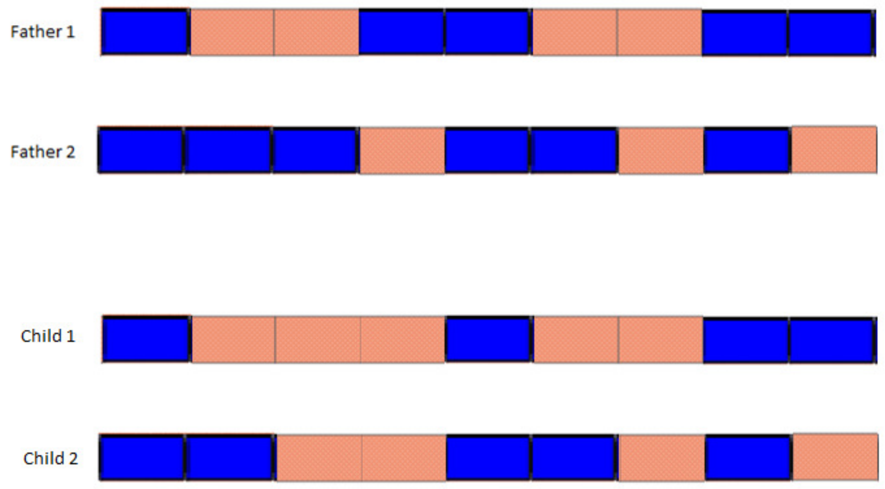

Figure 5.8: Probability crossover.

5.2.3.4 Mutation Operator

This operator causes tiny changes in the chromosome of individuals to explicitly maintain diversity. It applies a perturbation over each gene of the chromosome with a given probability. This perturbation corresponds to changes of profiles in some agents (e.g. agent a_2 who had assigned the *profile* P_1 has now associated the *profile* P_3 due to a mutation).

5.2.3.5 Replacement Policy

Finally, we decide which individuals are incorporated (or maybe reinserted) into the population for the next generation.

- *Generational* [(172)]: After recombination, the offspring generated by the selected parents fully replaces them. The selection strength is low when this scheme is applied (slow convergence). However, it potentially converges to the global optimum when enough generations are generated.
- *Steady-state* [(172)]: After crossover, the offspring generated by the selected parents may replace them if these are best fitted. There are numerous policies for individuals' replacement:
  - *Elitism* [(35)]: best fitted individuals fully replace the worst ones (quick convergence).
  - *Random replacement*: randomly chooses the individuals from the parents and children set. With certain probability, worst fitted individuals may replace the best ones.

  We also propose some *novel replacement schemes*:

- *Boltzmann criterion* [(101)]: The best fitted individual is chosen and another one (which may not be the second best fitted individual) is inserted with a given probability as simulated annealing does.
- *Similarity criterion*: we propose to select the best fitted individual and its most different one in terms of genes.
- *Taboo criterion*: we also propose another scheme, which lies in storing a list of non-promising individuals (based on their age for instance) in order to avoid inserting duplicated or inappropriate individuals. This option imposes additional memory requirements and more evaluations. Instead, we save up so much time in incorporating useless individuals during a given amount of time or iterations.

5.2.3.6 Refinement Method

MAs are quicker than traditional GAs for many problem domains because these apply an LS procedure. The present section describes the refinement method we propose.

LS is an MH for solving optimisation problems. An LS algorithm starts out from a candidate solution and, thus, iteratively moves to a neighbour solution, generating the neighbourhood. To carry out this action, a neighbourhood relation must be defined on the search space. In our case, we state that two candidate solutions are neighbours if only one gene differs in both chromosomes. Note that we propose a "simple" LS due to the lack of time of a dynamic environment but a more complex LS mechanism may be used when computing times are more flexible.

The following pseudo-code illustrates the LS algorithm:

```
void Local_Search (Chromosome & candidate_solution)
begin
    Chromosome best_solution = candidate_solution;
    Chromosome neighbour = candidate_solution;
    for $i \leftarrow 0$ to $candidate\_solution.size() - 1$ do
        Agent a = neighbour.getAgent(i);
        for $j \leftarrow 0$ to $a.get\_number\_profiles() - 1$ do
            neighbour.change_profile(i,j);//profile $j$ for agent $i$
            if $(neighbour.fitness() > best\_solution.fitness())$ then
                best_solution = neighbour;
            end
            neighbour = best_solution;
            j++;
        end
        candidate_solution = best_solution;
        i++;
    end
end
```

Algorithm 6: Basic LS pseudo-code.

Another relevant task is to decide the right frequency, which should be considered to apply the LS over the population and how many individuals must be affected. Chapter 6 will suggest an LS frequency and a percentage of affected individuals for the multi-skill call centre use case.

5.2.4 A Parallel Model

Roughly speaking, parallel implementations can be categorised into *coarse-grained* and *fine-grained* implementations (Cantú-Paz, 1998) [(31)]:

- *Coarse-grained*: these approaches maintain a population on each computing node where individuals are migrated according to a given policy.
- *Fine-grained*: these implementations keep an individual on each processing node, which operates as a neighbour for selection and reproduction.
- Other variants, like genetic algorithms for online optimisation problems, introduce *time-dependence* or *noise* in the fitness function.

Note that our approach can be encapsulated in the group of coarse-grained implementations. Specifically, we will employ an *island model* where there are multiple interconnected islands (see an example in Figure 5.9).

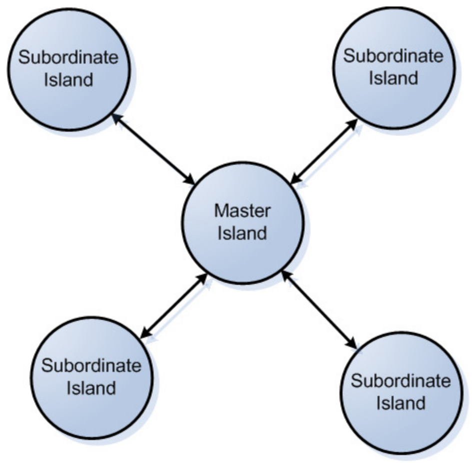

Figure 5.9: Example of the island model for *4* subordinate islands and *1* master island with a star topology.

It is not always straightforward to control the internal dynamics of a parallel Memetic Algorithm based on the island model, especially while seeking to ensure a fair balance between exploration and exploitation in a dynamic real-world environment. In real production environments, engineers do not always have enough time to test out and compute all the possible combinations to determine the optimal island connectivity configuration as there are many factors that may have an effect on the overall performance and accuracy (number of islands, topology, migration and replacement policies, amount of migrants, frequency of migrations, number of individuals in each island, type of synchronism, etc). This problem is even more severe when dealing with dynamic optimisation under uncertainty such as in the Multi-Skills Call Centre problem.

5.2.4.1 Background

Determining the right configuration for a parallel MA is not truly a new issue as many authors have already worked on finding an appropriate setting-up. In 1987, Pettey (1987) [(145)] put forward a distributed model in which the best-fitted individuals of each node were migrated to each neighbour node in each generation, fully replacing the worst-fitted individuals of those neighbours. Almost at the same time, Tanese (1987) [(163)] proposed a parallel implementation where each population was broken into a small number of subpopulations. Afterwards, each subpopulation was assigned to (and processed in) a different processing node within the system. The island model proposed in Cohoon (1987) [(39)] is an implementation of a distributed scheme where the idea of random migrant selection and replacement was put forward. In this proposal, each island was an isolated entity, which was capable of selecting individuals, crossing them and evaluating their fitness value.

After that, Gordon (1992) [(73)] as well as Adamidis (1994) [(1)] reinforced the term of island model in their parallel proposals, while Collins (1992) [(40)] launched a grid model where individuals were placed in a node and interacted with their neighbours. In 1995, some authors went into the migrants' selection in greater depth. This way, Belding (1995) [(19)] established an approach where the first n individuals were selected as migrants in relation to a predefined order.

Whitley (1997) [(177)] underlined that migration in parallel implementations caused additional selective pressure whereas Nowostawski (1999) [(141)] proposed a new taxonomy for PGAs based on a dynamic demes model. Special mention for Cantú-Paz (1998-2000) [(31)]-[(33)] who provided the most complete review of the state of the art on PGAs. Then, Alba (2001) [(3)] highlighted the importance of using asynchronous policies. In all the experiments conducted, asynchronous algorithms outperformed their equivalent synchronous counterparts in real time. Hu (2002) [(86)] described a model, which is inspired by the stratified competition frequently seen in society and biology. The proposal defined stratified levels with fitness value limits. Individuals moved from low-fitness subpopulations to higher-fitness subpopulations whether they surpassed the fitness-based admission threshold of the receiving subpopulation. Higher fitness levels implied higher selection pressure (exploitation).

More recently, Lozano (2008) [(120)] put forward an explicit measure of diversity, which entailed the replacement of existing individuals with lower values for the features being measured. The authors claimed to outperform existing replacement strategies presented in the literature, maintaining high levels of diversity. In contrast, Ruciński (2010) [(151)] examined the impact of the migration topology on an island model. This study compared different topologies and migration strategies in large networks. The authors concluded that the migration topology was a key factor to enhance the performance of a parallel global optimisation algorithm. Particularly, they recommended the use of ring topologies and suggested avoiding fully-connected and Barabasi-Albert topologies where a fast information spread over the entire network was allowed.

The most recent work on this topic, which went one step further, is Araujo (2011) [(10)]. The authors investigated, on a real parallel setup, a new strategy to enhance diversity in an island model. The proposal focused on the migrant selection phase of a genetic algorithm, taking into consideration the genotypic differences of the immigrant individual, which was incorporated in a receiving subpopulation.

5.2.4.2 Topology

As previously mentioned, we have employed an island model in which every processing node is a steady-state MA. The topologies we will analyse are the following ones (see Figure 5.10 for a better understanding):

- *Star topology*: We consider multiple subordinate islands, which correspond to simple steady-state MAs. These islands are connected to a master island (another simple steady-state MA), which coordinates and synchronises the rest of subordinate islands (see Figure 5.10.A).
- *Bidirectional ring topology*: each island sends and receives individuals from only other *2* islands, the previous one and the following one (see Figure 5.10.B).
- *All-to-all topology*: every island is fully connected to the rest of islands (see Figure 5.10.C).

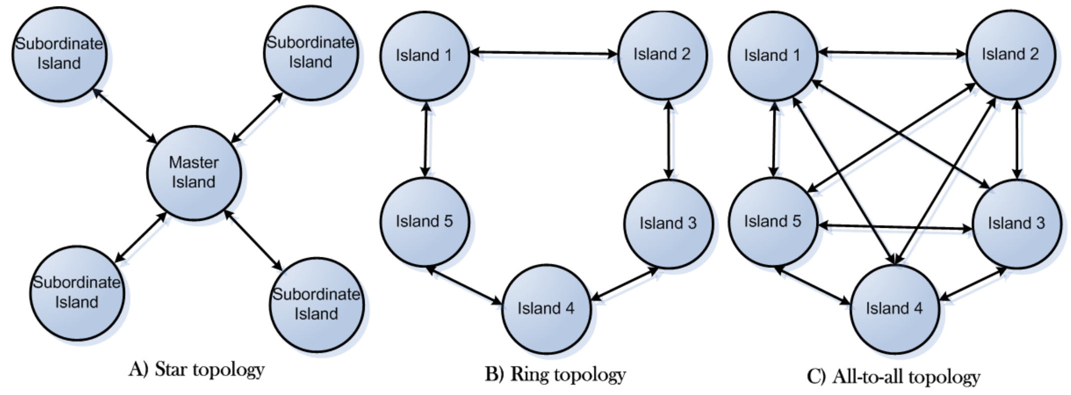

Figure 5.10: Topologies being analysed: A) Star Topology, B) Bidirectional Ring Topology, C) All-to-all Topology

5.2.4.3 Migration & Replacement Policies

This section proposes different policies to define what individuals should be transferred to the neighbouring islands and which ones should be replaced in the receiving populations. The combinations of policies are listed below:

- *Best-fitted individuals for worst-fitted individuals (BFI-WFI)*: The best-fitted individuals from the source population replace the worst-fitted individuals from the receiving population. We substitute individuals who are "further" in terms of fitness value to the source ones.

- *Best-fitted individuals for random individuals (BFI-RI)*: The best-fitted individuals from the source population randomly replace individuals from the receiving population.

- *Best-fitted individuals for best-fitted individuals (BFI-BFI)*: The best-fitted individuals from the source population replace the best-fitted individuals from the receiving population. We replace individuals who are "closer" in terms of fitness value to the source ones.

- *Best-fitted individuals for most different individuals (BFI-MDI)*: The best-fitted individuals from the source population replace the most different individuals (according to the number of different genes) to them, existing in the receiving population.

- *Best-fitted individual + "Annealing" for worst-fitted individuals (BFIA-WFI)*: We select the best-fitted individual from the source population and a set of the following best-fitted individuals with probability η, proportional to the number of generations spent (the more generations are executed, the higher the probability is). This proposal is inspired by the simulated annealing approach, which was first pioneered by Kirkpatrick (1983) [(101)]. This way, the probability of not choosing the best-fitted individuals for migration is $(1 - \eta)$ (in this case, we randomly select another individual from the source population). Afterwards, the best-fitted individual from the source population and the set of "annealing" individuals replace the worst-fitted individuals from the receiving population. Bear in mind that we always select the best-fitted individual of the source population to ensure a minimum of fast convergence as dynamic environments require prompt responses while also preserving diversity. Finally, we substitute individuals who are "further" in terms of fitness to the source ones.

Chapter 6

Evaluation of the Application

This chapter evaluates our approach on the MSCC domain. Section 6.1 highlights the magnitude, in terms of volume, of our application domain. Section 6.2 describes the dataset employed. Section 6.3 points out the hardware descriptions of the SunFire sever in which the evaluations have been performed. Section 6.4 analyses the selected metrics for testing and comparing our approach. Section 6.5 examines the forecast module for five different CGs as there are too many CGs to accomplish an exhaustive study for all of them. Section 6.6 evaluates the search module by studying several time intervals from days with different complexity; this section also compares our search module with other acknowledged techniques. In Section 6.7, we will analyse our complete approach (forecast module + search module) for one-day campaign. We will also compare how our complete approach outperforms other conventional call centre's algorithms.

6.1 Magnitude of our Call Centre

This section presents some numbers in order to expose the magnitude of our CC. In table 6.1, we can appreciate that our environment is quite complex and very dynamic.

Measure	**Value**
Maximum number of CGs	1,035
Maximum number of simultaneous incoming calls	2,500
Maximum number of incoming calls per hour	60,000
Maximum number of incoming calls per day	700,000
Minimum number of simultaneous agents	0
Maximum number of simultaneous agents	2,100
Minimum number of agents concurrently assigned to a single group	0
Maximum number of agents concurrently assigned to a sole group	526
Mean number of agents concurrently assigned to a single group	3
Minimum number of potential profiles per agent	1
Maximum number of potential profiles per agent	108
Mean number of potential profiles per agent	16

Table 6.1: Figures that reflect the magnitude of our call centre.

Obviously, the number of incoming calls is not the same all the time as it depends upon many factors. When agglomerating many data and considering a coarse-grain, forecasting becomes much easier as the variability at high level (e.g. monthly and daily level) is reduced and thus easy to forecast. However, our predictions rely on a fine-grain process as forecasts refer to each successive state (remind that time-frames can range from *60* to *3,000* seconds). We can perceive that fact in the following figures (Figures from Figure 6.1 to Figure 7.5).

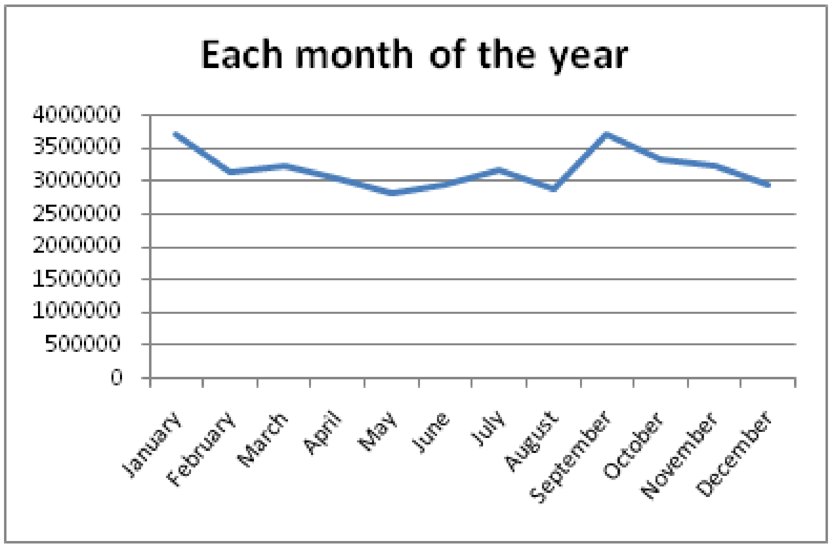

Figure 6.1: Incoming calls during a year at monthly level.

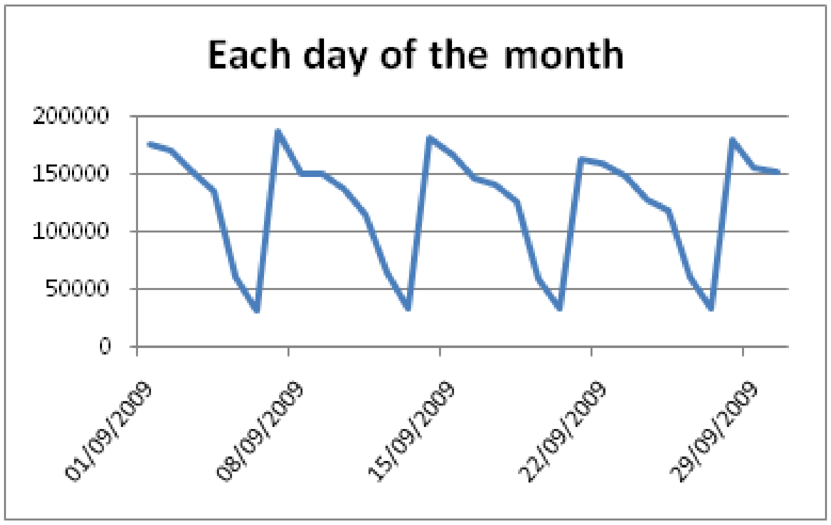

Figure 6.2: Incoming calls during the most intricate month (September) at daily level.

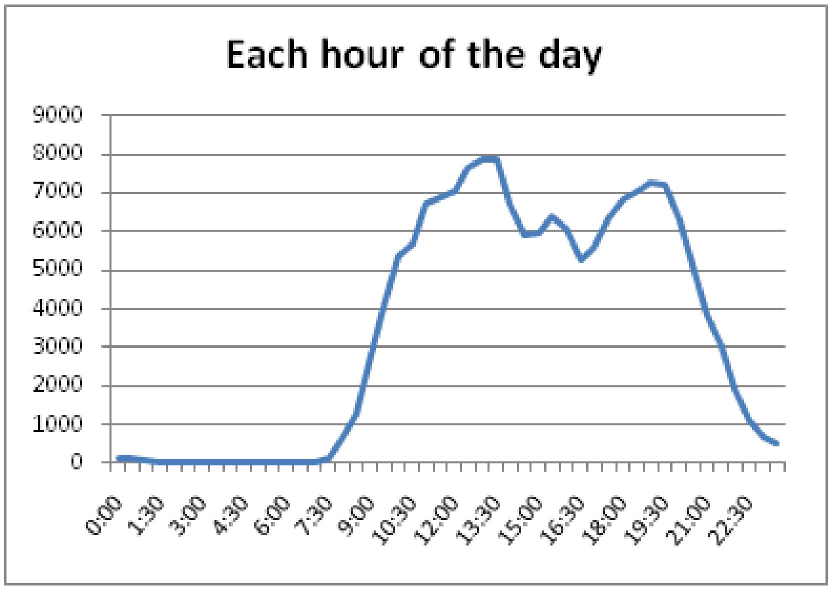

Figure 6.3: Incoming calls during the most complex day of September (September 9) at hourly level.

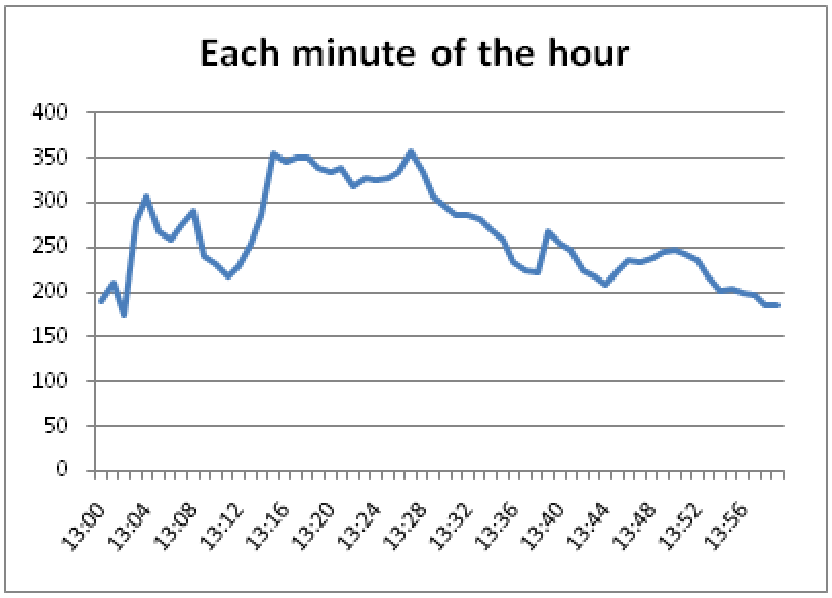

Figure 6.4: Incoming calls during the most complex hour of “September 9” at minutely level.

A pattern can be relatively easily found at macroscopic level (month of the year and day of the month). However, predictions are harder when considering the hour of the day and much harder at minute level (our concern in this work). Thus, these figures corroborate how complex is to predict the incoming flow in our environment. However, drop calls and available agents are easier to forecast because:

- the drop call rate is highly correlated with the volume of incoming calls as Figure 6.5 illustrates,
- and the number of available agents can be inferred from timetables and mean processing times as well as current load and other well-known factors. Figure 6.6 shows the volume of existing agents by CG.

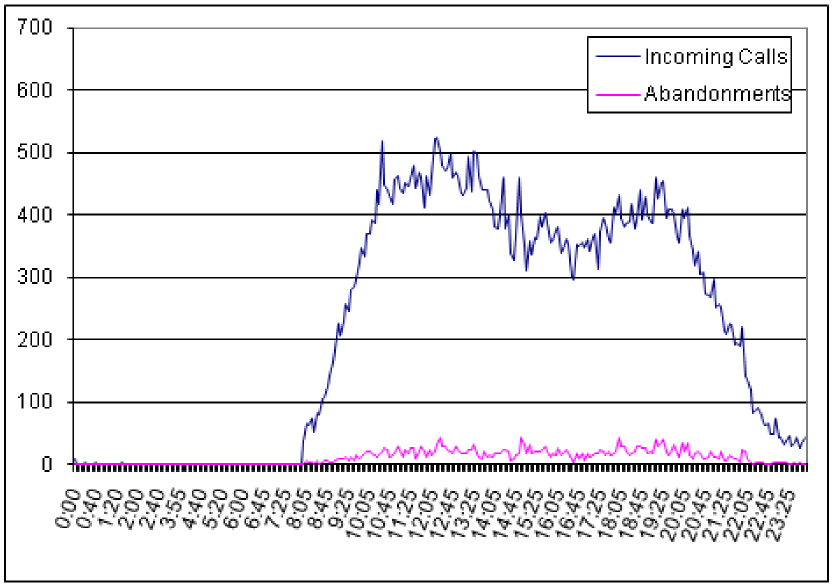

Figure 6.5: Incoming calls and abandonments (drop calls) during a common day.

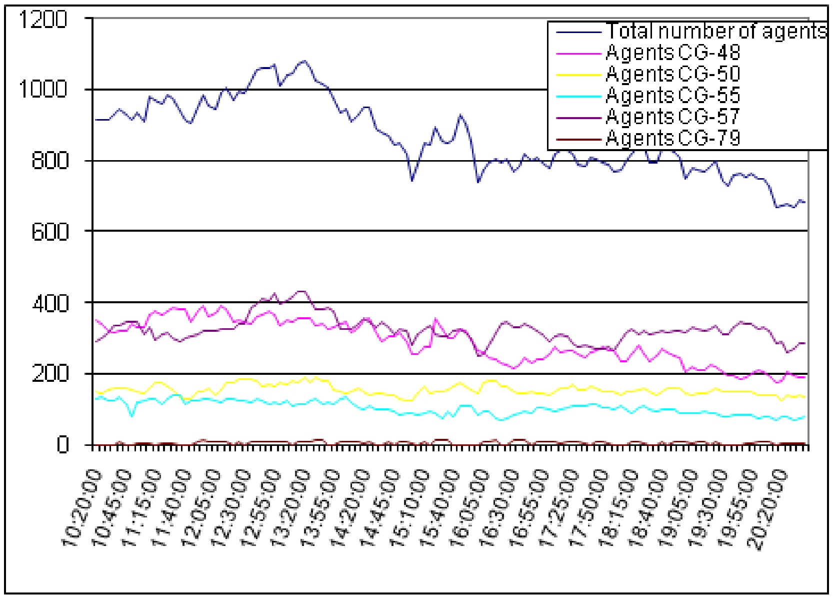

Figure 6.6: Number of agents for the *5* most representative CGs.

6.2 Dataset Employed

A key thing is to create a suitable dataset, hunting for a fair balance between the amount of data and a representative period of time measured in terms of days, carved up in hours and minutes (microscopic level ⇒ fine grain). In our case, our forecasting module will work with *122* numeric attributes (divided in *45* components, see Section 5.1) and thousands of registers, which correspond to *1*-minute interval information from records stored during several months. The frequency of predictions depends upon the size of the time-frame, determined by the adaptive time-frame mechanism. The search module will work with the predicted output generated by the forecasting module in the previous step, so that there will be as many inputs as time-frames.

In the dataset, the number of selected days must be a multiple of *7* because the predictor *week-day* has imperative influence on the training and validation processes as Section 5.1 demonstrates. Moreover, the number of days must be large enough to represent every possible pattern (cases). Therefore, the number of days to take into account should be, at least, *91* days in order to cover all possible patterns with the aforesaid considerations. It is very important to divide data like this as this composition allows for trend and seasonality detection.

Our problem presents *1,035* CGs; hence, the dataset is too large to do an exhaustive study for all of them (obviously, the forecast module has been trained, considering every CG). Consequently, *5* representative CGs with different behaviour in terms of oscillations, arrival rates, processing times and nature, have been carefully picked in order to perform a generic enough approach (see Figure 6.7). These oscillations intuitively imply a higher complexity. We have assigned an index to each CG that designates their complexity level, which ranges from the most convoluted CG to the simplest one (labelled from *1* to *5*). For business reasons, we are not allowed to reveal real CGs names but this is something we should not be concerned about in this work.

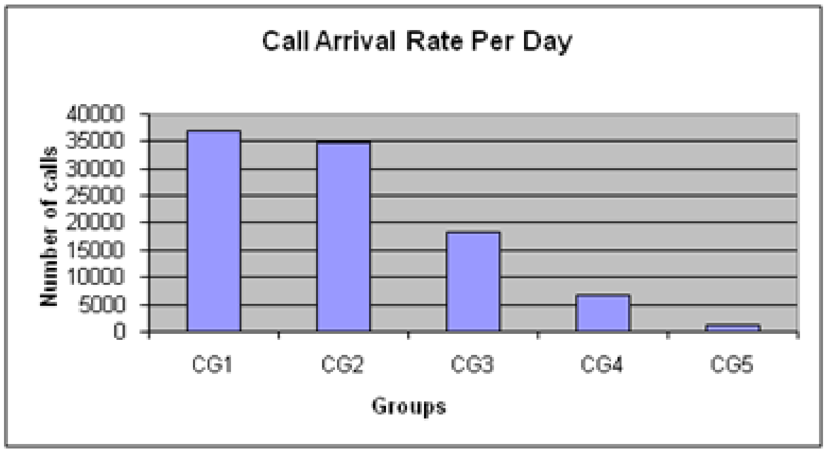

Figure 6.7: Call arrival rate per day, grouped by CGs.

Afterwards, a different model has been exclusively developed for each CG because of differences among CGs. Then, the whole dataset has been split into subsets, contemplating every CG.

Once we have a single dataset for each CG, this is shuffled and then randomly divided into three subsets, following the cross-validation structure [(104)] (see Figure 6.8): training (*55%*), generalisation (*20%*) and validation (*25%*).

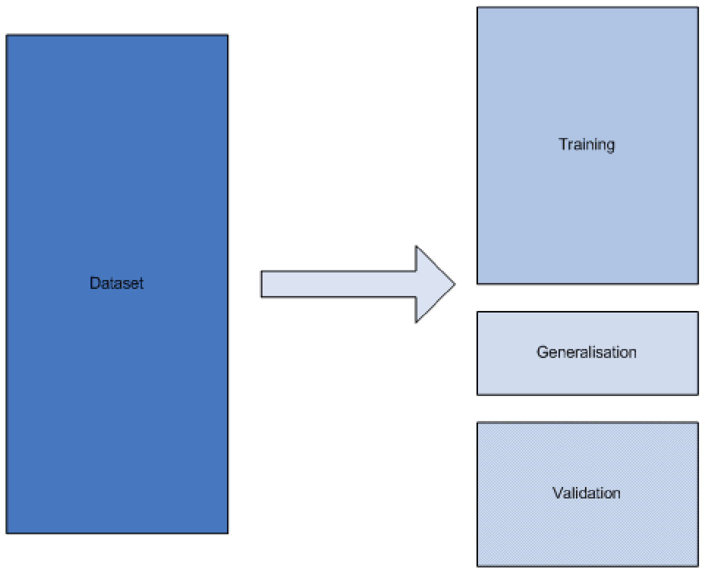

Figure 6.8: Dataset partitions: training, generalisation and validation.

The training dataset, which is the largest partition, is used for training our ANN. Instead, the generalisation partition is used at the end of each epoch to observe whether (or not) our ANN correctly handles unseen data. Once the training process has finished, the validation partition is showed to our ANN to determine its real precision and accuracy.

Although we will analyse *5* CGs for the forecasting module in Section 6.5, we will validate the search module, considering all the CGs as the fitness function needs all the information. To evaluate the search module, we have chosen several time intervals from days with different complexity (see Section 6.6). Therefore, we can discover the benefits of our approach, depending on the dynamism of the system. Besides, we will analyse our complete approach (forecasting module + search module) for one-day campaign in Section 6.6.

6.3 Hardware Description

This section describes the key hardware features of our *SunFire 4,900* server in which all experiments have been launched. These features are the following ones:

- *64*-bit Chip Multithreading UltraSPARC IV technology, with over *2x* the throughput of previous generations.
- Scales up to *8* x *1.35*-GHz UltraSPARC IV CPUs with *16* MB L2 cache per processor.
- Up to *16* simultaneous compute threads with up to *64* GB memory.
- Solaris-TM 8, Solaris 9, and Solaris 10 Operating System.
- Robust capabilities in the Solaris 10 OS such as predictive self-healing to increase reliability, Solaris containers for increased utilisation, and dTrace to optimise application performance.
- *9.6* GB/second SunTM Fireplane interconnect.
- N+1 hot-swap power supplies/hot-pluggable disks.
- Sun systems controller for remote system administration.
- Automatic system recovery to maximise uptime.
- Integrated fibre channel disk subsystem, multi-pathing-ready, supporting up to 12 FC-AL disks.
- 9 PCI slots help ensure a highly scalable, well-balanced system.
- 17-RU tower/desk-side, rack mountable.

6.4 Metrics

Metrics are usually specific for a given subject area and are often valid only within a certain domain, so that these cannot be directly interpreted outside it. We have selected several metrics to evaluate our forecasting and search modules as well as the complete process.

6.4.1 Forecasting Metrics

In order to make the forecast process more understandable, we define the error, e, as the difference between the real output value, f, and the predicted output, y. To evaluate the forecasting techniques, we will apply the following metrics:

- *Mean Absolute Error* (MAE) - average of the absolute errors: $MAE = \frac{1}{n}\sum_{i=1}^{n} |f_i - y_i|$

- *Standard Deviation* (SD) - the standard deviation of a statistical population is the square root of its variance: $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (f_i - y_i)^2}$

- *Range statistics* - minimum absolute error and maximum absolute error.

- *Pearson's correlation* - which is defined as the covariance of the two variables divided by the product of their standard deviations: $Pearson_{f_i,y_i} = \frac{covariance(f_i,y_i)}{\sigma_{f_i} \cdot \sigma_{y_i}}$

- *Analysis of variance* (ANOVA) - collection of statistical models, and their associated procedures, in which the observed variance in a particular variable is partitioned into components attributable to different sources of variation.

6.4.2 Search Metrics

In order to compare all the search algorithms in terms of quality of the solution, a metric to represent that quality is required. We presume that quality comparisons must be made over the same problem instances. Comparisons over different problem instances are normally weaker as those instances may have dissimilar structures so that the conclusions might be completely erroneous.

To evaluate the search module, we will examine the following metrics:

- *Worst solution*: Less fitted value, considering e executions (*50*).
- *Best solution*: Best fitted value, considering e executions (*50*).
- *Mean solution*: Mean value from e executions (*50*).
- *Standard deviation*: Standard deviation from e executions (*50*).
- *Performance*: Ratio of the current fitness value with respect to the best fitted value. It can be calculated as follows: $Performance = \frac{current\_fitness\_value(technique_i)}{best\_fitted\_value}$

6.4.3 Call Centre Metrics

Most MSCCs employ more than *30* different metrics to verify how operations are going. However, sometimes, just observing a subset of variables may accomplish our goals. Metrics refer to customer satisfaction, quality, productivity, agent utilisation or costs per contact (for outbound MSCC).

To evaluate the complete approach, we will consider the most important metric, from our point of view, for any MSCC: the *service level*, which is defined as the percentage of customer calls that have to queue shorter than a specified amount of time (in our MSCC, *20* seconds): $Service\_Level = \frac{answered\_calls\_in\_less\_than\_20\_seconds}{total\_number\_of\_incoming\_calls}$

This metric covers aspects such as quality, productivity, client satisfaction among others.

6.5 Comparison of Forecasting Techniques

In order to evaluate the accuracy of our approach to the prediction of incoming calls, as compared to the predictive techniques described above, a series of experiments has been conducted on the MSCC domain.

The techniques to be analysed and then compared with our approach (uRprop) have been selected from the survey of related work from Section 3.2.1: Single Exponential Smoothing, Simple Time Series, Stationary Time Series, Damped Trend Time Series, several ARIMA Models, Regression Models, Logistic Regression and Artificial Neural Networks (*7* different learning algorithms).

To perform the throughout comparison, we have used R's forecast package [(147)] and R's RSNNS package [(150)] to evaluate those forecast techniques. Note that we have selected those packages because they are reliable, well-implemented open-sources so that we can analyse the inner of the implementations. Other data mining tools such as SPSS Clementine or SAS Enterprise Miner have also truly powerful algorithms, but we have not applied them to this comparison as we would not have any insights about the implementation of the algorithms behind them.

In order to avoid extracting biased conclusions, we have executed the models *50* times and averaged the results. The models have been run under two of the cores of our SunFire 4,900 server (one processor for the forecast module and another one for the interfaces and data pre-processing).

Obviously, the number of incoming calls is not the same all the time since it depends upon many factors (commercial campaigns, load peaks, daytime, etc.). When agglomerating much data, forecasting becomes much easier as the variability of call arrivals at high level (e.g. monthly and daily level) is reduced and more or less easy to forecast (coarse-grain forecasting). However, our predictions rely on a fine-grain process as forecasts refer to each successive time point over a rolling prediction window.

Now, we present the results achieved during the experimental phase. Figure 6.9 illustrates the cross-validation MAE comparison between Single Exponential Smoothing, Simple Time Series, Stationary Time Series, Damped Trend Time Series, two ARIMA models, Linear Regression, Logistic Regression and our uRprop ANN. This confirms that, although each CG has a different behaviour, our ANN regularly behaves better than other conventional predictive techniques (we do not claim a universally better approach). Tables 7.2 - 7.6 better reflect the performance of each predictive technique since Figure 6.9 shows CGs with very different MAE levels and the differences among them may seem less important than what they actually are.

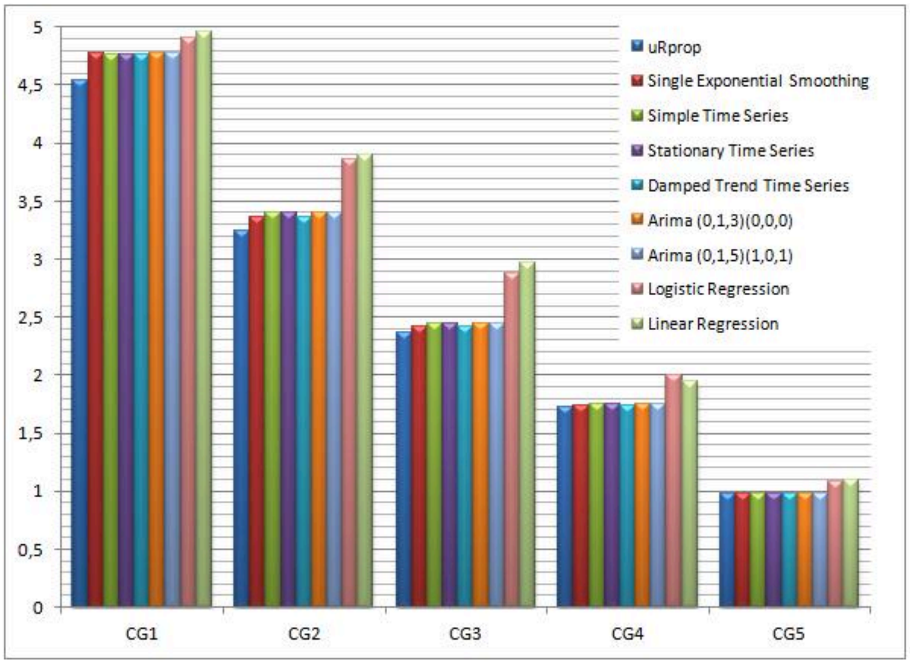

Figure 6.9: General forecasting comparison (cross-validation MAE is represented in the Y-axis while the results obtained by each technique for each CG are given in the X-axis). A lower MAE implies a better performance.

Tables 7.2 - 7.6 gather the results of the comparisons performed for each CG.

These tables contain the mean values from the *50* executions we have launched for each predictive technique with many thousands of occurrences (records from three months). These tables also include further information such as minimum error, maximum error, mean absolute error, standard deviation and Pearson correlation. Note that, although differences in the results may sometimes seem small, those apparently small errors (sometimes there are differences in the first or second decimal) are accumulated and can entail annual losses of thousands Euros. The explanation is that there are *1,035* CGs, up to *700,000* incoming calls in a single day, so that small differences scale up very fast.

Method	Min. Error	Max. Error	MAE	SD	Correlation
uRprop	*-76*	*123*	*4.533*	*8.288*	*0.937*
Single Exponential Smoothing	-150	127	4.777	8.470	0.934
Simple Time Series	-160	129	4.767	8.508	0.934
Stationary Time Series	-161	129	4.767	8.508	0.934
Damped Trend Time Series	-161	128	4.768	8.509	0.934
Arima (0.1.3)(0.0.0)	-154	129	4.779	8.489	0.934
Arima (0.1.5)(1.0.1)	-153	128	4.781	8.477	0.934
Backprop	-123	128	4.606	8.221	0.938
BackpropMomentum	-90	130	4.58	8.219	0.938
Rprop	-105	127	4.619	8.165	0.939
Quickprop	-99	128	4.639	8.254	0.938
BackpropChunk	-90	132	4.579	8.213	0.938
RBF	-68	159	4.832	10.698	0.892
BackpropWeightDecay	-95	127	4.574	8.314	0.938
Logistic Regression	-118	120	4.905	8.471	0.920
Linear Regression	-127	119	4.956	8.499	0.918

Table 6.2: Results for Call Group 1: Method, minimum error, maximum error, mean absolute error, standard deviation and Pearson correlation.

Method	Min. Error	Max. Error	MAE	SD	Correlation
uRprop	*-32*	*36*	*3.244*	*5.360*	*0.968*
Single Exponential Smoothing	-40	39	3.356	5.507	0.967
Simple Time Series	-36	39	3.405	5.567	0.966
Stationary Time Series	-36	39	3.406	5.509	0.966
Damped Trend Time Series	-40	39	3.356	5.507	0.967
Arima (0.1.3)(0.0.0)	-36	39	3.402	5.564	0.966
Arima (0.1.5)(1.0.1)	-37	39	3.397	5.554	0.966
Backprop	-35	39	3.259	5.432	0.968
BackpropMomentum	-37	37	3.257	5.344	0.968
Rprop	-32	37	3.273	5.339	0.968
Quickprop	-33	36	3.319	5.39	0.968
BackpropChunk	-37	37	3.265	5.333	0.968
RBF	-40	136	3.951	6.403	0.947
BackpropWeightDecay	-34	37	3.255	5.385	0.968
Logistic Regression	-32	39	3.853	5.958	0.961
Linear Regression	-32	40	3.899	6.106	0.952

Table 6.3: Results for Call Group 2: Method, minimum error, maximum error, mean absolute error, standard deviation and Pearson correlation.

Method	Min. Error	Max. Error	MAE	SD	Correlation
uRprop	*-19*	*29*	*2.368*	*3.960*	*0.948*
Single Exponential Smoothing	-24	29	2.417	4.036	0.946
Simple Time Series	-24	30	2.445	4.062	0.946
Stationary Time Series	-24	30	2.445	4.063	0.945
Damped Trend Time Series	-24	29	2.417	4.036	0.946
Arima (0.1.3)(0.0.0)	-24	30	2.444	4.060	0.946
Arima (0.1.5)(1.0.1)	-25	29	2.44	4.052	0.946
Backprop	-20	29	2.436	4.002	0.947
BackpropMomentum	-20	29	2.394	3.963	0.948
Rprop	-21	28	2.403	3.948	0.948
Quickprop	-19	28	2.457	4.038	0.946
BackpropChunk	-22	28	2.419	2.985	0.947
RBF	-24	45	3.17	6.538	0.851
BackpropWeightDecay	-20	29	2.374	3.958	0.948
Logistic Regression	-33	28	2.880	4.580	0.942
Linear Regression	-32	28	2.962	4.624	0.943

Table 6.4: Results for Call Group 3: Method, minimum error, maximum error, mean absolute error, standard deviation and Pearson correlation.

Method	Min. Error	Max. Error	MAE	SD	Correlation
uRprop	*-20*	*20*	*1.728*	*3.129*	*0.948*
Single Exponential Smoothing	-22	21	1.732	3.183	0.947
Simple Time Series	-22	21	1.75	3.208	0.946
Stationary Time Series	-22	22	1.749	3.203	0.946
Damped Trend Time Series	-22	21	1.732	3.183	0.947
Arima (0.1.3)(0.0.0)	-22	21	1.748	3.203	0.946
Arima (0.1.5)(1.0.1)	-22	21	1.744	3.198	0.946
Backprop	-20	20	1.836	3.178	0.947
BackpropMomentum	-20	19	1.766	3.122	0.948
Rprop	-21	19	1.761	3.116	0.949
Quickprop	-19	21	1.883	3.217	0.945
BackpropChunk	-21	20	1.856	3.195	0.946
RBF	-21	34	2.256	5.111	0.855
BackpropWeightDecay	-21	20	1.778	3.132	0.948
Logistic Regression	-21	29	1.993	3.239	0.930
Linear Regression	-20	30	1.949	2.631	0.928

Table 6.5: Results for Call Group 4: Method, minimum error, maximum error, mean absolute error, standard deviation and Pearson correlation.

Method	Min. Error	Max. Error	MAE	SD	Correlation
uRprop	*-10*	*13*	*0.976*	*1.353*	*0.777*
Single Exponential Smoothing	-12	18	0.981	1.804	0.602
Simple Time Series	-12	18	0.981	1.804	0.602
Stationary Time Series	-12	18	0.981	1.803	0.603
Damped Trend Time Series	-12	18	0.981	1.804	0.603
Arima (0.1.3)(0.0.0)	-12	18	0.981	1.804	0.603
Arima (0.1.5)(1.0.1)	-12	18	0.981	1.803	0.603
Backprop	-5	17	0.948	1.694	0.617
BackpropMomentum	-5	17	0.986	1.741	0.588
Rprop	-5	18	0.778	1.762	0.574
Quickprop	-4	18	1.079	1.802	0.548
BackpropChunk	-5	17	0.967	1.726	0.598
RBF	-5	18	1.002	1.773	0.567
BackpropWeightDecay	-5	16	0.966	1.688	0.620
Logistic Regression	-8	13	1.084	1.798	0.748
Linear Regression	-8	13	1.1	1.823	0.745

Table 6.6: Results for Call Group 5: Method, minimum error, maximum error, mean absolute error, standard deviation and Pearson correlation.

Looking at Tables 7.2 - 7.6, we perceive that uRprop (our approach) has less variability in the predictions as the standard deviation is lower for every CG. However, we observe that Time Series and ARIMA only obtain a slightly greater MAE for those CGs with very few incoming calls (CG4 & CG5). When the volume of incoming calls is reduced, Time Series are worth considering as they are very fast to train and detect trends very quickly.

While ARIMA and Time Series emphasise the "recent past", our ANN is more flexible because it does not only consider previous tendencies and time points but also covers historical patterns from other days and contextual information (e.g. if there is a commercial campaign, the oscillations of the inbound traffic will be higher). The capability of considering historical data is really valuable as we can discover trending features like the peak hours' effect and even more complex relationships. We can also observe that regression models do not perform accurately as compared to time series on this dynamic domain.

Another remarkable result is that Time Series and ARIMA models outperform Regression Models in most CGs, probably, due to the capability of considering trends and seasonality rather than simply considering relationships between the dependent variable (output) and any one of the independent variables when these vary. The main advantage of Regression Models is the clearness to understand and track the model although it is hard to choose the variables to generate the model considering seasonality and trend, which are crucial to better understand the behaviour of a CC.

Single Exponential Smoothing, Simple Time Series, Stationary Time Series and Damped Trend Time Series require short computing times. Nevertheless, their performance is limited because a single coefficient α is not enough to perfectly fit the outcome. ARIMA models sometimes outperform the Time Series being studied but ARIMA models require longer computing times.

On the one hand, the main problem with our approach is that we need longer training times than ARIMA, Time Series or Regression Models. Besides, our implementation is much more complex than these techniques. When not having a multimodal problem, we recommend the use of simpler techniques: Linear Regression for linear dependencies and Times Series for stable patterns or those that merely depend upon the recent past.

On the other hand, although there are no huge differences between our approach and the rest of the above-mentioned techniques in terms of MAE for a given instant, we strongly recommend our approach as far as slight differences in terms of MAE for a given point may induce huge mean errors throughout a day (cascade effect).

We can also notice that the minimum and maximum errors are sometimes quite big. Those errors occur at the beginning of the day when people start calling to the call centre and the system becomes very active in very few seconds (e.g. at 9 p.m.). Fortunately, the system gets more or less stable very quickly and those errors decrease a lot.

Up until now, we have proved that our approach outperforms several forecasting techniques but, can we beat other ANN's learning algorithms? Our approach sometimes gets trapped in local minima so that we cannot claim that our approach can behave better than other learning algorithms for ANNs as there are other nice exact approximations. However, our environment is very dynamic and complex, and we have *1,035* CGs with very different behaviour. So, can we outperform other learning algorithms given our requirements? Luckily, the answer is "yes". At least, we can offer more accurate results for those CGs, which are more dynamic and have more incoming calls. This makes sense because the search space is consequently more complex. Nonetheless, other learning algorithms might behave better for those CGs with fewer calls (e.g. CG5). Obviously, this type of CGs is not really relevant due to the low volume of call arrivals (it is less critical to have deviations).

Figure 6.10 demonstrates that our learning algorithm outperforms Backpropagation, Backprop with Momentum, Quickpropagation, Rprop with weights backtracking, Backpropagation with weight decays, Backpropagation with chunkwise updating of the weights and Radial Basis Function for the most convoluted CGs (CG1 and CG2).

For the easiest ones (CG4 and CG5), learning algorithms like Rprop with weights backtracking slightly outperform our uRprop.

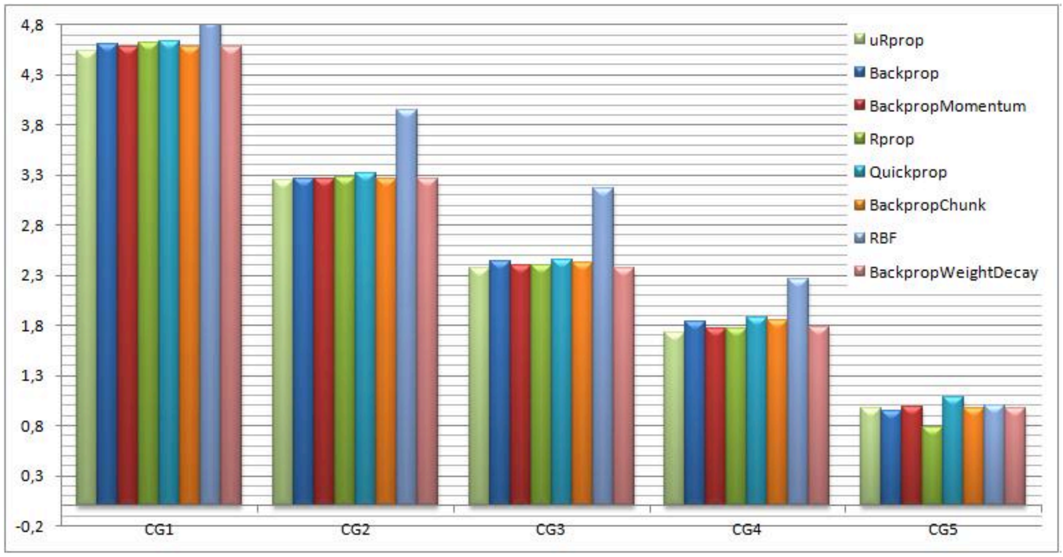

Figure 6.10: ANN Comparison. Cross-validation MAE is in the Y-axis and the results obtained by each ANN learning algorithm for each CG are in the X-axis. A lower MAE implies a better performance.

Finally, in order to compare the means of the samples, we will apply the ANOVA test. When we have only two samples, we can use the t-test to compare the means of the samples but it might become unreliable in case of more than two samples.

ANOVA compares several unmatched groups, based on the assumption that the populations are Gaussian. The P value verifies whether the populations really have the same mean. ANOVA is based on the assumption that all the populations have the same variance. When the P value is small, we may conclude that the variances of the populations are different in each experiment. As a consequence, the populations are different. If the variances are truly different, then the populations are different regardless of what ANOVA concludes about differences among the means.

Table 6.7 shows the results for the ANOVA test. According to the results gathered in Table 6.7, we can state that the results and experiments are relevant as we see that the chosen CGs and selected forecasting techniques are truly different and representative.

Anova	Df	Sum Sq	Mean Sq	F value	Pr(>F)
ind	4	131.774	32.944	1100.8	$<$ 2.2e-16
Residuals	75	2.245	0.030		

Table 6.7: ANOVA test

6.6 Experiments with the Search Module

This section provides an exhaustive analysis of the search module. In Subsection 6.6.1, we present the experiments we have performed with the simple and parallel memetic algorithm in order to determine the right configuration of the search module. In Subsection 6.6.2, we compare our simple and parallel approaches with other famous MHs.

6.6.1 Analysing the Search Module

6.6.1.1 The Simple Memetic Algorithm

The configuration of each island (a simple MA) has so much relevance in the global performance of the parallel MA. For this reason, we now present *8* key operator configurations in order to understand how a good (or bad) tuning can make an impact on the final outcome. A given configuration can entail a faster or slower convergence and a deeper or lighter exploitation of the search space, thereby finding a better or worse optimum for a given elapsed time.

We have employed a dataset extracted from a normal day. Every configuration has been run over the same problem instance *30* times and we have displayed the mean fitness values.

The configurations that we have been working with are (see Figure 6.11):

- *Configuration-1* employs a binary tournament selection and considers that children will inherit the common points in their parents (potentially, the best genes) and randomly receive the rest of genes from them. Then, it applies a perturbation over each gene of the chromosome with a probability of *0.03.* This perturbation corresponds to changes of profiles in some agents (e.g. agent a_2 who had assigned the profile P_1 has now associated the profile P_3 due to a mutation). Finally, it considers elitism with a probability of *0.93* to replace the worst individuals of the population for next generation. And, with a probability of *0.07*, a worse individual may be captured (steady-state scheme).

From this configuration, we perceive that Configuration-1 can process many generations compared to the rest of configurations (excluding Configuration-8). Configuration-1 also obtains the best fitted solution. The convergence is favourable for a *5*-minute execution (around generation-300). Note that most time-intervals have that duration so that the improvement during a complete day is noticeable. After that point the improvement is minor, albeit we can observe another important slope around generation-500. When the dynamism is high, this configuration is also very appealing because this configuration steeply slopes. Besides, when the time-frame increases, the configuration is also appropriate as it still goes on improving the fitness value.

- *Configuration-2* differs from Configuration-1 in the mating-selection as it considers mating by similarity. For this reason, the number of generations is reduced. This configuration allows for diversity but the convergence is slower than Configuration-1. Instead, Configuration-2 almost always improves and may be good for stable systems (longer time-frame).

- *Configuration-3* applies a mating based on the differences on the fitness values. Each individual is mated with its most different individual in terms of fitness: highest difference in fitness value. We notice that this mating operator is faster but the results seem to be worse than Configuration-2.

- *Configuration-4* applies a random mating and a special selection in which the best fitted individual is taken as well as its most different individual in terms of genes (like similarity but for final selection). This configuration always increases and has a nice slope, although the two first configurations seem to perform quite better.

- *Configuration-5* imitates Configuration-4 except for the mating mechanism. In this case, each individual is mated with its most different individual in terms of fitness. It has a poorer performance and can carry out fewer generations.
- *Configuration-6* employs a mating by highest fitness difference and applies a replacement policy in which the best fitted individual and the worst fitted individual after reproduction fully replace the best and the worst individuals from the populations respectively. The performance and slopes are poor.
- *Configuration-7* proposes a similar scheme to Configuration-6 except for the random mating. More generations are carried out and better performance than Configuration-6.
- *Configuration-8* applies a random mating and our crossover inspired in simulated annealing. We also consider elitism for the replacement policy and mutations by ranges. We also perceive a fine slope at the beginning. If we had more time, we might increase the initial temperature value to explore more at the beginning. Potentially, we could obtain better fitted individuals.

Figure 6.11 shows some of the most relevant configurations that we have tested out during *600* seconds (*10* minutes). X-axis represents the fitness value while the Y-axis stands for the number of generations.

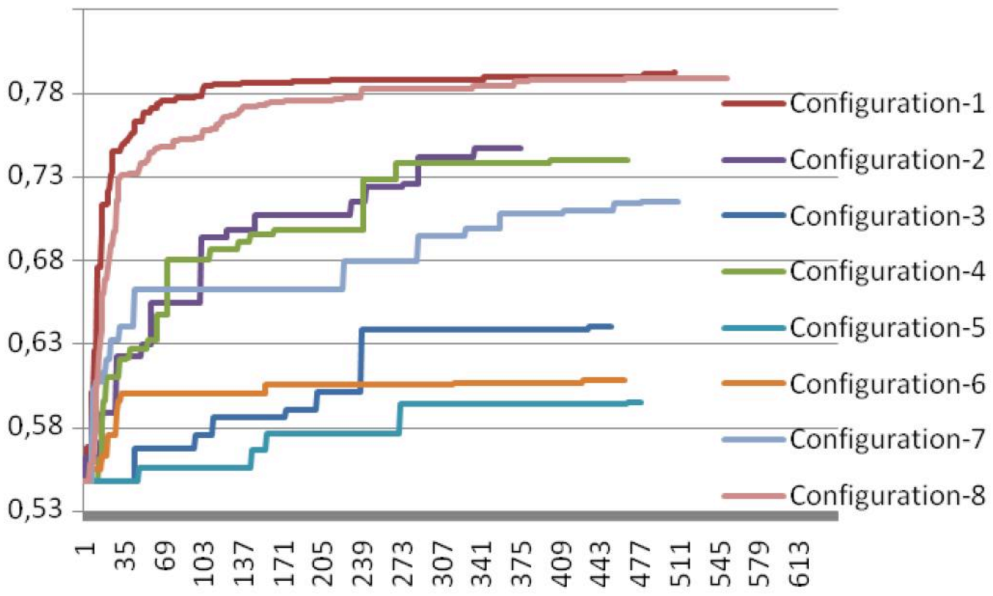

Figure 6.11: Fitness obtained for *8* different configurations of evolutionary operators. Y-axis is the fitness value while X-axis represents the iterations. A higher fitness value implies a better performance.

Nevertheless, a manual batch of experiments may take too much time and might guide the results towards a preconceived idea about the right setting-up. In order to automate the experiments for the single MA, we have developed a meta-GA, using historical data. Another option would be to develop a script to test out different combinations but it would take too much time to perform the complete combinatorial. For this reason, we believe that a meta-GA can be of valuable help in the decision making process.

The chromosome of the meta-GA has *7* genes: mating operator, selection operator, crossover scheme, replacement scheme, mutation severity, percentage of individuals to apply LS and number of iterations to run LS. Figure 6.12 presents the evolution of the meta-GA in terms of fitness value per generation.

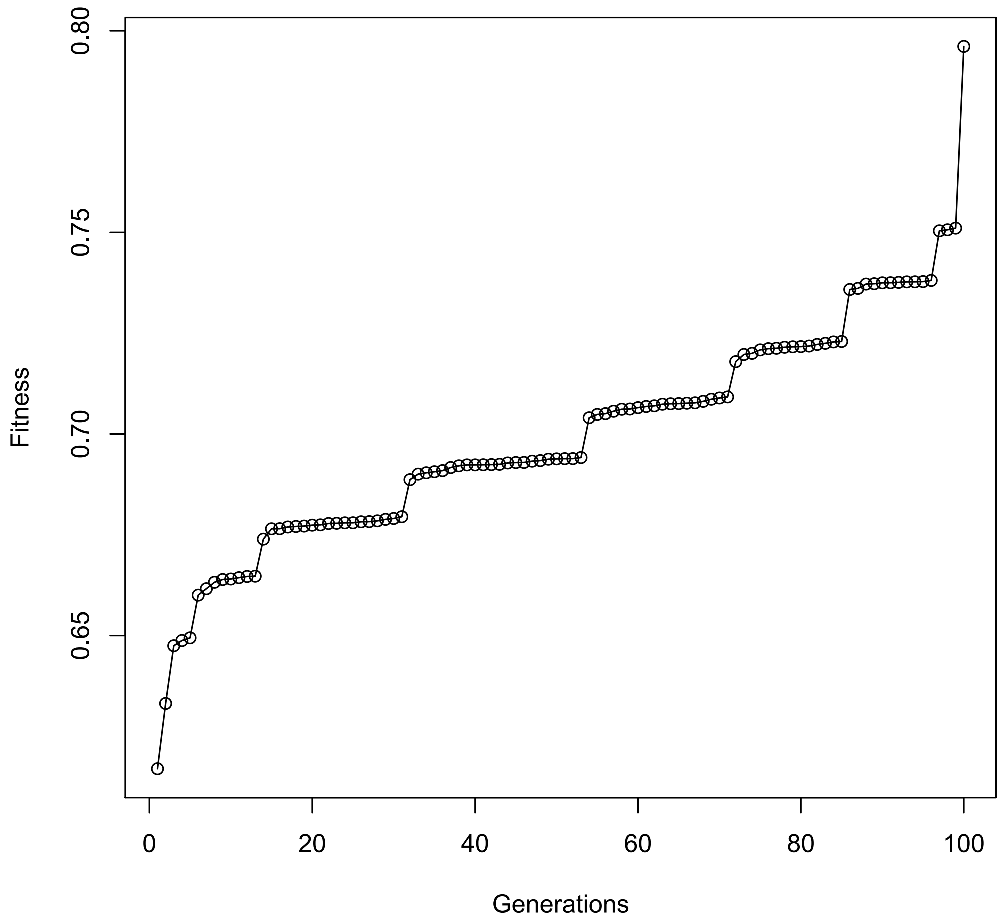

Figure 6.12: Fitness evolution of the meta-GA specifically designed for the tuning of the islands.

The best configuration found is similar to Configuration-1. This configuration gets the best fitness value for different time-frame sizes because it performs a nice exploration at the beginning and an effective exploitation at the end. The exact configuration of our single MA is the following one:

- *Encoding*: We encode every solution as an array of integers whose indexes represent the available agents at a given instant and the array contents refer to the profile assigned to each agent.

- *Fitness function*: We measure the service level resulting from the configuration of agents and incoming calls (see Section 5.2).
- *Population size*: The population of each island contains *30* different individuals encoded as hinted above.
- *Initialisation*: The initial population is randomly generated.
- *Selection*: Since the population must be bred each successive generation with new individuals, we have chosen a binary tournament selection.
- *Crossover*: The offspring inherits the common points in their parents and randomly receives the rest of genes from them.
- *Mutation*: We apply a perturbation over each gene of the chromosome with a probability of *0.03*.
- *Replacement policy*: We consider elitism with a probability of *0.93* to replace the worst-fitted individuals of the population in next generation. And, with a probability of *0.07*, a worse-fitted individual may be captured. Note that our basic GA relies on a steady-state scheme.

6.6.1.2 The Parallel Memetic Algortihm

We now detail and analyse the experiments conducted over two different problem instances (medium and high difficulty, respectively) for the parallel MA. We follow the same approach as for the simple MA. Firstly, we will show some configurations and then reveal the results achieved by the meta-GA for parallel MAs.

The two problem instances are composed by real data taken from our call centre during two different days at the same hour: a one-day campaign and a normal day. The size of each snapshot where each configuration has been executed is *600* seconds (*10* minutes). Note that around *800* incoming calls (n) simultaneously arrive during a normal day in such a time interval, whereas up to *2,450* simultaneous incoming calls may arrive during this interval during a commercial campaign. The number of agents (m), for each time interval, oscillates between *700* and *2,100*, having *16* different skills for each agent on average, grouped in skill profiles of *7* skills on average. The total number of the types of calls considered for this study is *167*. When the workload (n/m) is really high, finding the right assignment among agents and incoming calls becomes fundamental.

Some parameters of the parallel MA's configuration have been fixed (migration frequency, amount of migrants, synchronism type, number of processing nodes and same configuration in the isolated islands) while others have been varied (topology and migration and replacement policies) in order to understand what policies perform best in dynamic environments. Specifically, the following parameters of the parallel MA have been "frozen":

- *Migration frequency*: Each *60* seconds, all the islands are blocked for selection, migration and evaluation. If a generation is in process during the blockade in any of the islands, we take the previous stable population and go on with the process once the migrations have been carried out.

- *Amount of migrants*: Each island sends a set of migrants according to the migration policy, which represents the 10% of the size of the population (in practice, *3* individuals).

- *Synchronism*: We have applied a synchronous scheme in which every island waits for every incoming set of migrants they have to evaluate.

- *Number of processing nodes*: We consider *5* islands for every topology as this is the number of available processors, which are fully utilised by the CPUs.

The topologies and the migration and replacement policies to experiment with were fully described in Chapter 5. As for the islands, we have employed the configuration given by the meta-GA in the previous section.

For a fair comparison, every configuration has been run over the same problem instance *30* times. Table 6.8 shows the results obtained from the experimental phase for the medium-difficulty problem instance while Table 6.9 illustrates the respective ones for the high-difficulty problem instance. These tables show the best, worst and mean solution (and its standard deviation) out of the *30* executions performed. Ranking refers to the relative comparison between configurations, considering that the best setting-up represents the highest performance.

Policy	*Topology*	*Min*	*Max*	*Mean*	*SD*	*Ranking*
BFI-WFI	Star	0.846698	0.847310	0.847092	0.0003	9
BFI-RI	Star	0.846744	0.847361	0.847102	0.0003	8
BFI-BFI	Star	0.846195	0.847068	0.846511	0.0004	12
BFI-MDI	Star	0.847119	0.847742	0.847471	0.0003	5
BFIA-WFI	Star	0.847119	0.847742	0.847489	0.0003	4
BFI-WFI	Ring	0.847141	0.848006	0.847535	0.0004	3
BFI-RI	Ring	0.846933	0.847908	0.847290	0.0004	7
BFI-BFI	Ring	0.847119	0.847742	0.847364	0.0003	6
BFI-MDI	Ring	0.853954	0.860611	0.858281	0.0031	2
BFIA-WFI	Ring	0.857322	0.861109	0.859702	0.0017	1
BFI-WFI	Hub	0.846149	0.847488	0.846856	0.0005	10
BFI-RI	Hub	0.846654	0.847201	0.846848	0.0002	11
BFI-BFI	Hub	0.834190	0.835465	0.834838	0.0005	14
BFI-MDI	Hub	0.831358	0.831984	0.831603	0.0003	15
BFIA-WFI	Hub	0.845520	0.846874	0.846378	0.0006	13

Table 6.8: Results of each parallel MA configuration for each island topology in *30* executions (medium difficulty problem instance). Values refer to the fitness values obtained by each combination of migration & replacement policies. The caption of each combination of policies is: Best-fitted individuals for worst-fitted individuals (*BFI-WFI*), Best-fitted individuals for random individuals (*BFI-RI*), Best-fitted individuals for best-fitted individuals (*BFI-BFI*), Best-fitted individuals for most different individuals (*BFI-MDI*) and Best-fitted individual + "Annealing" for worst-fitted individuals (*BFIA-WFI*).

We perceive that the bidirectional ring seems to be the most appropriate topology for dynamic environments, most likely because this topology allows for opportune convergence while preserving the required diversity. The star topology also entails high-quality outcomes but quickly gets stagnated. The reason is that the master island receives many migrants from the subordinate islands after some migrations (and it is even worse when there are many subordinate islands), implying that populations eventually become very similar. This intuitively involves a lack of diversity so that the gain of fitness gets fatally damaged. This phenomenon affects much more strongly to the hub topology as, being all the islands interconnected to each other, the diversity diminishes too much after one or two migrations.

A second key conclusion is that the replacement of individuals is another important feature to set-up. In this manner, replacing the worst-fitted individuals in the receiving population by the best-fitted individuals of the source population does not always behave better than taking the most different individuals. The process of analysing differences in the chromosomes in contrast implies that the parallel MA can run fewer generations (as it is a costly operation) but entails better fitness values in the end. The underlying principle may be that fitness-based comparisons can occasionally be misleading or deceptive, leading to the situation in which two close individuals in terms of genes in common may have associated very different fitness values, whereas two far chromosomes in terms of genes in common may have assigned close fitness values (Whitley, 1991) [(176)]. Another consequence of measuring gene differences as compared to gauging fitness values is that the lift of the fitness curve has a smoother slope in the first generations. Naturally, replacing the best-fitted individuals of the receiving population by the best-fitted ones of the source population implies a slower convergence in each processing node as we will find a larger percentage of less fitted individuals.

Policy	*Topology*	*Min*	*Max*	*Mean*	*SD*	*Ranking*
BFI-WFI	Star	0.793660	0.793941	0.793796	0.0001	8
BFI-RI	Star	0.794102	0.794197	0.793561	0.0008	9
BFI-BFI	Star	0.791377	0.792888	0.792280	0.0007	12
BFI-MDI	Star	0.794265	0.794932	0.794693	0.0003	5
BFIA-WFI	Star	0.794288	0.795012	0.794688	0.0003	6
BFI-WFI	Ring	0.794610	0.795595	0.795223	0.0004	3
BFI-RI	Ring	0.794677	0.795216	0.794978	0.0002	4
BFI-BFI	Ring	0.794313	0.795221	0.794654	0.0004	7
BFI-MDI	Ring	0.792158	0.798497	0.796137	0.0028	2
BFIA-WFI	Ring	0.795679	0.798864	0.797696	0.0014	1
BFI-WFI	Hub	0.792373	0.792873	0.792669	0.0002	11
BFI-RI	Hub	0.791816	0.793589	0.792864	0.0008	10
BFI-BFI	Hub	0.790809	0.791874	0.791324	0.0004	14
BFI-MDI	Hub	0.790148	0.791492	0.790646	0.0006	15
BFIA-WFI	Hub	0.791097	0.791840	0.791566	0.0003	13

Table 6.9: Same as Table 6.8 but results now refer to the highly-difficult problem instance.

Let us now **apply the Meta-GA** for parallel MAs in order to automatically analyse how these parameters may have an impact on the relationship between topologies and problem structure.

Given the values for the *6* genes of the Meta-GA's chromosome, there are *6,480* possible combinations ($8 \cdot 3 \cdot 9 \cdot 5 \cdot 2 \cdot 3 = 6,480$). This may seem an easy search space but every evaluation takes time, as we have to re-execute the internal parallel MA each time, which is unfeasible in a production environment that requires fast adaptations. Of course, we can optimise this by avoiding recalculations previously made by the Meta-GA. The challenge should now be to develop a fast and effective Meta-GA that avoids performing too many iterations to find the right configuration or, at least, a good enough approximation (see the Meta-GA previously described).

In Figure 6.13, we can see that the best-fitted individual in the population of the Meta-GA evolves very quickly. We can even find the optimal configuration around generation-175. Best configuration found has been: *8* populations, bidirectional ring, *30* individuals per population, BFIA-WFI scheme, migrations each *60* seconds, *20%* of migrants.

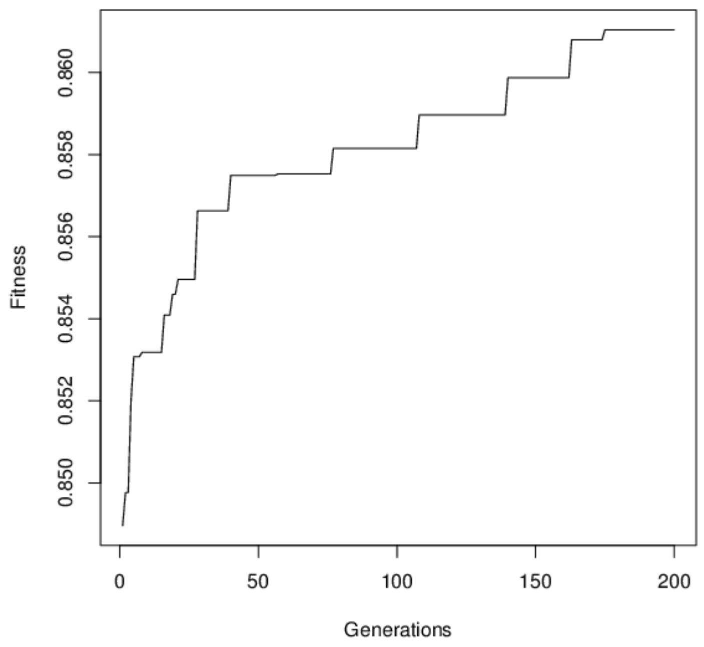

Figure 6.13: Fitness value of the best-fitted individual in the population of the Meta-GA generation-by-generation. We perceive that there is continuous evolution, especially at the beginning, and the fitness value reaches appealing levels.

In Figure 6.14, we can observe that, in few iterations, we can find a set of good candidates/individuals as the mean fitness of the population on each generation is quite high in less than *30* generations. This reflects that our Meta-GA does not only provide a single good solution but also multiple high-quality candidate solutions.

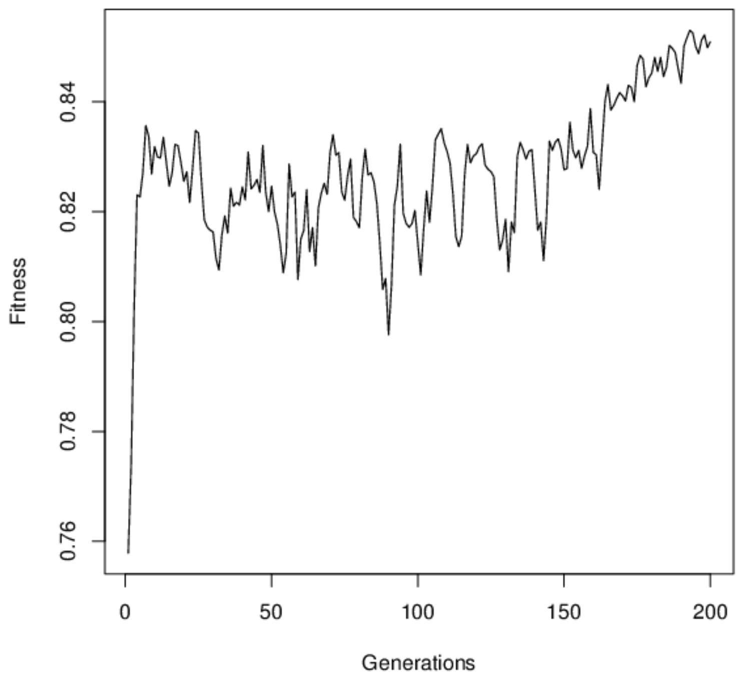

Figure 6.14: We show, for each generation, the mean fitness value of the individuals that compose the population of the Meta-GA. It reflects the mean quality of the individuals as the algorithm evolves.

Our Meta-GA also clearly outperforms its panmictic version as Figure 6.15 demonstrates. One of the main reasons of this good performance lies in the capability of evolving the topology and the migration and replacement policies when needed. The topology has an imperative impact on the migration and replacement policies since we can perceive a variation on the migration policies when the topology evolves (e.g., in generation-60).

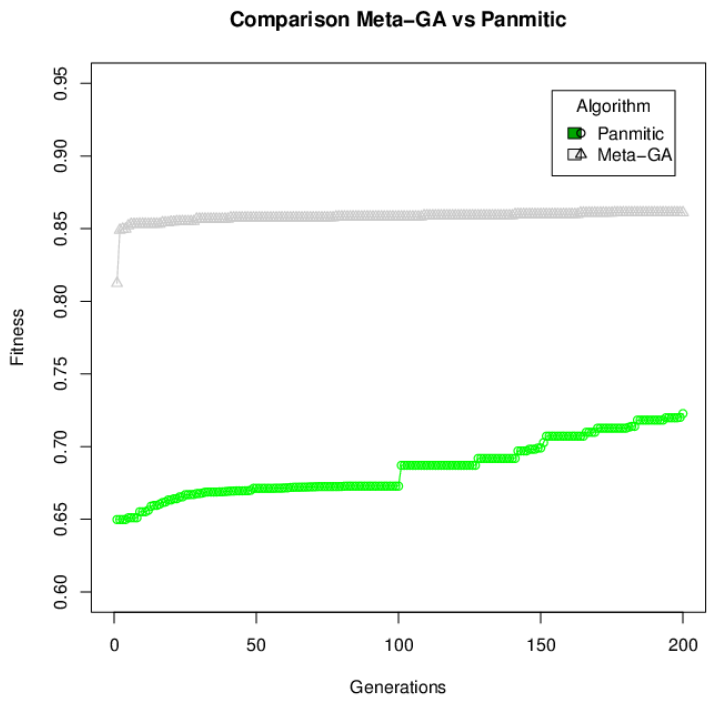

Figure 6.15: Fitness-based comparison between the panmictic algorithm and our Meta-GA. This figure shows the uplift of our Meta-GA as compared to the panmictic version of our parallel MA.

In Figure 6.16, we show the number of individuals having each topology in their genes generation by generation. As there are 20 individuals in the population of our Meta-GA, the sum-up of the three curves is always *20*.

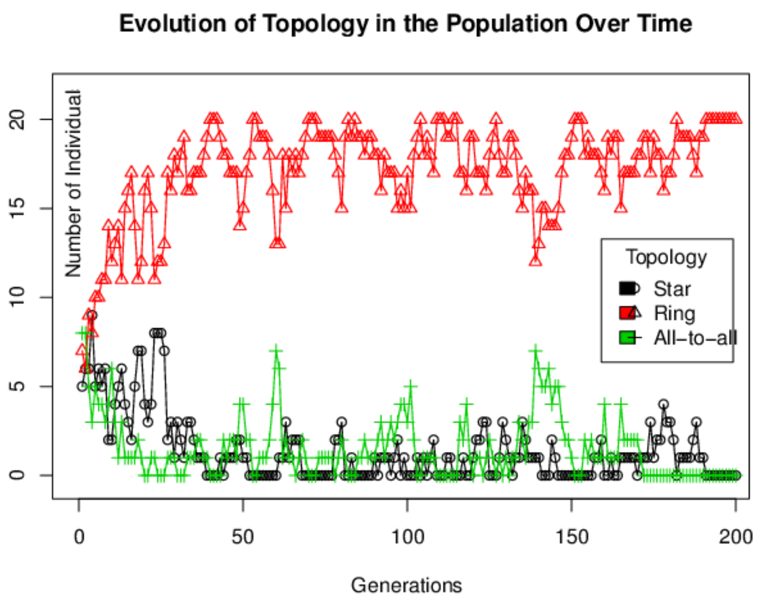

Figure 6.16: Evolution of Topology in the Population Over Time. We can see the number of individuals having each topology in their genes generation-by-generation. As there are *20* individuals in the population of the Meta-GA, the sum-up of the three curves is always *20*.

Again, bidirectional ring outperforms other more connected topologies, especially when the number of islands increases. When this happens, the population quickly converges towards the same solution. Therefore, bidirectional ring seems to be the most appropriate topology for dynamic environments, most likely because this topology allows for opportune convergence while maintaining the desired diversity. It is important to highlight that, for this problem, it is crucial to have a connected topology rather than several isolated islands working in parallel (this problem requires a collaborative scheme).

The star topology also entails high-quality outcomes but quickly suffers premature convergence. The reason is that the master island receives many migrants from the subordinate islands after some migrations (and it is even worse when there are many subordinate islands), implying that populations eventually become very similar. This intuitively entails a lack of diversity, so that the gain of fitness gets importantly damaged. This phenomenon hits harder to the hub topology as, being all the islands interconnected to each other, the diversity diminishes too much after one or two migrations.

The two previous paragraphs confirm the results of the previous section, reflecting that each problem structure needs a different island topology configuration. Dynamic, complex problems should be supported by medium-connected topologies like the bidirectional ring in order to make the parallel MA evolve properly.

In Figure 6.17, we show the number of individuals having each combination of migration-replacement policies in their genes generation-by-generation. As there are *20* individuals in the population of our Meta-GA, the sum-up of the three curves is consequently *20*.

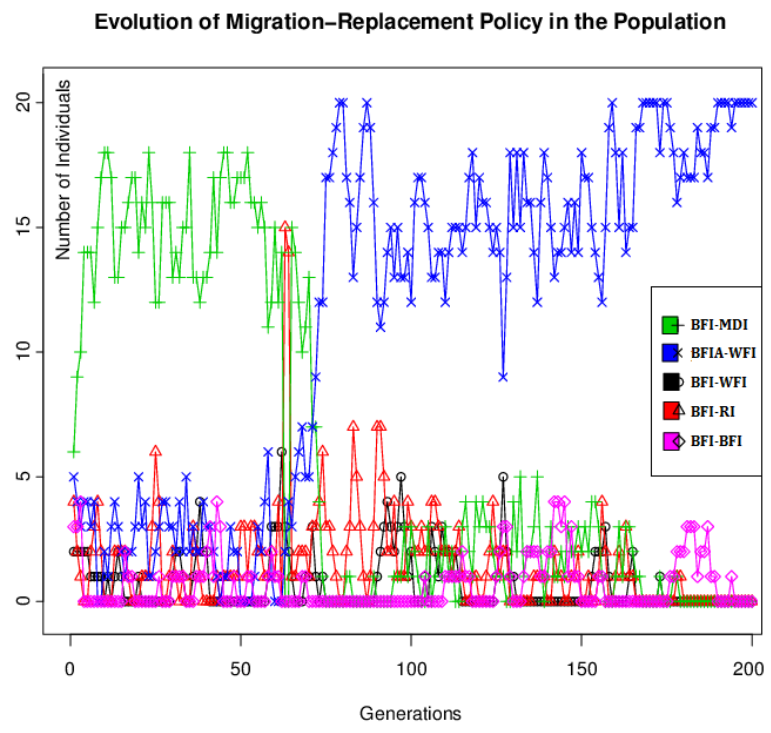

Figure 6.17: Evolution of Migration-Replacement Policy in the Population Over Time. We can see the number of individuals having each combination of migration-replacement policies in their genes generation-by-generation. As there are 20 individuals in the population of the Meta-GA, the sum-up of the three curves is consequently *20*.

The migration and replacement of individuals is another important feature to set-up. In this manner, replacing the worst-fitted individuals in the receiving population by the best-fitted individuals of the source population does not always behave better than taking the most different individuals. The process of analysing differences in the chromosomes in contrast implies that the internal parallel MA can run fewer generations (as it is a costly operation) but entails better fitness values in the end.

The fundamental principle may be that fitness-based comparisons can occasionally be misleading or deceptive, leading to the situation in which two close individuals in terms of genes in common may have associated very different fitness values, whereas two far chromosomes in terms of genes in common may have assigned close fitness values. Another consequence of measuring gene differences as compared to gauging fitness values is that the lift of the fitness curve has a smoother slope in the first generations. Naturally, replacing the best-fitted individuals of the receiving population by the best-fitted ones of the source population implies a slower convergence in each processing node as we will find a larger percentage of less fitted individuals. This way, the best migration policy has been sending the best fitted-individual with some non-necessarily best-fitted individuals (annealing set) as it provides diversity.

Another finding has been that having too many individuals on each population makes the algorithm slower and fewer generations are executed. Best values seem to range from *15* to *30* individuals per population.

The migration frequency is also important in the performance. Migrations should not be done with too much frequency, each population needs to evolve separately enough time. Of course, the amount of migrants should not be too big as the internal parallel MA may converge too fast to the same solutions. The impact is higher when the number of islands is rather large.

We have seen that parallel MAs can deal with complex, real-world application domains although they require specific tuning, depending on the nature of the problem being faced. This way, we have presented a Meta-GA for fine-tuning parallel MAs based on the island model.

6.6.2 Comparison of Metaheuristics

In a previous section, we have compared our forecasting module with other acknowledged predictive techniques. We have seen that our approach is not universally the best one as other learning algorithms outperform our uRprop when these have more time to train the models (the ANN for each CG) or when the dynamism is low. Fortunately, our learning algorithm behaves better for those CGs with higher dynamism when the time to train the models is reduced. But, we still have to determine the right assignment among agents and tasks, given the predictions provided by the forecasting module. This task is carried out by our search module, which will be compared with other famous MHs.

Now, similarly to the previous section, we describe the (two) problem instances (medium and high difficulty, respectively) that we have created from our dataset to test out our search module. For a fair comparison, every MH will be run over the same problem instances *50* times. These two problem instances are composed by real data taken from our MSCC's production environment during two different days at the same hour (from *12:40* to *12:45*, *300* seconds): a one-day campaign and a normal day. The size of the time-frame to execute all the MHs has been *300* seconds (*5* minutes) because it is the commonest time-frame size. We have selected the interval *[12:40-12:45]* as it is precisely the most critical hour of the day (highest load of the day: *n/m*). Note that around *800* incoming calls (n) simultaneously arrive during a normal day in such a time interval, whereas up to *2,450* simultaneous incoming calls may arrive during this interval throughout a commercial campaign. The number of agents (m), for each time interval, oscillates between *700* and *2,100*, having *16* different skills for each agent on average (*minimum=1* and *maximum=108*), grouped in profiles of *7* skills on average. The total number of CGs considered for this study is *167*. Therefore, when the workload (*n/m*) is really high, finding the right assignment among agents and incoming calls becomes fundamental.

Table 7.7 summarises the results obtained by each MH in *50* executions, starting from *50* different randomly generated initial solutions.

In our comparative study, we present dissimilar MHs, which cover diverse search strategies. Theoretically, due to the local character of the basic LS, it is complicated to reach a high-quality solution because the algorithm usually gets trapped in a neighbourhood when a local minimum is found. This occurs because the engine is always looking for better solutions, which probably do not actually exist in the neighbourhood. For this reason, sometimes, it is more appropriate to allow deterioration movements in order to switch to other regions of the search space.

This is precisely the shrewd policy of SA whose temperature allows for many oscillations (the probability of accepting a worse solution decreases according to the time) at the beginning of the process and only few ones at the end (fewer chances to select a worse solution as the algorithm is supposed to be refining the solution at this point). Specifically, we have chosen Cauchy's criterion because the convergence is faster than Boltzmann's and we only have *300* seconds to run the complete process. In Cauchy's scheme, the temperature is defined as $T = \frac{T}{(1+it)}$, where *it* is the iteration number and the initial temperature is $T = (\frac{\mu}{-log(\Phi)}) \cdot f(S*)$ where *f(S*)* is the cost of the initial solution, Φ stands for the probability of accepting a "μ" worse solution than the current one ($\Phi = \mu =$ *0.3*). Besides, this scheme avoids decreasing the distance between two solutions when the process converges (jumps in the neighbourhood). Therefore, the temperature must be high enough at the beginning to better explore the search space (its neighbourhood) and low enough at the end to intensify the search as well (exploitation of promising areas). The stopping condition must agree with the number of neighbours generated. The maximum of neighbour solutions generated each time is *L(T)=30* and the probability of accepting a worse solution is $exp(\frac{-\delta}{T_{it}})$ given that δ*=f(Neighbour_Solution)-f(Current_Solution)* and T_{it} is the temperature at iteration *it*.

We perceive from Table 7.7 that SA behaves worse than the other MHs except for the easiest instance of the problem. This may occur because we are not plenty of time in our environment and the power of SA relies on a progressive cooling. If we cool off the temperature too fast, we are missing the effectiveness of accepting worse solutions in some cases. Instead, if we cool off the temperature too slowly, we may be accepting worse solutions systematically without converging. We have applied a trade-off between exploration and exploitation but the computing time (*300* seconds) seems to be limited to apply SA to our environment (perhaps, things might change when having more time).

Another option to increase the diversity in the solutions is to enlarge the environment, as VNS does. This philosophy consists in making a systematic change upon the environment when the LS is used, increasing the environment when the process gets stagnated. In the VNS, the search is not restricted to only one environment as in the basic LS; instead, the neighbourhood changes as the algorithm progresses. In our experiments, we have considered three different environments e_{max} $= 3$: $(e_1 \rightarrow nh_1 = 0.3 \cdot n; e_2 \rightarrow nh_2 = 0.5 \cdot n; e_3 \rightarrow nh_3 = n;)$. These steps are repeated during *300* seconds (our stopping condition). Albeit we only consider three distinct neighbourhoods, the improvement of the VNS compared to basic LS is noteworthy. Consequently, the remarkable factor becomes the change in the number of neighbourhoods and their sizes as well as considering how the algorithm reacts in response. Table 7.7 also shows how VNS only slightly outperforms SA for the hardest instance of the problem.

Another strategy is to start from different initial solutions as ILS accomplishes. ILS generates a random initial solution and afterwards applies a basic LS. Subsequently, this solution is systematically mutated and thus refined. For ILS, the complete process is repeated during *300* seconds wherein the LS is the one proposed in Section 3.1.1 and the perturbation affects to the *3%* of agents. We can observe that ILS obtains solutions, which vaguely improve those given by SA and VNS for the hardest problem instance, although it performs worse for the simplest problem instance as Table 7.7 corroborates.

Another way to find an accurate solution involves using methods based on populations, such as MAs. If the diversity of the solution is low, then the MA converges to the closest neighbour. Nevertheless, when the selective pressure is high, individuals may be alike or even identical. To speed-up convergence, MAs apply an LS procedure upon a set of chromosomes (candidate solutions) that are refined every certain number of generations. Incorporating a hybridisation mechanism to the GA is valuable as the algorithm is improved in all respects (exploration and exploitation). The configuration of the GA's operators is the one provided previously whereas the LS mechanism is given in Section 3.1.1. Table 7.7 points out how our MA does not only outperform all the presented MHs for both problem instances but also remains more unwavering (less differences among best, worst and mean fitness values).

Algorithm	*Best solution*	*Worst solution*	*Average*	*Standard dev.*	*Effectiveness*
	Medium Hard	*Medium Hard*	*Medium Hard*	*Medium Hard*	*Medium Hard*
MA	0.796 0.758	0.785 0.751	0.796 0.754	0.001 0.001	100 100
ILS	0.768 0.728	0.755 0.722	0.763 0.725	0.002 0.003	95.85 96.15
VNS	0.790 0.727	0.766 0.723	0.775 0.724	0.005 0.001	97.36 96.02
SA	0.782 0.721	0.773 0.709	0.779 0.716	0.001 0.003	97.86 94.96

Table 6.10: Results obtained by the MHs in *50* executions starting from random initial solutions for two problem instances: medium and hard (larger number of incoming calls and high variability). Values refer to the fitness obtained by all the MHs.

It is important to remark that differences among techniques are not huge after reaching a fitness of *0.8* since the complexity exponentially increases in our environment. Therefore, minor improvements on the fitness value after that point are hard to obtain but very valuable to accomplish a fair workforce distribution.

Hitherto, we have demonstrated that our (single) MA has been able to outperform other MHs in our real-world production environment. However, we described our search module as a parallel MA based on an island model with the configuration given by the Meta-GA. So, can this architecture obtain better results than the single MA? Certainly, yes it can. Nevertheless, the improvement, which is remarkable, cannot be impressive as the complexity increases asymptotically. Table 6.11 compares the results obtained by the parallel MA with those obtained by the single MA. The parallel MA improves the results of the single MA in a *4%* for the hardest problem instance and *6.8%* for the easiest one. Although there is no a linear increment of fitness, the results are definitely better.

Algorithm	*Best solution*	*Worst solution*	*Average*	*Standard dev.*	*Effectiveness*
	Medium Hard	*Medium Hard*	*Medium Hard*	*Medium Hard*	*Medium Hard*
PMA	0.834 0.818	0.823 0.783	0.829 0.809	0.003 0.002	100 100
MA	0.796 0.758	0.785 0.751	0.796 0.754	0.001 0.001	96.01 93.20

Table 6.11: Results obtained by our Single and Parallel MAs in *50* executions starting from random initial solutions for the two problem instances studied. Values refer to the fitness.

Now, we will show the results of the Anova test for the MHs in Table 6.12. We see that P is very small, so that we can conclude that the variances of the populations are truly different in each experiment. In other words, we can state that the experiments are valid and different to each other.

Anova	Df	Sum Sq	Mean Sq	F value	Pr(>F)
ind	4	3.230	0.8075	13216	<2e-16
Residuals	235	0.014	0.0001		

Table 6.12: ANOVA test

6.7 Final Comparison

We have analysed the two main modules of our approach (forecasting module + search module) for separate and seen that these outperform other famous CC techniques.

We will combine these modules and introduce the dynamic time-frame described in Chapter 4. Specifically, we will compare our approach throughout a demanding working day (there was a commercial campaign during the day). This way, we have run the algorithm over a whole day with approximately *315,000* calls (up to *28,800* calls/hour and *2,450* simultaneous calls) under *12* double-core processors of a *Sun Fire E4,900* server (one for the interfaces, another one data pre-processing, another one for the database, two processors for controlling, two processors for the forecasting module, and the last five ones for the search module) with *96GB RAM*. The mean number of agents in each time-frame is *2,100*, having *16* different skills for each agent on average (*minimum=1* and *maximum=108*). The total number of CGs is *1035*. The mean processing times differ a lot, depending on the CG (from seconds to minutes).

Now, we compare our approach with classical SBR [(64)], ED-SBR (an improvement of classic SBR) and Koole's algorithm [(22)]. Figure 7.6 illustrates the real service level given by these techniques during a demanding working day. The graphs compile the *real service levels* for each CG (not the fitness vale presented for the search module), considering the relevance of each one (weighted CGs).

Since incoming traffic mainly arrives from *9* a.m. until *8* p.m.; therefore, we need to put special focus on the results for this time-interval and, particularly, for the load peaks that occur around *13* p.m. (see point *32* in Figure 7.6), *15* p.m. (see point *66* in Figure 7.6) and *19* p.m. (see point *100* in Figure 7.6) because, in these points, the load is much higher.

Our approach clearly improves the results reached by other CC algorithms in these critical points (load peaks). For the rest of points, we see that our algorithm usually behaves better than the rest of techniques. Classic SBR and ED-SBR sometimes offer a similar configuration of agents than our approach for some time points and, consequently, the same service levels; but, on average, the service levels are clearly worse than ours. Only in few points, the service level provided by ED-SBR and SBR is slightly higher than ours (e.g. around *11:45*, point *17*). This happens because in these points, our predictions had a greater error and SBR and ED-SBR consider the current state of the system.

However, we can see that differences are tiny in these critical points and we present more stable results over the time. This corroborates that an adaptive middle-term time-frame is recommended as algorithms can reach nearly optimal solutions while short-term algorithms often collapse in local optima. But, short-term algorithms present a high adaptability to changes that long-term time-frame techniques cannot cope with. These long-term based techniques generally extract patterns from historical records and are only appropriate for stable environments. For this reason, our algorithm and SBR outperform Koole's approach, which is designed for more stable MSCCs. Koole's algorithm finds very accurate solutions when the dynamism is more reduced such as classical staffing. Nevertheless, this is not the case of our environment and this kind of techniques cannot be efficiently applied to our MSCC.

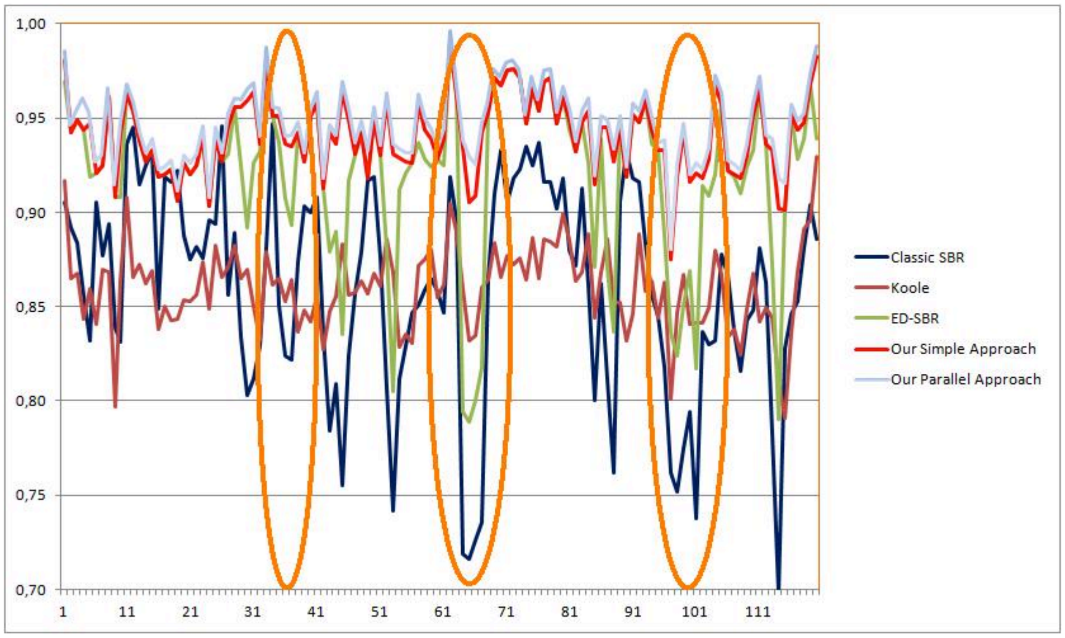

Figure 6.18: Service level given by different techniques for a whole campaign day. X-axis represents the time and Y-axis represents the real service level (not a fitness value).

Table 6.13 compares the results obtained by all techniques presented in Figure 7.6. Table 6.13 presents the mean service level for each time point, its standard deviation and the effectiveness, considering that our method represents the highest performance. Note that we are actually comparing the behaviour of our approach with other conventional CC techniques during a complete day rather than focusing on specific time-frames as we have presented until now.

It is crucial obtaining accurate results for isolated time-frames, but we cannot obviate that our global approach is being executed continuously, so that the transitions among system states (for each time-frame) must be taken into account.

Algorithm	*Real service level*	*Standard deviation*	*Effectiveness*
Our Parallel Approach	0.987	0.016	100
Our Simple Approach	0.941	0.020	95.339
ED-SBR	0.901	0.043	91.286
SBR	0.860	0.056	87.132
KOOLE	0.733	0.029	74.265

Table 6.13: Comparison of our simple and parallel approach with other relevant (call centre) algorithms over time.

The majority of the time-frames has a size of *300* seconds and, sometimes, *600* seconds. Only in certain peaks (around 9:00, 13:00, 15:00 and 19:00), the time-frame has a minimum size of *60* seconds. From 23:55 to 8:35, the normal size of the time-frame oscillates between *1,500* and *3,000* seconds.

Chapter 7

Conclusions, Contributions and Future work

The present chapter summarises the ideas exposed in this dissertation in Section 7.1 and highlights the major contributions of our work in Section 7.2. We also give some guidelines for future work in Section 7.3.

7.1 Summary and Conclusions

We have presented a novel approach to the problem of workforce distribution in dynamic multi-agent systems based on blackboard architectures (common repository of knowledge). We have seen that these systems are extremely complex and entail quick adaptations to a changing environment that only high-speed greedy heuristics can handle. These greedy heuristics consist in a permanent re-planning, considering the current system state. Intuitively, these quickly taken decisions are not appropriate for middle and/or long term planning due to the incessant erroneous movements.

However, we have demonstrated that the use of parallel memetic algorithms, which are more versatile than classical heuristics, can guide us towards more accurate solutions. With the intention of applying parallel memetic algorithms to such a dynamic environment, we have put forward a reformulation of the traditional problem of workforce distribution in dynamic multi-agent systems based on backboard architectures, which coalesces predictions of future system states with a precise search mechanism, by dynamically enlarging or diminishing the time-frame considered. We have claimed that the size of the time-frame depends upon the dynamism of the system (smaller when there is high dynamism and larger when there is low dynamism).

The present work has also illustrated how nearly optimal solutions each v seconds (size of the time-frame) outperforms continuous bad distributions when the right size of the time-frame is determined, and predictions and optimisations are correctly carried out. Particularly, we have proposed a neural network with an upgraded resilient propagation learning algorithm for predicting future system variables and a parallel memetic algorithm based on an island scheme to perform the assignment of incoming tasks to the right available agents.

Our approach has been tested out on a real-world production environment from Telefónica, which is a large multinational telephone operator. We have shown that our approach not only outperforms other conventional techniques for separate but also as a unified technique. Therefore, we have obtained more accurate predictions than other famous forecast techniques for various problem instances. Besides, our search module based on a parallel memetic algorithm has outperformed other meta-heuristics under different scenarios. Additionally, the combination of the two modules with the adaptive middle-term time-frame has involved fine results. This corroborates that an adaptive middle-term time-frame can be a very powerful approach when having the required tools to implement it.

7.2 Summary of Contributions

The contributions of this work can be devised from diverse perspectives although the main contribution is the presentation of a novel approach to the workforce distribution problem, which coalesces forecasting with optimisation by considering an adaptive time-frame mechanism. We also apply this approach to a real-world production environment (multi-skill call centre) from one of the largest telecommunications operators around the world (Telefónica), outperforming other existing call-centre techniques.

Typically, traditional process management systems distribute tasks to queues from which agents pull work, regularly without opting for the precise tasks they actually desire to deal with. Instead, our approach enhances workforce distribution by additionally injecting real-time knowledge of the task, individual skill sets, and availability and utilisation of the workforce, allowing for dynamic and active distribution of tasks over time.

Additionally, our method provides further clearness on customer service level agreements (easier to fulfil service level requirements) and endows with insights into optimisation, offering outstanding customer service.

Our approach also enables us to work at a lower level of granularity (fine-grain) than short-term algorithms do (coarse-grain), because our search algorithm has more time to find a solution than conventional techniques, thanks to the predictions of future states. We can then work at agent's profile level instead of predefined sets of agents as other methods impose (other techniques cannot work at an individual agent level). Other conventional techniques consider steady environments, which are far from the soundness of a truly dynamic method.

Furthermore, other technical contributions of this thesis can be summarised as follows:

- This work proposes a scalable and parallelisable approach to a real-world NP-hard problem, using different techniques from the field of artificial intelligence. This approach lies in a new mechanism, which combines an accurate forecasting module with a powerful, adaptive search module.
- We contribute to the state-of-the-art with a sophisticated and robust method that is self-adapted to the changes of the system over time (time series for determining the dynamism of the system and a Markov model for performing the system transitions).
- We present an upgraded version of the resilient back-propagation learning algorithm, specially adapted to highly-dynamic environments like the multi-skill call centre.
- Novel migration and replacement policies for parallel genetic algorithms are proposed in order to preserve a fair balance between exploration and exploitation in the search process. These operators are inspired in other meta-heuristics schemes like simulated annealing and tabu search.
- A novel meta-genetic algorithm for the configuration of parallel evolutionary algorithms is also presented in this work.
- We also propose a method for evaluating partial fitness functions in order to speed-up the evaluations of candidate solutions.
- Three exhaustive comparisons among different classical forecasting techniques, various classical heuristics for dynamic multi-agent systems and other meta-heuristics applicable to dynamic multi-agent systems are provided from multiple points of view.

Finally, the contributions to the scientific literature have produced *11* peer-reviewed publications that are fully described in the Appendix A of this manuscript:

1. *Publications in international journals*:

   (a) Millán-Ruiz, D. and Hidalgo, J.I.: *Forecasting Call-Centre Arrivals.* Journal of Forecasting, Wiley, Ed. Derek W. Bunn, 2013.

   (b) Arnaldo, I., Contreras, I.; Millán-Ruiz, D., Hidalgo, J.I. and Krasnogor, N.: *Matching Island Topologies to Problem Structure in Parallel Evolutionary Algorithms.* Soft Computing. A Fusion of Foundations, Methodologies and Applications. Special Issue on Bio-Inspired Algorithms with Structured Populations, Springer, 2013.

2. *Publications in relevant international conferences*:

   (a) Millán Ruiz, D. and Hidalgo, J.I.: *Migration and Replacement Policies for Preserving Diversity in Dynamic Environments.* Proceedings of the 12th European Conference on the Applications of Evolutionary Computation (EvoApplications 2012, Málaga, Spain, April 11-13, 2012.

   (b) Millán Ruiz, D., Hidalgo, J.I.: *Comparison of Metaheuristics for Workforce Distribution in Multi-Skill Call Centres.* Proceedings of the International Joint Conference on Computational Intelligence (ICEC 2010), Valencia, Spain, October 24-26, 2010.

   (c) Millán Ruiz, D., Hidalgo, J.I.: *A Parallel Memetic Algorithm for Dynamic Workload Distribution in Multi-Agents Systems.* Proceedings of the 3rd Workshop on Parallel Architectures and Bioinspired Algorithms held In conjunction with PACT 2010, Vienna, Austria, September 11-15, 2010.

(d) Millán Ruiz, D., Pacheco, J., Hidalgo, J.I., Vélez, JL: *Forecasting in a multi-skill Call Centre.* Proceedings of the 10th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2010), Zakopane, Poland, June 13-17, 2010.

(e) Millán Ruiz, D., Hidalgo, J.I.: *A Memetic Algorithm for Workforce Distribution in Dynamic Multi-Skill Call Centres.* Proceedings of the 10th European Conference on Evolutionary Computation in Combinatorial Optimisation (EVOCOP 2010), p. 178-189, Istanbul, Turkey, April 7-9, 2010.

(f) Meléndez, J., Lopez, B., Millán-Ruiz, D.: *Probabilistic models to assist maintenance of multiple instruments.* Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2009), p. 1499-1503, Palma de Mallorca, Spain, September 22-26, 2009.

(g) Pacheco, J., Millán-Ruiz, D., Vélez, JL: *Neural Networks for Forecasting in a multi-skill Call Centre.* Proceedings of the 11th International Conference on Engineering Applications of Neural Networks (EANN 2009), p. 291-300, London, UK, August 27-29, 2009.

(h) Martinez-Lopez, R., Millán-Ruiz, D., Martin-Dominguez, A.; Toro-Escudero, MA: *An Architecture for Next-Generation of Telecare Systems Using Ontologies, Rules Engines and Data Mining.* Proceedings of the International Conferences on Computational Intelligence for Modelling, Control and Automation, Intelligent Agents, Web Technologies and Internet Commerce, and Innovation in Software Engineering (CIMCA 2008), p. 31-36, Vienna, Austria, December 10-12, 2008.

3. *Publications in national conferences*:

(a) Millán Ruiz, D., Hidalgo, J.I.: *Algoritmo memético paralelo para la distribución de esfuerzo en centros de llamadas dinámicos multiagente y multitarea.* Proceedings of the 7th Spanish Conference on Meta-heuristics, Evolutionary Algorithms and Bioinspired Algorithms (MAEB 2010), Valencia, Spain, September, September 8-10, 2010.

7.3 Areas of Future Research

To conclude, we propose some guidelines for future work. We recommend analysing more datasets and more problem instances because we may obtain different conclusions with regard to the configuration of our complete approach. So, if the arriving load is easy to predict we might consider simpler forecast techniques.

Another possible improvement would entail the analysis of more complex feature selection techniques. We have employed PCA, which detects linear combination of features in order to reduce the number of inputs but there are non-linear approximations like the wrapper subset evaluation.

A deeper study of constraint handling should be done as our proposal is dependent on our specific domain (e.g. we may have different ranges for the levels of constraints).

For the dynamism levels, we can also have a continuous approximation (without levels) for those dynamic multi-agent systems where agents are not humans so that we do not need to care about the agents' rights (we can potentially change their profiles at any time without regulation constraints).

Additionally, we suggest that an analogous study for the search module comparison should be done, considering multi-objective evolutionary approximations (such as SPEA-II and NSGA-II), given our problem formulation.

Bibliography

[1] Adamidis, P.: *Review of Parallel Genetic Algorithms.* Technical Report, Department of Electrical and Computer Engineering, Aristotle University, Thessalonik, 1994.

[2] Ahrens, J.H. and Ulrich, D.: *Computer Methods for Sampling from Gamma, Beta, Poisson and Binomial Distributions.* Computing 12 (3): 223-246, 1974.

[3] Alba, E. and Troya, J.M.: *Analyzing synchronous and asynchronous parallel distributed genetic algorithms.* Future Generation Comp. Syst., pages 451-465, 2001.

[4] Alberts, L.J.: *Churn prediction in the mobile telecommunication industry: An application of Survival Analysis in Data Mining.* http://www.personeel.unimaas.nl/westra/PhDMaBateaching/ Graduation-Students/LaurensAlberts2006/Presentatie.ppt, 2006.

[5] Andrews, B.H. and Cunningham, S.M.: *L.L. Bean improves call-center forecasting.* Interfaces, 25(6):1-13, 1995.

[6] Andrews, G.R.: *Foundations of Multithreaded, Parallel, and Distributed Programming.* Addison-Wesley, ISBN 0-201-35752-6, 2000.

[7] Antipov, A. and Meade, N.: *Forecasting call frequency at a financial services call centre.* Journal of operation research, pp. 953-960, 2002.

[8] Aouchiche, M; Bonnefoy, J.M.; Fidahoussen, A.; Caporossi, G.; Hansen, P.; Hiesse, L.; Lacheré, J. and Monhait, A.: *Variable neighborhood search for extremal graphs: Variations on graffiti.* Congressus Numerantium, 148:129-144, 2001.

[9] Aquino, I.; Perez, C.; Chavez, J.K. and Oporto, S.: *Daily Load Forecasting Using Quick Propagation Neural Network with a Special Holiday Encoding.* Neural Networks, 2007. IJCNN 2007. International Joint Conference on Neural Networks, 2007.

[10] Araujo, L. and Merelo, J.J.: *Diversity Through Multiculturality: Assessing Migrant Choice Policies in an Island Model.* IEEE Transactions on Evolutionary Computation, Vol. 15, No. 4, 2011.

[11] Asiki, A.; Tsoumakos, D. and Koziris, N.: *An Adaptive On-line System for Efficient Processing of Hierarchical Data.* Proceedings of the 18th International ACM Symposium on High Performance Distributed Computing (HPDC'09), Garching, Germany, 2009.

[12] Audibert, J.; Munos, R. and Szepesvári, C.: *Tuning bandit algorithms in stochastic environments.* ALT, 2007.

[13] Auer, P.; Cesa-Bianchi, N. and Fischer, P.: *Finite time analysis of the multiarmed bandit problem.* Machine Learning, 47(2-3) pp. 235-256, 2002.

[14] Avramidis, A. N., Deslauriers, A. and L'Ecuyer. P. *Modeling daily arrivals to a telephone call center.* Management Science, 50, 896-908, 2004.

[15] Avramidis, A.N.; Chan, W.; Gendreau, M.; L'Ecuyer, P. and Pisacane, O.: *Optimizing daily agent scheduling in a multiskill CC*. European Journal of Operational Research, 2009.

[16] Baeza-Yates, R. and Ribeiro-Neto, B.: *Modern Information Retrieval.* Acm Press Series, Addison Wesley, 1999.

[17] Baker, K.R. and Kanet, J.J.: *Job shop scheduling with modified due dates.* Journal of Operations Management 4, 11-22, 1983.

[18] Baker, S.A and Milner, K. R.: *A Process Migration Harness for Dynamic Load Balancing.* In Edwards, pages 52-61, 1991.

[19] Belding, T.: *The distributed genetic algorithm revisited.* Proceedings of the Sixth International Conference on Genetic Algorithms, pages 114-121, 1995.

[20] Van den Bergh, K.: *Predicting Call Arrivals in Call Centers.* Thesis, 2006.

[21] Den Besten, M. and Stutzle, T.: *Neighborhoods revisited: Investigation into the effectiveness of variable neighborhood descent for scheduling.* In MIC 2001, pages 545-549, Porto, 2001.

[22] Bhulaii, S.; Koole, G. and Pot, A.: *Simple Methods for Shift Scheduling in Multiskill Call Centers.* M&SOM 10(3), 411-420, 2008.

[23] Bianchi, L., Jarrett, J. and Hanumara, R.C.: *Improving forecasting for telemarketing centers by ARIMA modelling with intervention*, International Journal of Forecasting, 14, 497-504, 1998.

[24] Bishop, C.M.: *Neural Networks for Pattern Recognition.* Oxford University Press. ISBN 0-19-853849-9, 1995.

[25] Bowerman, B.L. and O Connell, R.T.: *Forecasting and Time Series: An Applied Approach*. Duxbury classic series, third edition, 1993.

[26] Brain, Z. and Addicoat, M.: *Using Meta-Genetic Algorithms to tune parameters of Genetic Algorithms to find lowest energy Molecular Conformers*. Proc. of the Alife XII Conference, Odense, Denmark, 2010.

[27] Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H.,Zeltyn, S. and Zhao, L.: *Statistical Analysis of a Telephone Call Center: A Queueing-Science Perspective*. November 2002.

[28] Brucker, P.: *Scheduling algorithms*. 2^{nd} ed. Springer, Heidelberg, 1998.

[29] Buhmann, M. D.: *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, ISBN 978-0-521-63338-3, 2003.

[30] Cadieu C.; Kouh M.; Pasupathy A.; Conner C.E.; Riesenhuber M. and Poggio T.: *A Model of V4 Shape Selectivity and Invariance*. J Neurophysiol 98: 1733-1750, 2007.

[31] Cantú-Paz, E.: *A Survey of Parallel Genetic Algorithms*. Calc. Parallelles, 1998.

[32] Cantú-Paz, E.: *Topologies, migration rates, and multi-population parallel genetic algorithms*. Proceedings of GECCO, pages 91-98, 1999.

[33] Cantú-Paz, E.: *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Press, ISBN:0792372212, 2000.

[34] Caragiannis, I.; Flammini, M.; Kaklamanis, C.; Kanellopoulos, P. And Moscardelli, L.: *Tight bounds for selfish and greedy load balancing*. ICALP, LNCS, 2006.

[35] Chakraborty, B. and Chaudhuri, P.: *On The Use of Genetic Algorithm with Elitism in Robust and Nonparametric Multivariate Analysis*. Austrian Journal of Statistics, Volume 32, Number 1-2, pp. 13-27, 2003.

[36] Chauvet, F.; Proth, J.M. and Soumare, A.: *The simple and multiple job assignment problems*. International Journal of Production Research 38(14), 3165-3179, 2000.

[37] Chen, Z.; Yang, M.; Francia, G. and Dongarra, J.: *Self Adaptive Application Level Fault Tolerance for Parallel and Distributed Computing*. ipdps, pp.414, IEEE International Parallel and Distributed Processing Symposium, 2007.

[38] Clune, J.; Goings, S.; Punch, B. and Goodman, E.: *Investigations in Meta-GAs: Panaceas or Pipe Dreams?*. GECCO'05, June 25-29, Washington, DC, USA, 2005.

[39] Cohoon, J.; Hegde, S.; Martin, W. and Richards, D.: *Punctuated equilibria: a parallel genetic algorithm*. Proceedings of the Second International Conference on Genetic Algorithms, pages 148-154, Hillsdale, NJ, USA, 1987.

[40] Collins, R.: *Studies in Artificial Evolution*. Ph.D. dissertation, Department of Computer Science, University of California at Los Angeles, 1992.

[41] Cotez, P.; Rio, M.; Rocha, M. and Sousa, P.: *Internet Traffic Forecasting using Neural Networks*. International Joint Conference on Neural Networks, 2006.

[42] Cutkosky, M.R.; Tenenbaum, J.M. and Glicksman J.: *Collaborative Engineering over the Internet*. Communication of the ACM, 39(9):78-87, 1996.

[43] Cotta, C. and Fernández, A.J.: *Memetic Algorithms in Planning, Scheduling*

and Timetabling. Evolutionary Scheduling, pp. 1-30, K. Dahal, K.-C. Tan, P.I. Cowling (eds.), Springer-Verlag, Berlin-Heidelberg, 2007.

[44] Cvijovic, D.; Klinowski, J.: *Taboo search - an approach to the multiple minima problem*. Science, 664-666, 1995.

[45] Darbellay, G.A. and Slama, M.: *Forecasting the short-term demand for electricity: Do neural networks stand a better chance?*. International Journal of Forecasting, volume 16, 71-83, 2000.

[46] Dasgupta, K.; Singh, R.; Viswanathan, B.; Chakraborty, D.; Mukherjea, S.; Nanavati, A. and Joshi, A.: *Social ties and their relevance to churn in mobile telecoms networks*. EDBT'08. March 25-30, Nantes, France, 2008.

[47] Dechter, R.: *Constraint Processing*. Morgan Kaufmann. ISBN 1-55860-890-7, 2003.

[48] Dhakal, S.; Hayat, M.M.; Pezoa, J.E.; Yang, C. and Bader, D.A.: *Dynamic Load Balancing in Distributed Systems in the Presence of Delays: A Regeneration-Theory Approach*. IEEE Transactions on Parallel and Distributed Systems, 485-497, 2007.

[49] Di Caro, G. and Dorigo, M.: *Mobile agents for adaptive routing*. In H. El-Rewini, editor, Proceedings of the 31st International Conference on System Sciences (HICSS-31), pages 74-83. IEEE Computer Society Press, Los Alamitos, CA, 1998.

[50] Dorigo, M.: *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italie, 1992.

[51] Dutta, M. and Chaubey, V.K.: *Performance Analysis of All-Optical WDM Network with Wavelength Converter Using Erlang C Traffic Model*. Communications

in Computer and Information Science, Information Processing and Management, Springer, ISBN 978-3-642-12213-2, pp. 238-244, 2010.

[52] Eager, D.L.; Lazowska, E.D. and Zahorjan, J.: *Adaptive Load Sharing in Homogeneous Distributed Systems.* IEEE Trans. Software Eng., vol. 12, no. 5, pp. 662-675, May 1986.

[53] Easton, F.F. and Goodale, J.: *Schedule recovery: Unplanned absences in service operations*, Decision Sciences, 36, 459-488, 2005.

[54] Eddy, S. R.: *What is dynamic programming?* Nature Biotechnology, 22, 909-910, 2004.

[55] Erman, L.; Hayes-Roth, F.; Lesser, V.R. and Reddy, D.R.: *The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty.* Computing Surveys, 12(2):213-253, 1980.

[56] Fahlman, S. and Lebiere, C.: *The Cascade-Correlation Learning Architecture.* National Science Foundation under Contract Number EET-8716324 and Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976, 1991.

[57] Fang, Y.: *Hyper-Erlang Distribution Model and its Application in Wireless Mobile Networks.* Wireless Networks, Springer Netherlands, ISSN 1022-0038, Volume 7, Number 3, 2001.

[58] Festa, P. and Resende, M.G.C.: *GRASP: An annotated bibliography.* Essays and Surveys on Metaheuristics, pp. 325-367, Kluwer Academic Publishers, 2002.

[59] Franklin, S. and Graesser, A.: *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents.* Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

[60] Gallager, R.G.: *Discrete stochastic processes*. Kluwer, Boston, 1996.

[61] Gans, N., Koole, G. and Mandelbaum, A.: *Telephone Call Centers: a Tutorial and Literature*. Review. 2, 2002.

[62] García Pajares, R., Benítez, J.M. and Sáinz Palmero, G.: *Feature Selection for Time Series Forecasting: A Case Study*. ISBN: 978-0-7695-3326-1, September, 2009.

[63] Garey, M. and Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman. ISBN 0-7167-1045-5, 1979.

[64] Garnett, O. and Mandelbaum, A.: *An Introduction to Skills-Based Routing and its Operational Complexities*. Teaching Note, 2000.

[65] Gayon, J. and Matthew, C.: *Darwinism's Struggle for Survival: Heredity and the Hypothesis of Natural Selection*. Cambridge University Press, pp. 158, 1998.

[66] Glover, F.: *Tabu Search - Part I*, ORSA Journal on Computing, 190-206, 1989.

[67] Glover, F. *Tabu Search - Part II*, ORSA Journal on Computing, 4-32, 1990.

[68] Glover F.: *A Template for Scatter Search and Path Relinking*. Lecture Notes in Computer Science, 1363: 1-53, 1997.

[69] Glover, F. and M. Laguna: *Tabu Search*. Kluwer, Norwell, MA, 1997.

[70] Gog, A.; Chira, C.; Dumitrescu, D. and Zaharie, D.: *Analysis of Some Mating and Collaboration Strategies in Evolutionary Algorithms*. 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 08, 2008.

[71] Goldberg, D. E.: *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley. pp. 41, 1989.

[72] Gomory, R.E.: *Outline of an algorithm for integer solutions to linear programs*. Bull. Amer. Math. Soc. 64 275-278, 1958.

[73] Gordon, V.S.; Whitley, D. and Böhn, A.: *Dataflow parallelism in genetic algorithms*. Parallel Problem Solving from Nature, 2, pages 533-42, 1992.

[74] Gouarderes, G.; Minko, A. and Richard, L.: *Simulation and multi-agent environment for aircraft maintenance learning*. AIMSA 2000, Varna, vol. 1904, pp. 152-166, ISBN 3-540-41044, 2000.

[75] Guttman, R.H. and Maes, P.: *Agent-mediated Integrative Negotiation for Retail Electronic Commerce*. Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET'98), Minneapolis, Minnesota, pp. 70-80, 1998.

[76] Gwiazda, T.: *Genetic Algorithms Reference*. Vol.1 Crossover for single objective numerical optimization problems. Lomianki, 2006.

[77] Hall, P., Kay, J., and Titterington, D.: *Asymptotically Optimal Difference-Based Estimation of Variance in Nonparametric Regression*. Biometrika, 77, 521-528, 1990.

[78] Hansen, P. and Mladenovic, N.: *Variable neighbourhood search: Principles and applications*. European Journal of Operations Research, 130:449-467, 2001.

[79] Harchol-Balter, M. and Downey, A.: *Exploiting process lifetime distributions for dynamic load balancing*. ACM Transactions on Computer Systems, vol. 15, pp. 253-285, 1995.

[80] Hart, W.E., Krasnogor, N. and Smith, J.E.: *Recent Advances in Memetic Algorithms*. Volume 166 of Studies in Fuzziness and Soft Computing, Springer Berlin Heidelberg New York, 2004.

[81] Hayes-Roth, B.: *A blackboard architecture for control*. Artificial Intelligence, pp. 251-321, 1985.

[82] Haykin, S.: *Neural Networks: A Comprehensive Foundation*. Second edition, Prentice Hall, ISBN 0132733501, 1998.

[83] Holland, J.H.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[84] Hoos, H.H. and Stutzle, T.: *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.

[85] Hopfield, J.J.: *Neural networks and physical systems with emergent collective computational abilities*. Proceedings of the National Academy of Sciences of the USA, vol. 79, no. 8, pp. 2554-2558, 1982.

[86] Hu, J. and Goodman, E.: *The hierarchical fair competition (HFC) model for parallel evolutionary algorithms*. Proceedings of the Congress on Evolutionary Computation, pages 49-54, Honolulu, USA. IEEE Press, 2002.

[87] Hur, D., Mabert, V.A. and Bretthauer, K.M.: *Real-time work schedule adjustment decisions: An investigation and evaluation*. Production and Operations Management 13 322-339, 2004.

[88] Ibrahim, R. and L'Ecuyer, P.: *Forecasting Call Center Arrivals: Fixed-Effects, Mixed-Effects, and Bivariate Models*, Manufacturing & Service Operations Management, INFORMS, 2012.

[89] Igel, C. and Hüsken, M.: *Improving the Rprop Learning Algorithm.* Proceedings of the Second International Symposium on Neural Computation, NC2000, pp. 115-121, ICSC Academic Press, 2000.

[90] Igel, C. and Hüsken, M.: *Empirical Evaluation of the Improved Rprop Learning.* Neurocomputing 50, 105-123, 2003.

[91] Inayatullah, M.; Ullah, F.K.; Khan., A.N.: *An Automated Grade Of Service Measuring System.* IEEE-ICET 2006, 2nd International Conference on Emerging Technologies, Peshawar, Pakistan 13-14, pp.230-237, 2006.

[92] Ingolfsson, A.; Cabral, E. and Wu, X.: *Combining integer programming and the randomization method to schedule employees.* TR, School of Business, University of Alberta, 2007.

[93] Ishibuchi, H. and Shibata, Y.: *A Similarity-based Mating Scheme for Evolutionary Multiobjective Optimization.* Lecture Notes in Computer Science, 2003.

[94] Jacobs, R.A.: *Increased rates of convergence through learning rate adaptation.* Neural Networks, 1(4):295-307, 1988.

[95] Jolliffe, I.T.: *Principal Component Analysis.* Series: Springer Series in Statistics, 2^{nd} ed., Springer, NY, XXIX, 487 p. 28 illus, ISBN 978-0-387-95442-4, 2002.

[96] Jongbloed, G. and Koole, G.: *Managing uncertainty in call centers using Poisson mixtures.* Applied Stochastic Models in Business and Industry, 17, 307-318, 2001.

[97] Kasabov, N.: *Introduction: Hybrid intelligent adaptive systems.* International Journal of Intelligent Systems, Vol.6, 453-454, 1998.

[98] Katayama, K. and Narihisa, H.: *Iterated local search approach using genetic*

transformation to the traveling salesman problem. Proceedings of GECCO'99, Vol. 1. Morgan Kaufmann, pp. 321-328, 1999.

[99] Kennedy, J. and Eberhart, R.: *Particle Swarm Optimization.* Proceedings of IEEE International Conference on Neural Networks, pp. 1942-1948, 1994.

[100] Kingsland, S. E.: *Modeling nature: episodes in the history of population ecology.* Chicago: University of Chicago Press, ISBN 0-226-43728-0, 1995.

[101] Kirkpatrick, S.; Gelatt, C. and Vecchi, M.P. *Optimization by Simulated Annealing.* Science 220 (4598): 671-680, 1983.

[102] Klungle, R. and Maluchnik, J.: *Call center forecasting at AAA Michigan.* The Journal of Business Forecasting, 20(4):8-13, 1998.

[103] Kocsis L. and Gyorgy A.: *Efficient Multi-start Strategies for Local Search Algorithms.* LNCS, Machine Learning and Knowledge Discovery in Databases, 705-720, 2009.

[104] Kohavi, R.: *A study of cross-validation and bootstrap for accuracy estimation and model selection.* Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence 2 (12): 1137-1143, 1995.

[105] Koole, G.; Pot, S. and Talim, J.: *Routing heuristics for multi-skill call centers.* Proceedings of the Winter Simulation Conference, p. 1813-1816, 2003.

[106] Koole, G.: *Call Center Mathematics: A scientific method for understanding and improving contact centers*, http://www.cs.vu.nl/ koole/ccmath/book.pdf, 2006.

[107] Kotsiantis, S. and Pintelas, P.: *Selective Averaging of Regression Models.* Annals of Mathematics, Computing & TeleInformatics, Vol 1, No 3, pp. 66-75, 2005.

[108] Kotsiantis, S., Kanellopoulos, D. and Pintelas, P.: *Local Additive Regression of Decision Stumps*. Lecture Notes in Artificial Intelligence, Springer-Verlag, Vol. 3955, SETN 2006, pp. 148-157, 2006.

[109] Kuchen, H. and Wagener, A.: *Comparison of Dynamic Load Balancing Strategies*. In K. Boyanov, editor, Parallel and Distributed Processing, pages 303-314. North-Holland, Amsterdam, 1990.

[110] Lai, T. L. and Robbins, H.: *Optimism in the face of uncertainty: Asymptotically efficient adaptive allocation rules*. Advances in Applied Mathematics, 6:4-22, 1985.

[111] Lan, Z.; Taylor, V.E. and Bryan, G.: *Dynamic Load Balancing for Adaptive Mesh Refinement Application*. Proc. Int. Conf. Parallel Processing (ICPP), 2001.

[112] Land, A.H. and Doig, A.G.: *An automated method for solving discrete programming problems*. Econometrica 28 497-520, 1960.

[113] Lawler, E. L. and Wood, D.E.: Branch-and-bound methods: A survey. Operations Research, 14(4):699–719, 1966.

[114] LeCun, Y.; Bottou, L.; Orr, G.B. and Müller, K.R.: *Efficient backprop*. Neural Networks: Tricks of the Trade, number 1524 in LNCS, chapter 1. Springer-Verlag, 1998.

[115] Lee, M.A. and Takagi, H.: *Integrating Design Stages of Fuzzy Systems using Genetic Algorithms*. Proc. of the 2^{nd} Int. Conf. on Fuzzy Systems (FUZZ-IEEE'93), Vol.1, pp. 612-617, 1993.

[116] Liao, W.: *Clustering of time series data - a survey*. Pattern Recognition, vol. 38, pp. 1857-1874, 2005.

[117] Lindeberg, T.: *Scale-space for discrete signals.* PAMI(12), No. 3, pp. 234-254, 1990.

[118] Lindley, D.V.: *Regression and correlation analysis.* New Palgrave: A Dictionary of Economics, v. 4, pp. 120-23, 1987.

[119] Liu, J.; Jin, X. and Wang, Y.: *Agent-Based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization.* IEEE TPDS, volume 16, pp. 586-598, 2005.

[120] Lozano, M.; Herrera, F. and Cano, J.R.: *Replacement strategies to preserve useful diversity in steady-state genetic algorithms.* Inf. Sci., vol. 178, p. 4421-4433, 2008.

[121] Luenberger, D. G. and Ye, Y.: *Linear and nonlinear programming.* International Series in Operations Research & Management Science, New York, Springer, pp. xiv+546, 2008.

[122] Makridakis, S.; Wheelwright, S.C. and Hyndman, R.J.: *Forecasting: Methods and Applications.* John Wiley and Sons, Inc., 1998.

[123] Mandelbaum, A. and Zeltyn, S.: *The Palm/Erlang-A Queue, with Applications to Call Centers.* Advances in Services Innovations, 2005.

[124] Mandic, D. and Chambers, J.: *Recurrent Neural Networks for Prediction: Architectures, Learning algorithms and Stability.* Wiley, 2001.

[125] Massaguer, D.; Balasubramanian, V.; Mehrotra, S. and Venkatasubramanian, N.: *MultiAgent Simulation of Disaster Response.* ATDM Workshop in AAMAS, 2006.

[126] Messerli, E.J.: *Proof of a convexity property of the Erlang B formula.* Bell System Technical Journal 51, 951-953, 1972.

[127] Meyn, S.P. and Tweedie, R.L.: *Markov Chains and Stochastic Stability.* London: Springer-Verlag, 1993.

[128] Micheli, A.: *Neural Network for Graphs: A Contextual Constructive Approach.* IEEE Transactions on Neural Networks, 20:3, 498-511, 2009.

[129] Michie, D.; Spiegelhalter, D.J. and Taylor, C.C.: *Machine Learning, Neural and Statistical Classification.* 1994.

[130] Millán-Ruiz, D. and Hidalgo, J. I.: *A Memetic Algorithm for Workforce Distribution in Dynamic Multi-Skil Call Centres.* Proceedings of the 10th European Conference on Evolutionary Computation in Combinatorial Optimisation (EVO-COP 2010), p. 178-189, Istanbul, Turkey, April 7-9, 2010.

[131] Millán-Ruiz, D.; Pacheco, J.; Hidalgo, J.I. and Vélez, J.L.: *Forecasting in a Multi-skill Call Centre.* Proceedings of the 10th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2010), Zakopane, Poland, June 13-17, 2010.

[132] Miller, B.L. and Goldberg, D.: *Genetic Algorithms, Tournament Selection, and the Effects of Noise.* Report 950006, 1995.

[133] Moscato, P.: *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms.* Caltech Concurrent Computation Program (report 826), 1989.

[134] Muhlenbein, H. and Schlierkamp-Voosen, D.: *Predictive Models for the Breeder Genetic Algorithm.* Evolutionary Computation, 1993.

[135] Müller, H., and Stadtmüller, U. (1987), *Estimation of Heteroscedasticity in Regression Analysis*, The Annals of Statistics, 15, 610-625, 1987.

[136] Nannen, V. and Eiben, A.E.: *A Method for Parameter Calibration and Relevance Estimation in Evolutionary Algorithms*. GECCO'06, July 8-12, Seattle, Washington, USA, 2006.

[137] Nannen, V. and Eiben, A.E.: *Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters*, IJCAI'07, pp. 71-78, 2007.

[138] Nemes S., Jonasson J.M., Genell A. and Steineck G.: *Bias in odds ratios by logistic regression modelling and sample size*. BMC Medical Research Methodology, 2009.

[139] Neri, F. and Cotta, C.: *Memetic algorithms and memetic computing optimization: A literature review*. Swarm and Evolutionary Computation 2:1-14, 2012.

[140] Neri, F.; Cotta, C. and Moscato, P.: *Handbook of Memetic Algorithms*. Studies in Computational Intelligence, Vol. 379, Springer-Verlag, Berlin Heidelberg, 2012.

[141] Nowostawski, M. and Poli, R.: *Parallel Genetic Algorithm Taxonomy*. Proceedings of KES, May, 1999.

[142] Osborn, G.: *Mnemonic for hyperbolic formulae*. The Mathematical Gazette, p. 189, volume 2, issue 34, 1902.

[143] Ozsu, M. T. and Valduriez, P.: *Principles of Distributed Database Systems*. Second Edition, Prentice Hall, ISBN 0-13-659707-6, 1999.

[144] Pacheco, J.; Millán-Ruiz, D. and Vélez, J.L.: *Neural Networks for Forecasting in a Multi-skill Call Centre*. Engineering Applications of Neural Networks, Communications in Computer and Information Science, Volume 43, 291-300, 2009.

[145] Pettey, C.; Leuze, M. and Grefenstette, J.: *A parallel genetic algorithm.* Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, Hillsdale, NJ, USA. Lawrence Erlbaum Associates, 1987.

[146] Phansalkar, V. and Sastry, P.S.: *Analysis of the Back-Propagation Algorithm with Momentum.* IEEE Transactions on Neural Networks, pp. 505-506, 1994.

[147] *R Forecast package.* Available at `http://robjhyndman.com/software/forecast`

[148] Riedmiller, M. and Braun, H.: *Rprop - A Fast Adaptive Learning Algorithm.* Proceedings of the International Symposium on Computer and Information Science VII, 1992.

[149] Rouhana, N. and Horlait, E.: *Dynamic Congestion Avoidance Using Multi-Agents Systems.* Lecture Notes In Computer Science, Vol. 2164, Proceedings of the Third International Workshop on Mobile Agents for Telecommunication Applications, 2001.

[150] *R SNNS package.* Available at `http://cran.r-project.org/web/packages/RSNNS/index.html`

[151] Ruciński, M.; Izzo, D. and Biscani, F.: *On the impact of the migration topology on the Island Model.* Parallel Computing, Vol. 36, pages 555-571, 2010.

[152] Russell, S.J. and Norvig, P.: *Artificial Intelligence: A Modern Approach.* 2nd ed. Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2, chapter 2, 2003.

[153] Sagias, N. C. and Karagiannidis, G. K.: *Gaussian class multivariate Weibull distributions: theory and applications in fading channels.* Institute of Electrical

and Electronics Engineers. Transactions on Information Theory 51 (10): 3608-3619, 2005.

[154] Setzler, H., Saydam, C. and Park, S.: *Call Volume Predictions: A Comparative Study*. Computers & Operations Research, 36, 1843-1851, 2009.

[155] Shabri, A. B.: *Comparison of Time Series Forecasting Methods Using Neural Networks and Box-Jenkins Model*. Matematika, ISSN 01278274, 2001.

[156] Shahsavar, M.; Najafi, A.A. and Niaki, S.T.A.: *Statistical Design of Genetic Algorithms for Combinatorial Optimization Problems*. Mathematical Problems in Engineering, vol. 17, 2011.

[157] Shen, H. and Huang, J.Z.: *Interday Forecasting and Intraday Updating of Call Center Arrivals*. Manufacturing and Service Operations Management, Volume 10, Issue: 3, Pages: 391-410, ISSN: 15234614, 2008.

[158] Shenker, S. and Weinrib, A.: *The Optimal Control of Heterogeneous Queuing Systems: A Paradigm for Load Sharing and Routing*. IEEE Trans. Computers, vol. 38, no. 12, pp. 1724-1735, Dec. 1989.

[159] Silva, F.M. and Almeida, L.B.: *Speeding up backpropagation*. Advanced Neural Computers, pages 151-158. North-Holland, 1990.

[160] Spears, W. and De Jong, K.: *An Analysis of Multi-Point Crossover*. Proc. Foundations of Genetic Algorithms Workshop, 1990.

[161] Sudholt, D.: *Parametrization and Balancing Global and Local Search*. Handbook of Memetic Algorithms, Studies in Computational Intelligence. Eds. F. Neri, C. Cotta, and P. Moscato, Vol. 379, pp. 55-72, 2011.

[162] Sywerda, G.: *Uniform crossover in genetic algorithms.* Proceedings of the third international conference on Genetic algorithms, George Mason University, United States, pp. 2-9, 1989.

[163] Tanese, R.: *Parallel genetic algorithms for a hypercube.* Proceedings of the Second International Conference on Genetic Algorithms, pages 177-83, 1987.

[164] Taylor, J.W.: *Density Forecasting of Intraday Call Center Arrivals Using Models Based on Exponential Smoothing.* Management Science, 2012.

[165] Thompson, G. M.: *Controlling actions in daily workforce schedules.* Cornell Hotel and Restaurant Administration Quarterly, 37, 82-96, 1996.

[166] Thompson, G. M.: *Labor staffing and scheduling models for controlling service levels.* Naval Res. Logist, 719-740, 1997.

[167] Timofeev, A.V.; Syrtzev, A.V. and Kolotaev, A.V.: *Network analysis, adaptive control and imitation simulation for multi-agent telecommunication systems.* Physics and Control, Proceedings International Conference, ISBN 0-7803-9235-3, 2005.

[168] Tollenaere, T.: Supersab: *Fast adaptive backpropagation with good scaling properties.* Neural Net-works, 3:561-573, 1990.

[169] Trehel, M.; Balayer, C. and Alloui, A.: *Modeling Load Balancing Inside Groups Using Queuing Theory.* Proc. 10th Int. Conf. Parallel and Distributed Computing System, Oct. 1997.

[170] Tych, W., Pedregal, D., Young, P. and Davies, J.: *Unobserved Component Model for Multi-rate Forecasting of Telephone Call Demand: The Design of a*

Forecasting Support System. International Journal of Forecasting, 18, 673-695, 2002.

[171] Vaughan, J.G: *Classification of Load Balancing Research in the Absence of Task Precedence Constraints*. Technical Report TR-CS-0001, Department of Computer Science, University College, Cork, Ireland, October, 1985.

[172] Vavak , F. and Fogarty, T.C.: *Comparison of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments*. IEEE International Conference on Evolutionary Computation (ICEC), 1996.

[173] Wei, W.W.: *Time series analysis: Univariate and multivariate methods*. New York: Addison-Wesley, 1989.

[174] Weinberg, J., Brown, L. D., and Stroud, J. R.: *Bayesian forecasting of an inhomegeneous Poisson process with applications to call center Data*. Journal of the American Statistical Association, 102, 1185-1199, 2007.

[175] *R-Weka package*. Available at `http://cran.r-project.org/web/packages/RWeka/index.html`

[176] Whitley, L.D.: *Fundamental Principles of Deception in Genetic Search*. Foundations of Genetic Algorithms, Vol. 1, pages 221-241, 1991.

[177] Whitley, D.; Rana, S. and Heckendorn, R.: *Island model genetic algorithms and linearly separable problems*. Proceedings of AISB, v. 1305, pages 109-125, 1997.

[178] Whitt W.: *Staffing a call center with uncertain arrival rate and absenteeism*. PO&M, 2006.

[179] Wil Michiels, E.A. and Korst, J.: *Theoretical Aspects of Local Search.* Monographs in Theoretical Computer Science. An EATCS Series. Springer 1 edition, ISBN-10: 3540358536, 2007.

[180] Wills, C.: *Rank-Order Selection Is Capable of Maintaining All Genetic Polymorphisms.* Genetics. 89(2): 403-417, 1978.

[181] Wooldridge, M.: *An Introduction to MultiAgent Systems.* John Wiley & Sons Ltd, paperback, 366 pages, ISBN 0-471-49691-X, 2002.

[182] Wright, A.H.: *Genetic Algorithms for Real Parameter Optimization.* Morgan Kaufmann, FOGA, pp. 205-218, 1991.

[183] Zaiyong, T. and Fishwick, P.A.: *Feed-forward Neural Nets as Models for Time Series Forecasting.* ORSA Journal of Computing, 1993.

[184] Zheng, Y. and Xu, R.: *An Adaptive Exponential Smoothing Approach for Software Reliability Prediction.* Wireless Communications, Networking and Mobile Computing, 2008.

Appendix A

Detailed Contribution to the Scientific Literature

I would be remiss if I did not give the proper credit to my co-authors, to whom I am greatly indebted for their collaboration on these works which contributed to my thesis work. To perfectly distinguish my personal contribution, the details of these publications are given below:

1. *Publications in international journals*:
    (a) *Publication*:
        i. *Details*: Millán-Ruiz, D. and Hidalgo, J.I.: *Forecasting Call-Centre Arrivals.* Journal of Forecasting, Wiley, Ed. Derek W. Bunn, 2013.
        ii. *Status*: In press. Accepted on October 20th, 2012.
        iii. *JCR Impact Factor*: 0.930.
        iv. *Summary*: This article presents a novel neural network-based approach to the intra-day forecasting of call arrivals in call centres. We apply the method to individual time series of arrivals for different customer call groups. To train the model, we use historical call data from 3 months and, for each day, we aggregate the call volume in 288 intervals of 5 minutes. With this data, our method can be used for predicting the

call volume in next 5-minute interval using either previous real data or previous predictions to iteratively produce multi-step ahead forecasts. We compare our approach with other conventional forecasting techniques. Experimental results have proven factual evidence in favour of our approach.

v. *Personal contribution*: My contribution was to design a novel learning algorithm for neural networks. Specifically, I designed a self-adaptive algorithm (inspired by the famous "resilient back-propagation") to analyse complex search spaces much faster than other alternatives and usually delivering better results. This algorithm was compared, 30 times, to other 15 forecasting techniques for 5 different problem instances. Note that this algorithm is the core of the predictive module proposed in this dissertation. Section 4.4 presents all these ideas, whereas Section 6.5 shows the comparison in detail.

(b) *Publication*:

i. *Details*: Arnaldo, I., Contreras, I.; Millán-Ruiz, D., Hidalgo, J.I. and Krasnogor, N.: *Matching Island Topologies to Problem Structure in Parallel Evolutionary Algorithms.* International Journal of Soft Computing: Special Issue on Bio-Inspired Algorithms with Structured Populations, Springer, 2013.

ii. *Status*: In press. Accepted on November 16th, 2012.

iii. *JCR Impact Factor*: 1.880.

iv. *Summary*: In the context of Parallel Evolutionary Algorithms, it has been shown that different population structures induce different search performances. Nevertheless, no work has shown clear cut evidence that there is a correlation between the solver's population structure and the problem's network structure. In this work, we verify this correlation performing a clear and systematic analysis of a large set of population structures (based on the well known β-graphs) and NK-Landscape problems. Furthermore, we go beyond our findings in these idealised experiments by analysing the performance of variable-topology EAs on a dynamic real-world problem, the multi-skill call centre.

v. *Personal contribution*: My contribution to this article was focused on the real-world problem: the multi-skill call centre. In order to study the relationship between the problem structure and the best configuration for a parallel evolutionary algorithm, I proposed a self-adaptive meta-genetic algorithm. Afterwards, I carried out a detailed analysis of the results for multiple configurations, linking these results with those obtained for the two theoretical problems, thereby drawing general conclusions from the relationship between the problem structure and the configuration of the islands within a parallel evolutionary algorithm. This algorithm composes the basis of the optimisation module described in this thesis in Section 4.5.

2. *Publications in relevant international conferences*:

   (a) *Publication*:

      i. *Details*: Millán Ruiz, D. and Hidalgo, J.I.: *Migration and Replacement Policies for Preserving Diversity in Dynamic Environments.* Proceedings of the 12th European Conference on the Applications of Evolutionary Computation (EvoApplications 2012, Málaga, Spain, April 11-13, 2012.

      ii. *Status*: Published in 2012.

iii. *Summary*: This paper seeks to resolve the difficulties arising from the configuration and fine-tuning of a Parallel Genetic Algorithm (PGA) based on the Island Model, when the application domain is highly dynamic. This way, the reader will find a number of useful guidelines for setting up a PGA in a real, representative dynamic environment. To achieve this purpose, we examine different (existing and new) migration and replacement policies for three different topologies. Of course, there are many other factors that affect the performance of a PGA such as the topology, migrant selection, migration frequency, amount of migrants, replacement policy, number of processing nodes, synchronism type, configuration in the isolated islands, diversity of policies in different islands, etc., which are also considered as a part of this study. The pivotal point of all the experiments conducted is the preservation of diversity by means of an appropriate balance between exploration and exploitation in the PGA's search process when the application domain is highly dynamic and strong time constraints arise. The experimental phase is performed over two problem instances from a real-world dynamic environment which is the multi-skill call centre.

iv. *Personal contribution*: In addition to the study of diversity maintenance for parallel evolutionary algorithms, I proposed a series of innovative migration and replacement policies, including ideas from other famous meta-heuristics into classic evolutionary operators. These policies and ideas are fully employed in the parallel memetic algorithm (optimisation module) of this thesis (see Section 4.5 and Section 6.6.1).

(b) *Publication*:

i. *Details*: Millán Ruiz, D., Hidalgo, J.I.: *Comparison of Metaheuristics* *for Workforce Distribution in Multi-Skill Call Centres.* Proceedings of the International Joint Conference on Computational Intelligence (ICEC 2010), Valencia, Spain, October 24-26, 2010.

ii. *Status*: Published in 2010.

iii. *Summary*: Call centre technology requires the assignment of a large volume of incoming calls to agents with the required skills to process them. In order to determine the right assignment among incoming calls and agents for a real production environment, a comparative study of meta-heuristics has been carried out. The aim of this study is to implement and empirically compare various representative meta-heuristics, which represent distinct search strategies to reach accurate, feasible solutions, for two different instances of the workforce distribution problem. This study points out how memetic algorithms can outperform other acknowledged meta-heuristics for two different problem instances from a real multi-skill call centre from one of the world's largest telecommunications companies.

iv. *Personal contribution*: My contribution was to implement and compare different meta-heuristics with the memetic algorithm that I developed for a previous publication. This comparison is fully included in this thesis in order to demonstrate that our memetic algorithm can outperform other meta-heuristics to solve our problem (see Section 6.6.2).

(c) *Publication*:

i. *Details*: Millán Ruiz, D., Hidalgo, J.I.: *A Parallel Memetic Algorithm for Dynamic Workload Distribution in Multi-Agents Systems.* Proceedings of the 3rd Workshop on Parallel Architectures and Bioinspired

Algorithms held In conjunction with PACT 2010, Vienna, Austria, September 11-15, 2010.

ii. *Status*: Published in 2010.

iii. *Summary*: This paper describes a parallel evolutionary approach to the problem of workload distribution in dynamic multi-agent systems based on blackboard architectures. Specifically, we focus on the multi-skill call centre use case. This type of call centres entails quick adaptations to a changing environment that only some greedy algorithms have been able to cope with. These greedy heuristics consist of a continuous re-planning, considering the current state of the system. As these decisions are greedily taken, the workload distribution may be poor for middle and/or long term planning due to incessant wrong movements. The use of parallel memetic algorithms, which are much more complex than classical, ad-hoc heuristics, can guide us towards more accurate and robust solutions. Now, the difficulty underlies in how to apply these techniques to uncertain, ever-changing environments. Specifically, in previous studies, we proposed a neural network to make accurate predictions and a single memetic algorithm as a heuristic optimisation mechanism, improving the results obtained by other well-known techniques of the call centre domain. In this study, we upgrade our approach by parallelising the memetic algorithm and carrying out a deeper analysis.

iv. *Personal contribution*: In this paper, I proposed a complete method for workforce distribution in multi-agent systems, which covered the ideas of previous publications (predictive module + optimisation module), explaining how to combine those components. Apart from linking previous components, I put forward an initial mechanism for the adaptive time-frame (a new feature) and a parallel version for the optimisation module (up until now, the memetic algorithm was sequential). The whole system is the basis of Chapter 4 of this thesis.

(d) *Publication*:

   i. *Details*: Millán Ruiz, D., Pacheco, J., Hidalgo, J.I., Vélez, JL: *Forecasting in a multi-skill Call Centre*. Proceedings of the 10th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2010), Zakopane, Poland, June 13-17, 2010.

   ii. *Status*: Published in 2010.

   iii. *Summary*: Call centre technology is subject to multiple improvements and innovations. Some of them try to improve agent performance and client satisfaction. There are two different but intrinsically linked problems. The first one is related to predictions of call arrivals, call abandonment, available agents having a certain skill, call pick-up and average delay time; and the second one deals with workload distribution within a multi-skill Call Centre. In this paper, we focus on forecasting call arrivals, which can be approached from several angles. Concretely, we analyse and compare neural networks, time series and regression models in this study.

   iv. *Personal contribution*: I conducted a comprehensive study of the state-of-the-art on call arrival forecasting techniques (neural networks, time series and regression models). This study has subsequently been extended and then included in this thesis (see Section 6.5).

(e) *Publication*:

i. *Details*: Millán Ruiz, D., Hidalgo, J.I.: *A Memetic Algorithm for Workforce Distribution in Dynamic Multi-Skill Call Centres.* Proceedings of the 10th European Conference on Evolutionary Computation in Combinatorial Optimisation (EVOCOP 2010), p. 178-189, Istanbul, Turkey, April 7-9, 2010.

ii. *Status*: Published in 2010.

iii. *Summary*: In this paper, we describe a novel approach for workforce distribution in dynamic multi-skill call centres. Dynamic multi-skill call centres require quick adaptations to a changing environment that only fast greedy heuristics can handle. The use of memetic algorithms, which are more complex than ad-hoc heuristics, can guide us to more accurate solutions. In order to apply memetic algorithms to such a dynamic environment, we propose a reformulation of the traditional problem, which combines predictions of future situations with a precise search mechanism, by enlarging the time-frame considered. Concretely, we propose a neural network for predicting call arrivals and the number of available agents, and a memetic algorithm to carry out the assignment of incoming calls to agents, which outperforms classical approaches to this dynamic environment. We also test our method on a real-world environment within a large multinational telephone operator.

iv. *Personal contribution*: For this article, I developed the first version of the optimisation module, consisting of a simple memetic algorithm. I also implemented (or adapted) other conventional call-centre techniques (e.g. skill-based routing, Thomson's algorithm and Koole's method) to perform the comparison. Although the results presented in this paper outperformed other call-centre techniques, I improved many aspects of the algorithm (see recent publications). All these changes are fully described in Section 4.4 and Section 6.7 of this dissertation.

(f) *Publication*:

i. *Details*: Meléndez, J., Lopez, B., Millán-Ruiz, D.: *Probabilistic models to assist maintenance of multiple instruments.* Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2009), p. 1499-1503, Palma de Mallorca, Spain, September 22-26, 2009.

ii. *Status*: Published in 2009.

iii. *Summary*: The paper discusses maintenance challenges of organisations with a huge number of devices and proposes the use of probabilistic models to assist monitoring and maintenance planning. The proposal assumes connectivity of instruments to report relevant features for monitoring. Also, the existence of enough historical registers with diagnosed breakdowns is required to make probabilistic models reliable and useful for predictive maintenance strategies based on them. Regular Markov models based on estimated failure and repair rates are proposed to calculate the availability of the instruments and Dynamic Bayesian Networks are proposed to model cause-effect relationships to trigger predictive maintenance services based on the influence between observed features and previously documented diagnostics.

iv. *Personal contribution*: My main contribution was the creation of probabilistic models for predictive maintenance. Although these probabilistic methods have not been fully applied to this dissertation, the method for determining the dynamism of the system (size of the prediction window and the transitions between system states) is based on the regular Markov models proposed in this paper. Therefore, those ideas are important inputs for the Chapter 4 of this thesis.

(g) *Publication*:

i. *Details*: Pacheco, J., Millán-Ruiz, D., Vélez, JL: *Neural Networks for Forecasting in a multi-skill Call Centre.* Proceedings of the 11th International Conference on Engineering Applications of Neural Networks (EANN 2009), p. 291-300, London, UK, August 27-29, 2009.

ii. *Status*: Published in 2009.

iii. *Summary*: Call centre technology requires the assignment of a large volume of incoming calls to agents with the required skills to process them. In order to perform the right assignment of call types to agents in a production environment, an efficient prediction of call arrivals is needed. In this paper, we introduce a prediction approach to incoming phone calls forecasting in a multi-skill call centre by modelling and learning the problem with an improved back-propagation Neural Network which have been compared with other methods. This model has been trained and analysed by using a real-time data flow in a production system from our call centre, and the results obtained outperform other forecasting methods. The reader can learn which forecasting method to use in a real-world application and some guidelines to better adapt an improved back-propagation neural network to his needs. A comparison among techniques and some statistics are shown to corroborate our results.

iv. *Personal contribution*: In this article, I developed a neural network learning algorithm, specifically adapted to highly changing environments. I also performed a batch of experiments that included a short list of predictive techniques. Although this neural network is not the final version that is included in this thesis (but the version described in the previously mentioned journal article), it was used as the starting point for this work and still shares some common aspects and parameters with the newest version (see Section 4.4).

(h) *Publication*:

   i. *Details*: Martinez-Lopez, R., Millán-Ruiz, D., Martin-Dominguez, A.; Toro-Escudero, MA: *An Architecture for Next-Generation of Telecare Systems Using Ontologies, Rules Engines and Data Mining.* Proceedings of the International Conferences on Computational Intelligence for Modelling, Control and Automation, Intelligent Agents, Web Technologies and Internet Commerce, and Innovation in Software Engineering (CIMCA 2008), p. 31-36, Vienna, Austria, December 10-12, 2008.

   ii. *Status*: Published in 2008.

   iii. *Summary*: The nonstop ageing of population in Europe makes the health system maintenance a worrying and complicated task for every government. In this context, telecare systems emerge as a cheap and effective approach. A telecare system helps elderly and care-dependent people to satisfy their needs and special requirements, requiring fewer resources and enabling users to be at their own homes. The purpose of this paper is to define a telecare Conceptual System Architecture based on ontologies, rules and inference engines, machine learning techniques and data mining procedures. Some open sources are also proposed in order to develop certain modules of this architecture. The approach tries to provide a representation of an entire telecare system, offering customised solutions for users, professional care facilities and service centres.

iv. *Personal contribution*: My main contribution focused on the data mining and machine learning pieces. Specifically, I designed a generic-purpose neural network that allowed for predicting the nervousness level of the patients, according to their behaviour. Although the learning algorithm was a simple back-propagation, the structure of the neural network was then adapted to the call centre environment in other publications.

3. *Publications in national journals and conference*:

   (a) *Publication*:

      i. *Details*: Millán Ruiz, D., Hidalgo, J.I.: *Algoritmo memético paralelo para la distribución de esfuerzo en centros de llamadas dinámicos multi-agente y multi-tarea.* Proceedings of the 7th Spanish Conference on Meta-heuristics, Evolutionary Algorithms and Bioinspired Algorithms (MAEB 2010), Valencia, Spain, September, September 8-10, 2010.

      ii. *Status*: Published in 2010.

      iii. *Summary*: In this paper, we describe a parallel approach to the problem of workforce distribution in multi-skill call centres. This type of call centres requires adaptive techniques, because they are changing environments. In previous studies, we proposed a new neural network for making predictions and a sequential memetic algorithm for assigning tasks to agents. In this study, we present an extension of the memetic algorithm (by means of an island model). Experimental results show those improvements.

iv. *Personal contribution*: In this article, my contribution was focused on showing the advantages of a parallel evolutionary approach as compared to a sequential evolutionary algorithm. These ideas were extended in a subsequent publication where I performed a deeper analysis. These results are also reflected in this thesis in Chapter 6.

Appendix B

Resumen Extendido en Español

Esta tesis describe un novedoso enfoque para resolver el problema de distribución de carga de trabajo en sistemas multi-agente dinámicos basados en arquitecturas de pizarra, enfocándose especialmente en un escenario real: el call center multi-tarea.

Para abordar este tipo de entornos dinámicos, tradicionalmente se han aplicado diversas heurísticas voraces que permiten dar una solución en tiempo real. Básicamente, dichas heurísticas realizan replanificaciones continuamente, considerando el estado del sistema en cada momento. Como las decisiones se toman de forma voraz sin hacer una planificación óptima, la distribución de la carga de trabajo puede ser pobre a medio y/o largo plazo.

El uso de algoritmos meméticos paralelos nos puede permitir encontrar soluciones mucho más precisas. Para aplicar este tipo de algoritmos, introducimos el concepto de ventana temporal adaptativa.

De esta forma, el tamaño de la ventana temporal depende del nivel de dinamismo del sistema en un instante dado. Este trabajo propone una serie de herramientas para determinar el dinamismo del sistema de forma automática, así como un novedoso módulo de predicción basado en una red neuronal y un potente método de búsqueda basado en meta-algoritmos meméticos paralelos para poder lidiar con en-

tornos dinámicos complejos. Para concluir, comparamos nuestro enfoque con otras técnicas del estado del arte en un entorno de producción real (Telefónica) obteniendo mejores resultados que el resto de técnicas actuales. También se proporciona un estudio exhaustivo de cada uno de los módulos.

Introducción

A lo largo de los últimos años, se puede percibir una tendencia orientada a resolver problemas de complejidad creciente, así como a trabajar en dominios de aplicación que, frecuentemente, involucran el cómputo de flujos de datos continuos y notablemente dinámicos. Estos arduos entornos son, habitualmente, difíciles de mantener por técnicas más convencionales.

Un problema clásico, inherentemente enmarcado dentro de este ámbito, es el estudio de **distribución de esfuerzo en sistemas multi-agente dinámicos**, como puede ser un centro de llamadas multi-agente y multi-tarea.

La variante básica del problema abordado puede formularse como una asignación dinámica de tareas a los agentes que tienen la capacidad de llevarlas a cabo a medida que transcurre el tiempo, respetando una serie de restricciones que pueden ser temporales, organizativas, legales o de cualquier otra índole.

Dicho esto, el estado del arte distingue una amplia gama de variantes de este problema aunque, dependiendo del dinamismo del sistema, podemos identificar dos grandes vertientes o escenarios: entornos dinámicos o entornos con una cierta estabilidad.

Por un lado, encontramos entornos dinámicos en los que se precisa una planificación a corto plazo y que, en consecuencia, imponen una continua re-planificación de tareas y recursos. Las soluciones ofrecidas para estos sistemas tienen como principal propósito el distribuir la carga de trabajo entre los agentes disponibles, aplicando heurísticas eminentemente básicas que consideran, tan sólo, el estado actual del entorno en cuestión. Este rasgo puede apreciarse en la idiosincrasia de los centros de llamadas multi-agente y multi-tarea altamente dinámicos.

Por otro lado, podemos toparnos con una variedad de entornos más estables y que, como resultado, permiten trabajar con ventanas de planificación a más largo plazo. Este tipo de entornos se puede descomponer, a su vez, en diversos grupos. Así pues, existen entornos en los que la lista de tareas (pool en inglés) es fija y conocida a priori por todos los agentes, como podría ser el caso del problema clásico de la planificación de tareas (scheduling, en inglés). También, hay entornos en los que una única tarea es asignada a cada agente durante un largo periodo de tiempo, como puede ser el caso del problema de asignación de trabajos (job assignment problem, en inglés). En otros casos, los agentes son asignados a patrones de tareas en base a sus habilidades, en lugar de ligarlos a tareas concretas (véase pattern-based scheduling, en inglés). De forma análoga, podemos incluir en este bloque a los centros de llamadas más estables (por ejemplo, aquellos que están destinados a servicios muy específicos como ocurre con los centros de soporte técnico).

De esta forma, podemos afirmar que todos los casos presentados reflejan una misma característica común: se trata de entornos estables y/o prefijados y que, por tanto, permiten que las heurísticas puedan ser más complejas y se tomen más tiempo a la hora de ofrecer una solución factible.

Enlazando con las ideas anteriores, este trabajo aborda los entornos encasillados en el primer grupo de variantes; dicho con otras palabras, vamos a proponer una solución para entornos dinámicos (aunque nos centraremos esencialmente en el caso de uso del centro de llamadas dinámico multi-agente y multi-tarea) basada en algoritmos predictivos y meméticos paralelos, utilizando una ventana temporal adaptativa.

La Tabla 7.1 resume algunas características fundamentales de los entornos descritos en los párrafos anteriores, en función de la ventana temporal considerada.

Table 7.1: Comparativa del problema de distribución de esfuerzo en función del tamaño de la ventana.

Tam. de la ventana	*Complejidad*	*Tiempo de respuesta*	*Adaptabilidad*	*Rendimiento*	*Uso de CPU*
Corto plazo	baja	bajo	media	medio	bajo
Medio plazo	alta	medio	alta	alto	alto
Largo plazo	media	alto	baja	bajo	medio

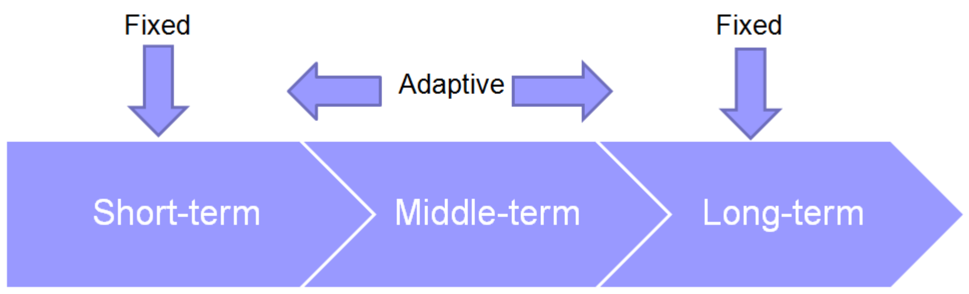

Figure 7.1: Mecanismo de ventana adaptativa.

La figura 7.1 muestra dónde se posiciona nuestro mecanismo de ventana adaptativa:

La hipótesis que subyace en la solución propuesta puede resumirse como: “en sistemas multi-agente altamente dinámicos es preferible hacer planificaciones más exactas a medio plazo (asumiendo un cierto ruido) que hacerlas de peor calidad, reiteradamente, a corto plazo”.

Y es que las heurísticas ad-hoc consideran el estado actual del sistema pero, debido a restricciones temporales, tienen que hacer operaciones muy simples para después enrutar cada tarea hasta al agente correspondiente (o grupos de agentes), en un tiempo cercano al real. Sin embargo, si conseguimos hacer predicciones de estados futuros a medio plazo con suficiente exactitud, podemos aplicar técnicas más complejas que permiten obtener soluciones suficientemente precisas para dichos estados.

Obviamente, nos estamos perdiendo algunas “fotos” de estados intermedios por el camino, pero si los “saltos” son suficientemente pequeños, podemos suponer una cierta estacionalidad en la que el cambio del sistema es mínimo y una misma solución es, potencialmente, factible durante ese intervalo. Dependiendo del dinamismo del sistema, los intervalos deben tener mayor o menor amplitud y pueden ser automáticamente determinados en función de la variabilidad de las predicciones (error en t-v, siendo v el tamaño de la ventana de predicción expresado en segundos).

Motivación y Objetivos Principales

El problema de la distribución de esfuerzo en sistemas multi-agente es un tema atractivo y desafiante a nivel de investigación, no sólo desde un punto de vista técnico, sino también desde una perspectiva de negocio. La eminente complejidad de este problema hace que sea aún más interesante y un miembro sólido dentro de la clase de problemas NP-duros. Otra de las razones para analizar este problema es que se detecta en muchos escenarios de nuestra vida cotidiana y es muy relevante para muchas aplicaciones industriales.

Visto desde un ángulo de computación paralela, este problema también es tentador ya que permite realizar un procesamiento paralelo de forma inherente. De esta forma, las tareas se pueden distribuir a través de varios nodos de procesamiento.

Bajo el punto de vista de la inteligencia artificial, este problema también es muy motivador porque involucra muchos campos, que van desde técnicas de predicción derivadas de la teoría de aprendizaje automático, hasta algoritmos de optimización que utilizan técnicas de mantenimiento de la diversidad (computación evolutiva) y otros sistemas de búsqueda local como el enfriamiento simulado o la búsqueda tabú.

El objetivo principal de este trabajo consiste en proporcionar una solución para realizar una correcta distribución de esfuerzo en sistemas multi-agente dinámicos basados en arquitecturas de pizarra. Por lo tanto, se ha de proporcionar un método predictivo eficiente, con el fin de predecir la situación real en la siguiente ventana de trabajo (futuro estado del sistema) y también se ha de realizar un algoritmo de optimización adaptativo con el propósito de determinar la asignación correcta <tarea-agente>.

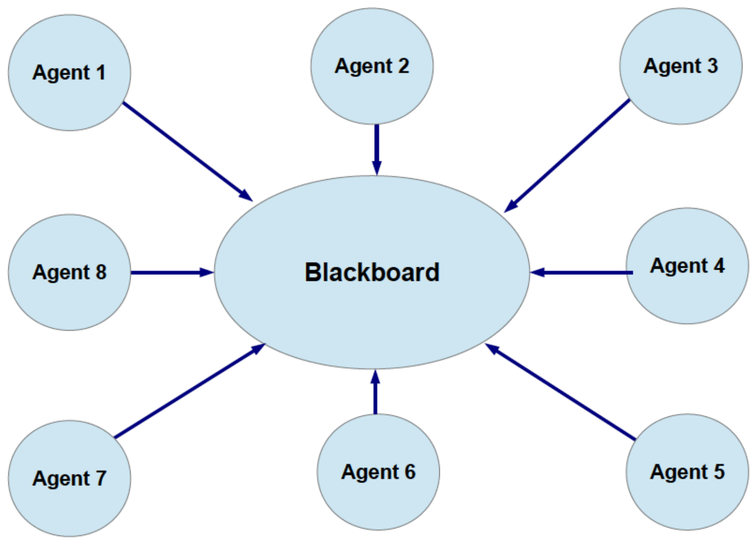

Figure 7.2: Ejemplo de una arquitectura de pizarra con *8* agentes.

Descripción del problema

Un centro de llamadas (CLL) es una oficina centralizada destinada a recibir y transmitir vastos volúmenes de llamadas a través del teléfono. En un CLL, el flujo de llamadas se suele descomponer en tráfico entrante y saliente. En este trabajo, nos centraremos en el tráfico entrante pues, a diferencia del saliente, no está planificado. El tráfico entrante se modela y clasifica en grupos de llamadas (GLL).

Cuando el CLL tiene más de un GLL y diversos operadores para atender las llamadas, se denomina CLL multi-agente y multi-tarea (CLLMM, en inglés, multi-skill call centre). En un CLLMM, podemos encontrar n llamadas de clientes agrupadas en k tipos de llamadas (GLL) y m operadores o agentes dispuestos a contestar dichas llamadas, siempre y cuando tengan las habilidades necesarias para atender esos GLLs.

La Figura 1 muestra la relación existente entre llamadas, GLL y agentes.

Dando un poco de rigor a la formulación del problema, podemos distinguir los siguientes elementos dentro de un CLLMM:

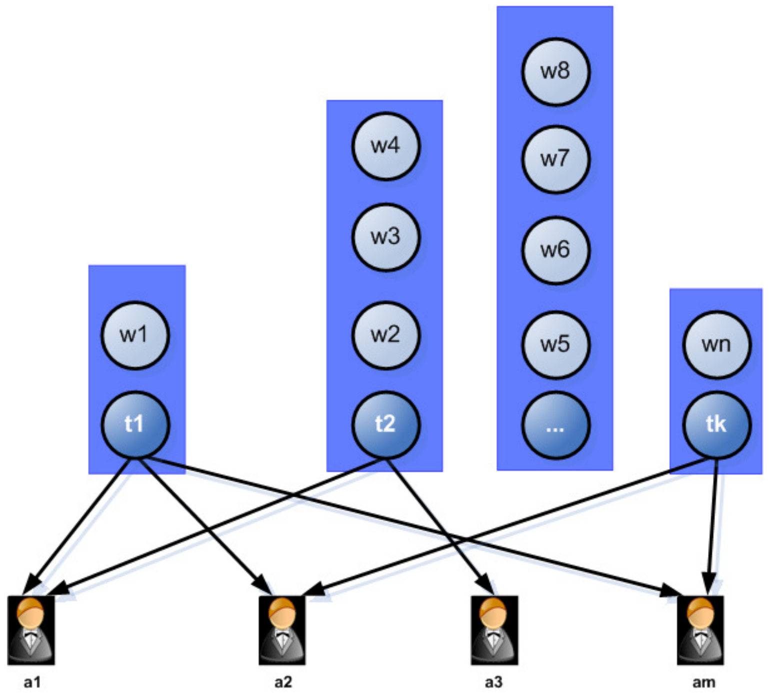

Figure 7.3: Esquema del tráfico entrante en un CLLMM.

- un conjunto finito de n llamadas (c_i) de clientes $C = \{c_1, c_2, ..., c_n\}$.

- un conjunto finito de k grupos de llamadas $GLL = \{gll_1, gll_2, ..., gll_k\}$, donde $k \leq n$ cuando cada GLL tiene, al menos, una llamada asignada.

- un conjunto finito de m agentes $A = \{a_1, a_2, ..., a_m\}$.

- un conjunto finito de k habilidades de agente $S = \{s_1, s_2, ..., s_k\}$, donde cada habilidad s_i representa la capacidad que tiene para atender el correspondiente GLL gll_i (mismo subíndice): $s_1 \sim gll_1, s_2 \sim gll_2, ..., s_k \sim gll_k$.

- un conjunto finito de d perfiles de habilidad $P = \{P_1, P_2, ..., P_d\}$, donde cada perfil de habilidad P_i es un conjunto finito de l habilidades de agente.

- un conjunto de n operaciones (ejecución o procesamiento de cada llamada C_i) $O = \{o_1, o_2, ..., o_n\}$, donde cada operación tiene asociada un tiempo de procesamiento que depende del GLL: $T = \{t_1, t_2, ..., t_k\}$.

Además, la solución debe satisfacer las siguientes condiciones:

- en O se define una relación binaria A que representa la precedencia entre operaciones. Si $(o_i, o_j) \epsilon A$, entonces o_1 tiene que realizarse antes que o_2.

- cada agente a_i tiene asociado un subconjunto finito no nulo de b elementos $P^b \subseteq P$ que contiene las habilidades que tiene a_i para atender diversos GLLs: $a_i \sim \{P^1, P^2, ..., P^b\}$.

- un mismo perfil P^i puede estar asignado a varios agentes (los que tengan esas destrezas) y, asimismo, cada agente a_i puede tener diferentes perfiles en función de sus destrezas $P^b \subseteq P$, aunque sólo puede tener un único perfil asignado en un instante t dado $< a_i, P_j >_t$.

- cada solución debe respetar las restricciones duras del problema e intentar evitar violar las restricciones blandas.

El objetivo consiste en obtener la mejor asignación $< a_i, P_j >_t$ para cada ventana temporal y que respete las restricciones del problema.

Para evaluar la calidad de cada solución obtenida, vamos a considerar el *nivel de servicio* que se define como el porcentaje de llamadas atendidas antes de una determinada cantidad de tiempo de espera (en nuestro caso será de *20 segundos*).

Como se puede intuir, la complejidad de este problema es enorme, pues no sólo tenemos una explosión combinatoria entre perfiles de habilidad, número de llamadas y de agentes, sino que además tenemos poco tiempo para dar una respuesta.

Detalle de la Solución

Anteriormente hemos visto cómo las técnicas de planificación a corto plazo distribuyen vorazmente (sin planificación) las tareas entre los agentes disponibles que pueden procesar dichas tareas, mientras que las técnicas de planificación a largo plazo trabajan con un flujo estable en el tiempo (algo que está fuera del ámbito de este trabajo).

De esta manera, las estrategias de planificación a corto plazo distribuyen el esfuerzo sin tener en cuenta el estado del sistema en el futuro (sólo la situación global actual), haciendo que las asignaciones *tarea-agente* sean inadecuadas en el medio y largo plazo.

Por el contrario, las estrategias de planificación a largo plazo proporcionan soluciones óptimas para un determinado estado del sistema, requiriendo tiempos de cálculo más largos. Sin embargo, si el sistema no es muy estable, podríamos tener serios problemas en el futuro, debido a que una configuración óptima para el estado actual del sistema puede no ser la mejor opción para un futuro punto en el tiempo.

La hipótesis subyacente de este trabajo es que la mayoría de sistemas multiagente dinámicos requieren asignaciones precisas de las tareas entrantes a los agentes disponibles más adecuados, teniendo en cuenta una ventana temporal auto-adaptativa en lugar asignaciones no planificadas continuamente o asignaciones estáticas en el tiempo.

En sistemas muy dinámicos, es preferible hacer una planificación a medio plazo, asumiendo cierto ruido en las predicciones, que hacer una mala planificación de forma repetida en el tiempo.

Básicamente, tenemos que ampliar (o reducir) la ventana de trabajo observada y pronosticar el estado real del sistema en un punto de tiempo futuro con el fin de aplicar algoritmos de búsqueda más sofisticados, los cuales pueden superar a las técnicas de planificación a corto plazo convencionales.

Ahí es donde surge la necesidad de realizar una predicción exacta de un estado del sistema futuro a medio plazo. Posteriormente, un potente algoritmo de búsqueda debe encontrar una solución viable para dicho estado futuro, asegurando de este modo un justo equilibrio entre diversidad (exploración) e intensidad (explotación).

El módulo de predicción se basa en una red neuronal que implementa un novedoso algoritmo de aprendizaje, mientras que el módulo de optimización se basa en un algoritmo memético paralelo que utiliza un algoritmo meta-genético externo para calibrar automáticamente los parámetros internos del algoritmo memético paralelo en función de la dinámica del sistema. A lo largo de esta sección, vamos a ir detallando rápidamente los componentes de nuestro enfoque.

Módulo predictivo

El módulo predictivo trata de predecir las variables que no están prefijadas en nuestro problema. En este caso, nos referimos a las llamadas entrantes y abandonos, y al número de agentes disponibles.

La forma de operar de este módulo consiste en realizar estimaciones del número de llamadas entrantes que habrán llegado a lo largo del próximo intervalo y de los abandonos que habrá para que, sumándole el número de llamadas encoladas, obtengamos el total de llamadas de cada tipo que tendremos en un estado futuro.

Asimismo, predeciremos el número de agentes disponibles en esos estados futuros (esta tarea es más fácil ya que sabemos los horarios a priori y sólo hay que predecir pequeños desvíos).

El algoritmo que proponemos se basa en Rprop y las modificaciones que hemos realizado quedan reflejadas en el siguiente pseudo-código:

$$\text{si } \left(\frac{\partial \Lambda\epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda\epsilon(t-1)}{\partial w_{ij}} > 0\right) \Rightarrow$$

$$\Rightarrow \begin{cases} \Delta_{ij}(t) = \min(\gamma \cdot \Delta_{ij}(t-1), \Delta_{max}) \\ \\ \Delta w_{ij}(t) = \begin{cases} \Delta_{ij}(t) & \text{si } (\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} < 0) \\ -\Delta_{ij}(t) & \text{si } (\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} > 0) \\ 0 & \text{otro caso} \end{cases} \\ \\ w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \\ \\ \frac{\partial \Lambda \epsilon(t-1)}{\partial w_{ij}} = \frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} \end{cases} \tag{7.1}$$

en caso contrario, si $(\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda \epsilon(t-1)}{\partial w_{ij}} < 0) \Rightarrow$

$$\Rightarrow \begin{cases} \Delta_{ij}(t) = \max(\frac{\Delta_{ij}(t-1)}{\mu}, \Delta_{min}) \\ \\ \Delta w_{ij}(t) = \begin{cases} -\frac{\Delta w_{ij}(t-1)}{2} & \text{si} \begin{cases} (\Lambda\epsilon(t) > \Lambda\epsilon(t-1)) \\ \&\& \\ (\Lambda\epsilon(t) < \alpha \cdot \Lambda\epsilon(t-1)) \end{cases} \\ -\Delta w_{ij}(t-1) & \text{si } \Lambda\epsilon(t) > \alpha \cdot \Lambda\epsilon(t-1) \\ 0 & \text{otro caso} \end{cases} \\ \\ w_{ij}(t+1) = \begin{cases} w_{ij}(t) - \frac{\Delta w_{ij}(t-1)}{2} & \text{si} \begin{cases} (\Lambda\epsilon(t) > \Lambda\epsilon(t-1)) \\ \&\& \\ (\Lambda\epsilon(t) < \alpha \cdot \Lambda\epsilon(t-1)) \end{cases} \\ w_{ij}(t) - \Delta w_{ij}(t-1) & \text{si } \Lambda\epsilon(t) > \alpha \cdot \Lambda\epsilon(t-1) \\ w_{ij}(t) & \text{otro caso} \end{cases} \\ \\ \frac{\partial \Lambda\epsilon(t)}{\partial w_{ij}} = 0 \end{cases} \quad (7.2)$$

en caso contrario, si $(\frac{\partial \Lambda\epsilon(t)}{\partial w_{ij}} \cdot \frac{\partial \Lambda\epsilon(t-1)}{\partial w_{ij}} = 0) \Rightarrow$

$$\Rightarrow \begin{cases} \Delta_{ij}(t) = \Delta_{ij}(t-1) \\ \\ \Delta w_{ij}(t) = \begin{cases} \Delta_{ij}(t) & \text{si } (\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} < 0) \\ -\Delta_{ij}(t) & \text{si } (\frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} > 0) \\ \Delta_{ij}(t-1) & \text{otro caso} \end{cases} \\ \\ w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \\ \\ \frac{\partial \Lambda \epsilon(t-1)}{\partial w_{ij}} = \frac{\partial \Lambda \epsilon(t)}{\partial w_{ij}} \end{cases} \qquad (7.3)$$

Hay que destacar que nuestro algoritmo puede caer en óptimos locales pero se puede entrenar de forma rápida y sencilla.

Módulo de optimización heurística

El módulo de optimización heurística parte de las predicciones del módulo predictivo. Como se ha comentado, el módulo de optimización consiste en un algoritmo memético paralelo. Este algoritmo memético está formado, a su vez, por islas donde cada isla es un algoritmo genético con una búsqueda local sencilla que refina los resultados cada cierto número de generaciones.

Las siguientes subsecciones explican cómo está desarrollado el algoritmo memético (a nivel de operadores) de cada isla, así como la búsqueda local que refina las soluciones en cada isla. Por último, se explica brevemente cómo funciona el algoritmo memético paralelo compuesto por dichas islas.

Codificación

El primer paso cuando se está diseñando un algoritmo memético, consiste en buscar una representación y codificación para el problema. Hay diversas formas de representar este problema ya que el cromosoma puede contener las tareas o los agentes, pero propondremos una representación de tipo entero donde cada posición del cromosoma representa a un agente y su contenido el perfil que tiene asignado en ese momento de entre los posibles perfiles que podría tener.

Si ponemos en cada posición la tarea y en el contenido cuántos agentes debería tener, también sería una representación viable pero nos permitiría tener menos flexibilidad a la hora de jugar con los perfiles de habilidad de cada agente. En cambio, los cromosomas serían de mayor tamaño ya que suele haber más agentes que tipos de tareas. No obstante, recomendamos fehacientemente estudiar el entorno en cuestión si se tienen pocos grupos e infinidad de agentes, en cuyo caso recomendamos la segunda opción. La opción que usaremos en este trabajo será la primera.

La Figura 7.4 muestra un ejemplo ficticio, a modo de ejemplo, con *10* llamadas ($c_0 - c_9$) agrupadas en *3* GLLs distintos ($gll_0 - gll_2$), *5* agentes ($a_0 - a_4$) y *4* perfiles de habilidad ($P_0 - P_3$), donde $P_0 = \{s_0, s_1\}, P_1 = \{s_1\}, P_2 = s_2$ y $P_3 = \{s_1, s_2\}$. Ahora, suponga que $a_0 \sim \{P_0, P_1\}, a_1 \sim \{P_0, P_2\}, a_2 \sim \{P_1, P_3\}, a_3 \sim \{P_2, P_3\}$ y $a_4 \sim \{P_0, P_1\}$.

Dada esta configuración y teniendo en cuenta que cada agente a_i sólo pueden tener asignado un perfil P_i en un instante dado; una solución factible podría ser la mostrada en la Figura 7.4.

Función de evaluación

La función de evaluación, como es bien sabido, es el mecanismo que permite evaluar la calidad de una solución dada. Esta función guía la búsqueda y decide qué individuos

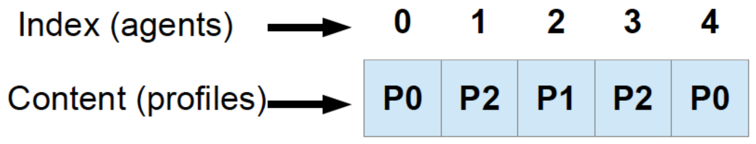

Figure 7.4: Ejemplo de codificación.

deben ser seleccionados para la próxima generación (de hecho, también depende de la política de reemplazo). La función de evaluación está íntimamente ligada al problema y es muy importante definirla correctamente.

La función de evaluación que proponemos está inspirada en la estimación del nivel de servicio total, aunque nosotros consideramos la prioridad de cada grupo ponderada por la carga real y verificamos si las restricciones se satisfacen o no. El nivel de servicio total modificado puede expresarse como:

$$NiveldeServicioTotal = \sum_{i=0}^{k}(Pr_i \cdot NS_i(\gamma_i, \alpha_i)) \cdot \mu \quad \{NS : \Re X[0,1]X[0,1] \rightarrow [0,1]\} \tag{7.4}$$

donde k hace referencia al número de GLLs, μ es un factor de normalización $(\frac{1}{\sum_{i=0}^{k} Pr_i})$, Pr_i es la prioridad del gll_i ponderado por su número de llamadas, y NS_i se define como el nivel de servicio del gll_i:

$$NS_i(\gamma_i, m_i) = 1 - P(Agentes\_ocupados) \cdot e^{-(\gamma_i - m_i)\frac{\tau_i}{\beta}} \tag{7.5}$$

dado que:

$$P(Agentes\_ocupados) = \left[1 + \frac{\gamma_i - m_i}{m_i} \cdot \sum_{\zeta=0}^{\gamma_i - 1} \frac{(\gamma_i - 1)...(\zeta + 1)}{m_i^{\gamma_i - \zeta - 1}}\right]^{-1} \tag{7.6}$$

P(Agentes ocupados) es la probabilidad de que todos los agentes con esa habilidad estén ocupados, γ_i i es la carga entrante del gll_i $(n_i \cdot \tau_i)$, m_i es el número de agentes asignados en el cromosoma al gll_i, τ_i es el tiempo de procesamiento medio de una llamada del gll_i, y β es el tiempo, expresado en segundos, en el que, si la llamada se atiende antes, se considera como exitosa, bajo un punto de vista de atención al cliente.

Adicionalmente, consideramos que puede haber restricciones duras y blandas en función de la relevancia de las mismas (en nuestro caso vendrán dadas por las unidades de negocio). Estas restricciones van ligadas a las tareas, a los agentes, al tiempo y acciones que pueden ser no deseadas.

Así pues, ni el algoritmo ni sus operadores podrán violar las restricciones duras (e.g. un agente no puede tener habilidades asignadas que no estén dentro de su conjunto de habilidades potenciales) y se penalizarán en mayor o menor medida si incumplen alguna restricción débil (e.g. en la medida de lo posible, no moveremos agentes de un grupo a otro de menor prioridad, si el de mayor prioridad no ha alcanzado un nivel mínimo de nivel de atención deseado).

Como las restricciones débiles nos vendrán dadas por reglas de negocio, podemos establecer niveles de restricciones y, dentro cada nivel, un orden. Para el salto entre niveles se ha empleado una función logarítmica, siendo el *nivel 0* el de máxima prioridad y el *nivel j*, el de menos. Con esto, nuestra función de evaluación queda como:

$$f = (NiveldeServicioTotal - penalizaciones) \quad f : [0,1]x[0,1] \rightarrow [-1,1] \qquad (7.7)$$

Finalmente, vamos a introducir el concepto de evaluación parcial para acelerar las evaluaciones, sobre todo, al ejecutar la búsqueda local. En cada generación, se aplicará la función f pero en el resto de ocasiones, sólo tenemos que recalcular los grupos afectados por una mutación o por la generación de un nuevo vecino (se verá más claro cuando veamos la búsqueda local).

Operadores genéticos y otros parámetros

A continuación, vamos a detallar la configuración de los operadores genéticos que se han empleado que, aún siendo básicos, han conseguido buenos resultados como veremos más adelante. Tras un estudio empírico, los operadores genéticos y demás parámetros quedarían de la siguiente forma:

- Población: cada población está formada por *20* individuos.
- *Inicialización*: para la creación de la primera población, todos los individuos se generan aleatoriamente. En posteriores intervalos de tiempo (el algoritmo se ejecuta de forma continua en el tiempo en el entorno de producción y no para un intervalo concreto como el que se presenta en el estudio), tomaremos el mejor individuo obtenido en el anterior intervalo (la variabilidad entre intervalos puede ser relativamente pequeña), y el resto de individuos se generan aleatoriamente y se refinan con una búsqueda local para no partir de soluciones muy pobres.
- *Selección*: con objeto de ver qué individuos pasan a la siguiente generación, hemos decidido aplicar t torneos binarios.
- *Cruce*: el siguiente paso consiste en producir una nueva generación de individuos. Específicamente, se considera que los hijos heredan los genes coincidentes en ambos padres y que el resto los reciben de forma aleatoria.
- *Mutación*: con el objetivo de causar pequeños cambios en el cromosoma y mantener de forma explícita la diversidad, y evitar una convergencia prematura, se provoca una perturbación sobre el 3% de los genes del cromosoma.

- *Política de reemplazo*: hay que decidir qué individuos se incorporan o reinsertan en la población tras una generación. Para ello, se emplea un esquema estacionario especial. Reemplazamos los peores individuos de la población por los dos mejores (elitismo) con una probabilidad de 0.7, mientras que con una probabilidad de 0.3 cogemos el mejor y el cromosoma más diferente a éste.

- *Condición de parada*: el algoritmo se ejecuta durante n segundos (tantos como dure la ventana, en nuestro caso n=300).

Búsqueda local

La búsqueda local es una meta-heurística que se usa para resolver problemas de optimización. La búsqueda local comienza por una solución candidata y, de forma iterativa, se va moviendo por el espacio de soluciones hacia otras soluciones vecinas, generando su vecindario. Para llevar a cabo esta tarea, se tiene que definir una relación de vecindad. En nuestro caso, diremos que dos soluciones son vecinas cuando sólo hay un gen de diferencia entre ellas.

El siguiente pseudocódigo ilustra el algoritmo de búsqueda local que se aplica cada *10* generaciones sobre los *5* mejores individuos (recuerde que la función de evaluación se refiere a la parcial):

```
void Búsqueda_Local (Cromosoma & solución_candidata)
begin
    Cromosoma mejor_solución = solución_candidata;
    Cromosoma vecino = solución_candidata;
    for $i \leftarrow 0$ to $solución\_candidata.tam() - 1$ do
        Agente a = vecino.obtieneAgente(i);
        for $j \leftarrow 0$ to $a.obtiene\_número\_perfiles() - 1$ do
            vecino.cambia_perfil(i,j);//perfil $j$ para el agente $i$
            if ($vecino.fitness() > mejor\_solución.fitness()$) then
                mejor_solución = vecino;
            end
            vecino = mejor_solución;
            j++;
        end
        solución_candidata = mejor_solución;
        i++;
    end
end
```

Algorithm 7: Pseudo-código de la búsqueda local propuesta.

Paralelizando el algoritmo memético

Ya sólo nos queda paralelizar el algoritmo y lo vamos a conseguir gracias a un modelo de islas de la siguiente forma:

- *Topologías*: usaremos una topología de islas en esquema de anillo, de hub (todos con todos) o en estrella con una población maestra y cuatro subordinadas que cada cierto tiempo intercambian algunas de sus soluciones con la maestra y ésta a su vez les envía otras soluciones.

- *Migración*: cada isla enviará un porcentaje variable de sus mejores soluciones a la isla correspondiente.
- *Reemplazo*: aplicaremos la misma filosofía del memético simple.
- *Frecuencia de migraciones*: cada *50* generaciones, la isla maestra bloquea asíncronamente al resto de islas subordinadas para pedirles, a cada una, un porcentaje variable de sus mejores soluciones.

También se ha desarrollado un *algoritmo meta-genético* que tiene en cada gen de su cromosoma los parámetros arriba indicados, con una amplia variedad de configuraciones.

Evaluación de resultados

En esta sección, vamos a analizar los resultados obtenidos al aplicar la configuración final propuesta en este trabajo, combinando el módulo de predicción con el módulo de optimización y haciendo uso de la ventana temporal auto-adaptativa. Pero antes de analizar el funcionamiento del sistema en su globalidad, vamos a estudiar cada módulo por separado.

De esta manera, vamos a ver los resultados del módulo predictivo para *5* GLLs y *91* días de datos históricos para validar la bondad del modelo. Para realizar la comparativa, utilizaremos los paquetes RNNS y RForecast de R y compararemos con *15* técnicas de reconocido prestigio dentro del mundo de los modelos predictivos. Estas técnicas se clasifican en series temporales, modelos de arima, redes neuronales y modelos de regresión lineales y no lineales. Las Tablas 7.2 - 7.6 muestran los resultados medios para *50* ejecuciones.

Asimismo, vamos a ver los resultados que se obtienen aplicando otras técnicas de renombre en el ámbito de la meta-heurística, como son el enfriamiento simulado, la búsqueda local iterativa y la búsqueda en entorno variable. Para ello, hemos aplicado estas técnicas sobre dos conjuntos diferentes de datos (dos ventanas de *300* segundos de amplitud comprendidas entre las 12:40 y las 12:45): uno de dificultad media (un día normal) y otro tomando un día gran volumen de llamadas y alta variabilidad (ver Figura 7.5).

Todas las ejecuciones han sido realizadas en un servidor Sun Fire E4900 (*2* procesadores para el modulo predictivo, otro para las interfaces y preprocesado, otros *5* para los algoritmos y los *4* restantes se han dejado para otras aplicaciones) con 96GB RAM.

Es importante destacar que todos los datos provienen de fuentes reales del entorno de producción del CLL de Telefónica.

Método	Mín. Error	Máx. Error	MAE	Desv.	Correlación
uRprop	*-76*	*123*	*4.533*	*8.288*	*0.937*
Suavizado Exponencial	-150	127	4.777	8.470	0.934
Series Temporales Simples	-160	129	4.767	8.508	0.934
Series Temporales Estacionarias	-161	129	4.767	8.508	0.934
Series Temporales Tendencias	-161	128	4.768	8.509	0.934
Arima (0.1.3)(0.0.0)	-154	129	4.779	8.489	0.934
Arima (0.1.5)(1.0.1)	-153	128	4.781	8.477	0.934
Backprop	-123	128	4.606	8.221	0.938
BackpropMomentum	-90	130	4.58	8.219	0.938
Rprop	-105	127	4.619	8.165	0.939
Quickprop	-99	128	4.639	8.254	0.938
BackpropChunk	-90	132	4.579	8.213	0.938
RBF	-68	159	4.832	10.698	0.892
BackpropWeightDecay	-95	127	4.574	8.314	0.938
Regresión Logística	-118	120	4.905	8.471	0.920
Regresión Lineal	-127	119	4.956	8.499	0.918

Table 7.2: Resultados para el GLL-1: Método, error mínimo, error máximo, error absoluto medio, desviación típica y correlación de Pearson.

Método	Mín. Error	Máx. Error	MAE	Desv.	Correlación
uRprop	*-32*	*36*	*3.244*	*5.360*	*0.968*
Suavizado Exponencial	-40	39	3.356	5.507	0.967
Series Temporales Simples	-36	39	3.405	5.567	0.966
Series Temporales Estacionarias	-36	39	3.406	5.509	0.966
Series Temporales Tendencias	-40	39	3.356	5.507	0.967
Arima (0.1.3)(0.0.0)	-36	39	3.402	5.564	0.966
Arima (0.1.5)(1.0.1)	-37	39	3.397	5.554	0.966
Backprop	-35	39	3.259	5.432	0.968
BackpropMomentum	-37	37	3.257	5.344	0.968
Rprop	-32	37	3.273	5.339	0.968
Quickprop	-33	36	3.319	5.39	0.968
BackpropChunk	-37	37	3.265	5.333	0.968
RBF	-40	136	3.951	6.403	0.947
BackpropWeightDecay	-34	37	3.255	5.385	0.968
Regresión Logística	-32	39	3.853	5.958	0.961
Regresión Lineal	-32	40	3.899	6.106	0.952

Table 7.3: Resultados para el GLL-2: Método, error mínimo, error máximo, error absoluto medio, desviación típica y correlación de Pearson.

Método	Mín. Error	Máx. Error	MAE	Desv.	Correlación
uRprop	*-19*	*29*	*2.368*	*3.960*	*0.948*
Suavizado Exponencial	-24	29	2.417	4.036	0.946
Series Temporales Simples	-24	30	2.445	4.062	0.946
Series Temporales Estacionarias	-24	30	2.445	4.063	0.945
Series Temporales Tendencias	-24	29	2.417	4.036	0.946
Arima (0.1.3)(0.0.0)	-24	30	2.444	4.060	0.946
Arima (0.1.5)(1.0.1)	-25	29	2.44	4.052	0.946
Backprop	-20	29	2.436	4.002	0.947
BackpropMomentum	-20	29	2.394	3.963	0.948
Rprop	-21	28	2.403	3.948	0.948
Quickprop	-19	28	2.457	4.038	0.946
BackpropChunk	-22	28	2.419	2.985	0.947
RBF	-24	45	3.17	6.538	0.851
BackpropWeightDecay	-20	29	2.374	3.958	0.948
Regresión Logística	-33	28	2.880	4.580	0.942
Regresión Lineal	-32	28	2.962	4.624	0.943

Table 7.4: Resultados para el GLL-3: Método, error mínimo, error máximo, error absoluto medio, desviación típica y correlación de Pearson.

Método	Mín. Error	Máx. Error	MAE	Desv.	Correlación
uRprop	*-20*	*20*	*1.728*	*3.129*	*0.948*
Suavizado Exponencial	-22	21	1.732	3.183	0.947
Series Temporales Simples	-22	21	1.75	3.208	0.946
Series Temporales Estacionarias	-22	22	1.749	3.203	0.946
Series Temporales Tendencias	-22	21	1.732	3.183	0.947
Arima (0.1.3)(0.0.0)	-22	21	1.748	3.203	0.946
Arima (0.1.5)(1.0.1)	-22	21	1.744	3.198	0.946
Backprop	-20	20	1.836	3.178	0.947
BackpropMomentum	-20	19	1.766	3.122	0.948
Rprop	-21	19	1.761	3.116	0.949
Quickprop	-19	21	1.883	3.217	0.945
BackpropChunk	-21	20	1.856	3.195	0.946
RBF	-21	34	2.256	5.111	0.855
BackpropWeightDecay	-21	20	1.778	3.132	0.948
Regresión Logística	-21	29	1.993	3.239	0.930
Regresión Lineal	-20	30	1.949	2.631	0.928

Table 7.5: Resultados para el GLL-4: Método, error mínimo, error máximo, error absoluto medio, desviación típica y correlación de Pearson.

Método	Mín. Error	Máx. Error	MAE	Desv.	Correlación
uRprop	*-10*	*13*	*0.976*	*1.353*	*0.777*
Suavizado Exponencial	-12	18	0.981	1.804	0.602
Series Temporales Simples	-12	18	0.981	1.804	0.602
Series Temporales Estacionarias	-12	18	0.981	1.803	0.603
Series Temporales Tendencias	-12	18	0.981	1.804	0.603
Arima (0.1.3)(0.0.0)	-12	18	0.981	1.804	0.603
Arima (0.1.5)(1.0.1)	-12	18	0.981	1.803	0.603
Backprop	-5	17	0.948	1.694	0.617
BackpropMomentum	-5	17	0.986	1.741	0.588
Rprop	-5	18	0.778	1.762	0.574
Quickprop	-4	18	1.079	1.802	0.548
BackpropChunk	-5	17	0.967	1.726	0.598
RBF	-5	18	1.002	1.773	0.567
BackpropWeightDecay	-5	16	0.966	1.688	0.620
Regresión Logística	-8	13	1.084	1.798	0.748
Regresión Lineal	-8	13	1.1	1.823	0.745

Table 7.6: Resultados para el GLL-5: Método, error mínimo, error máximo, error absoluto medio, desviación típica y correlación de Pearson.

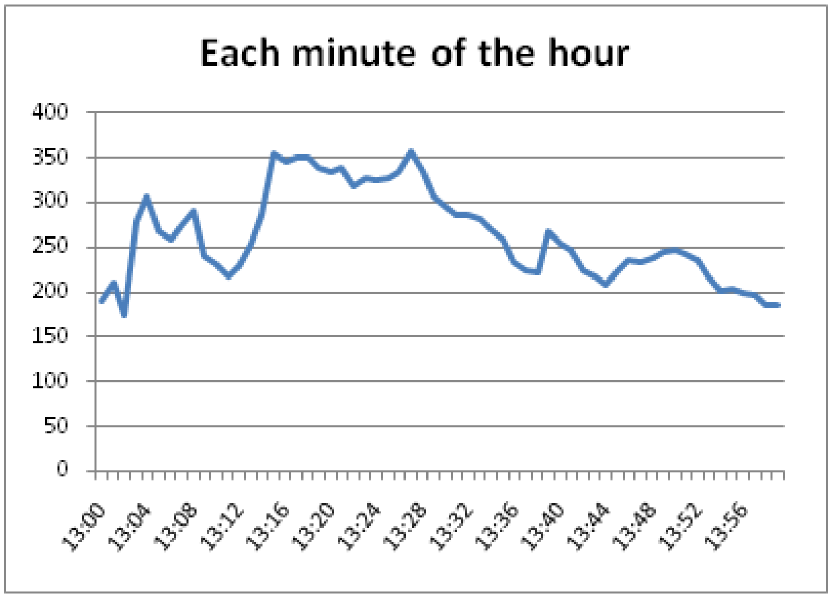

Figure 7.5: Número de llamadas entrantes durante la hora más compleja del día 9 de septiembre de 2009.

Algoritmo	*Mejor solución*	*Peor solución*	*Media*	*Desv. típica*
	Medio Difícil	*Medio Difícil*	*Medio Difícil*	*Medio Difícil*
Algoritmo memético paralelo	0.834 0.818	0.823 0.783	0.829 0.809	0.003 0.002
Algoritmo memético simple	0.796 0.758	0.785 0.751	0.796 0.754	0.001 0.001
Búsqueda local iterativa	0.768 0.728	0.755 0.722	0.763 0.725	0.002 0.003
Búsqueda entorno variable	0.790 0.727	0.766 0.723	0.775 0.724	0.005 0.001
Enfriamiento simulado	0.782 0.721	0.773 0.709	0.779 0.716	0.001 0.003

Table 7.7: Resultados obtenidos por cada meta-heurística en 50 ejecuciones para dos instancias de media y alta dificultad.

La Tabla 7.7 ilustra los resultados obtenidos, tras *50* ejecuciones, para las dos instancias del problema consideradas. Dicha tabla muestra la mejor y peor solución obtenidas por cada una de las técnicas estudiadas, así como la media y desviación típica de las *50* ejecuciones.

En la Tabla 7.7, se puede apreciar cómo el algoritmo memético paralelo obtiene mejores soluciones tanto en sus valores máximo y mínimo, como en media, comparándolo con el resto de técnicas. No sólo se trata de una técnica que nos permite alcanzar mejores soluciones en el espacio de búsqueda, sino que también nos permite obtener soluciones más estables, convirtiéndose en la técnica más robusta de las estudiadas.

El propio algoritmo memético simple, ya mejora al resto de técnicas en ambas instancias del problema, sobre todo para el caso más complejo.

Otro hecho a destacar radica en que a medida que se complica el problema, el paralelismo es mucho más importante ya que, aunque normalmente tenemos muchos agentes para atender llamadas, cuando los recursos son más escasos (en ciertas franjas horarias) es vital tener un algoritmo que permita una mejor gestión de distribución de la carga de trabajo. Precisamente, en esos puntos (p.e. alrededor de las 12:40) es donde hay que hacer un mayor hincapié y donde la robustez se hace más necesaria.

Con una mejor distribución de carga de trabajo, no sólo tenemos clientes más satisfechos, sino que la optimización conlleva a que los trabajadores también se sentirán mejor tratados (no sobreexplotados) y podrán realizar su trabajo de mejor agrado.

Algoritmo	*Nivel de servicio real*	*Desviación estándar*	*Eficacia*
Nuestro enfoque paralelo	0.987	0.016	100
Nuestro enfoque simple	0.941	0.020	95.339
ED-SBR	0.901	0.043	91.286
SBR	0.860	0.056	87.132
KOOLE	0.733	0.029	74.265

Table 7.8: Comparativa a lo largo del tiempo de nuestro enfoque simple y paralelo con otros algoritmos relevantes del mundo del call center.

Por último, vamos a ver el funcionamiento global del sistema combinando los módulos predictivo y módulo de optimización. La Tabla 7.8 resume los resultados obtenidos al ejecutar durante un día nuestro enfoque global paralelo, en comparación con otras técnicas convencionales del mundo del call center (ED-SBR y SBR clásico, algoritmo de Koole).

Vemos que nuestro enfoque es mucho más adecuado para entornos dinámicos como el call center que las técnicas convencionales, dado que éstas no realizan una planificación a largo plazo, sino que consideran el estado actual usando heurísticas más o menos complejas.

Para facilitar la visualización de los resultados de la tabla, la Figura 7.6 muestra el detalle de los resultados obtenidos por cada técnica a lo largo del día. Dichos resultados representan el nivel de servicio real obtenido en cada punto del día.

Es muy importante obtener resultados cuasi-óptimos para ventanas de trabajo aisladas, pero no podemos obviar que nuestro enfoque global se está ejecutando de forma continua, por lo que las transiciones entre los estados del sistema deben tenerse en cuenta. Nuestro enfoque demuestra que no sólo es el más adecuado para una determinada ventana temporal fija, sino que es el más adecuado a lo largo del día.

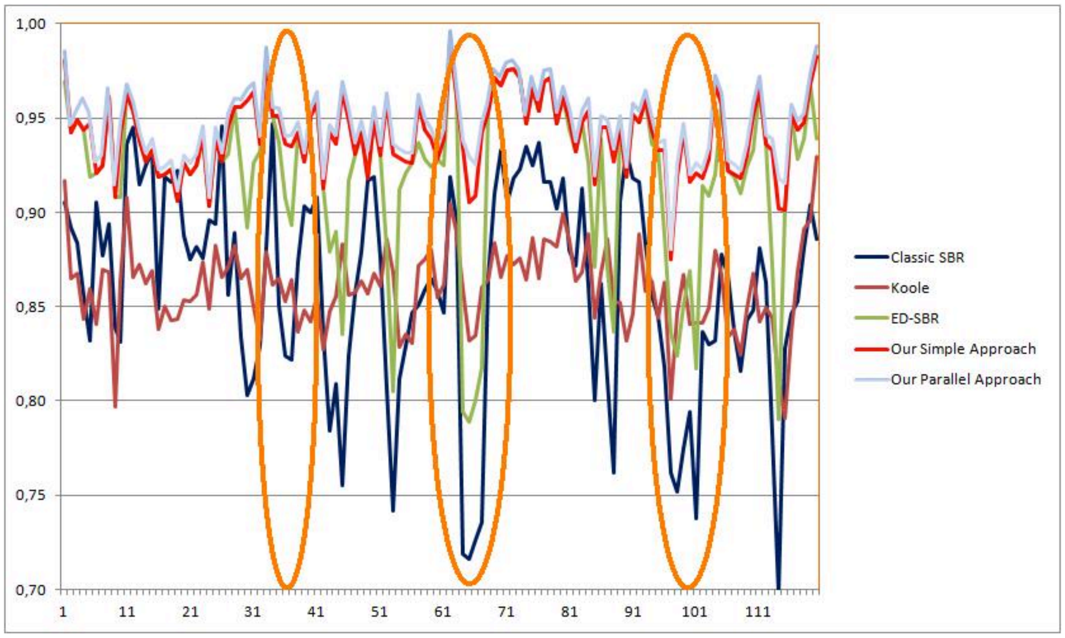

Figure 7.6: Nivel de servicio ofrecido por las diferentes técnicas de asignación de esfuerzo en el call center para un día de campaña. Eje X representa el tiempo y el eje Y representa el nivel de servicio real.

Conclusiones, Contribuciones y Trabajo Futuro

Conclusiones

En este trabajo hemos presentado un nuevo enfoque para resolver el problema de distribución de esfuerzo en centros de llamadas multi-agente y multi-tarea altamente dinámicos, mediante la combinación de algoritmos predictivos y algoritmos meméticos paralelos.

Hemos visto que este tipo de centros de llamadas requiere de técnicas adaptativas, pues se trata de un entorno extraordinariamente cambiante. El uso de algoritmos meméticos, los cuales son mucho más complejos que las técnicas ad-hoc, nos ayuda a obtener soluciones más precisas y robustas, siempre y cuando dichas soluciones vayan acompañadas de una adecuada predicción de las tareas a realizar.

Se ha demostrado que con un sistema de ventana de trabajo temporal auto-adaptativa, podemos superar las tradicionales técnicas de planificación a corto y largo plazo del mundo del call center.

Hemos propuesto una red neuronal basada en un novedoso algoritmo de aprendizaje para realizar las predicciones oportunas y un algoritmo memético paralelo como mecanismo de optimización heurística que permitía mejorar los resultados obtenidos por otras técnicas reconocidas dentro del mundo de los centros de llamadas. Además, dicho algoritmo constaba de un meta-agoritmo genético que le permitía calibrar automáticamente todos los parámetros de forma correcta.

En este estudio, se han presentado diferentes pruebas exhaustivas sobre diversos conjuntos de datos de diferente dificultad que demuestran que nuestro enfoque no sólo es el más potente de forma unificada, sino que sus propios módulos (predicción y optimización) son de por sí más potentes que otras alternativas del estado del arte.

Por último, queremos destacar que nuestro enfoque ha sido probado en un entorno real de producción (Telefónica).

Contribuciones

Las contribuciones de este trabajo pueden concebirse desde diversas perspectivas, aunque la principal aportación es la presentación de un nuevo enfoque para el problema de distribución de esfuerzo, que fusiona predicciones realistas con una adecuada optimización, considerando un mecanismo de ventana de trabajo adaptativa. También aplicamos este enfoque a un entorno de producción real (Telefónica), superando a otras técnicas existentes de centros de llamadas.

Nuestro enfoque también nos permite trabajar con un nivel de granularidad menor (grano fino) que los algoritmos de planificación a corto plazo (grano grueso), debido a que nuestro algoritmo de búsqueda tiene más tiempo para encontrar una solución que las técnicas voraces convencionales, gracias a las predicciones a medio plazo que realiza el módulo predictivo.

A continuación, presentamos otras contribuciones técnicas de este trabajo:

- Este trabajo propone un enfoque escalable y paralelizable para un problema NP-duro del mundo real.
- Se describe un método de ventana de trabajo auto-adaptativa, muy robusto que determina la dinámica del sistema en cada momento y un modelo de Markov para llevar a cabo las transiciones del sistema.
- Se presenta una versión mejorada del algoritmo de aprendizaje de retro-propagación para redes neuronales.
- Se proponen políticas de migración y reemplazo novedosas para algoritmos genéticos paralelos, inspiradas en otras meta-heurísticas.
- Se propone un novedoso algoritmo meta-genético para calibrar el memético paralelo que compone el módulo de optimización.
- También proponemos un método para evaluar las funciones fitness de forma parcial y así reducir los tiempos de evaluación de cada solución.
- Se aportan diversas comparativas exhaustivas de los diferentes componentes y del enfoque global.

Por último, la contribución científica de esta tesis se refleja en las *11* publicaciones que han sido fruto del presente trabajo de investigación:

1. *Publicaciones en revistas internacionales*:

   (a) Millán-Ruiz, D. and Hidalgo, J.I.: *Forecasting Call-Centre Arrivals.* Journal of Forecasting, Wiley, Ed. Derek W. Bunn, 2013.

(b) Arnaldo, I., Contreras, I.; Millán-Ruiz, D., Hidalgo, J.I. and Krasnogor, N.: *Matching Island Topologies to Problem Structure in Parallel Evolutionary Algorithms.* Soft Computing. A Fusion of Foundations, Methodologies and Applications. Special Issue on Bio-Inspired Algorithms with Structured Populations, Springer, 2013.

2. *Publicaciones en conferencias internacionales de relevancia*:

   (a) Millán Ruiz, D. and Hidalgo, J.I.: *Migration and Replacement Policies for Preserving Diversity in Dynamic Environments.* Proceedings of the 12th European Conference on the Applications of Evolutionary Computation (EvoApplications 2012, Málaga, Spain, April 11-13, 2012.

   (b) Millán Ruiz, D., Hidalgo, J.I.: *Comparison of Metaheuristics for Workforce Distribution in Multi-Skill Call Centres.* Proceedings of the International Joint Conference on Computational Intelligence (ICEC 2010), Valencia, Spain, October 24-26, 2010.

   (c) Millán Ruiz, D., Hidalgo, J.I.: *A Parallel Memetic Algorithm for Dynamic Workload Distribution in Multi-Agents Systems.* Proceedings of the 3rd Workshop on Parallel Architectures and Bioinspired Algorithms held In conjunction with PACT 2010, Vienna, Austria, September 11-15, 2010.

   (d) Millán Ruiz, D., Pacheco, J., Hidalgo, J.I., Vélez, JL: *Forecasting in a multi-skill Call Centre.* Proceedings of the 10th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2010), Zakopane, Poland, June 13-17, 2010.

   (e) Millán Ruiz, D., Hidalgo, J.I.: *A Memetic Algorithm for Workforce Distribution in Dynamic Multi-Skill Call Centres.* Proceedings of the 10th

European Conference on Evolutionary Computation in Combinatorial Optimisation (EVOCOP 2010), p. 178-189, Istanbul, Turkey, April 7-9, 2010.

(f) Meléndez, J., Lopez, B., Millán-Ruiz, D.: *Probabilistic models to assist maintenance of multiple instruments.* Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2009), p. 1499-1503, Palma de Mallorca, Spain, September 22-26, 2009.

(g) Pacheco, J., Millán-Ruiz, D., Vélez, JL: *Neural Networks for Forecasting in a multi-skill Call Centre.* Proceedings of the 11th International Conference on Engineering Applications of Neural Networks (EANN 2009), p. 291-300, London, UK, August 27-29, 2009.

(h) Martinez-Lopez, R., Millán-Ruiz, D., Martin-Dominguez, A.; Toro-Escudero, MA: *An Architecture for Next-Generation of Telecare Systems Using Ontologies, Rules Engines and Data Mining.* Proceedings of the International Conferences on Computational Intelligence for Modelling, Control and Automation, Intelligent Agents, Web Technologies and Internet Commerce, and Innovation in Software Engineering (CIMCA 2008), p. 31-36, Vienna, Austria, December 10-12, 2008.

3. *Publicaciones en conferencias nacionales de relevancia*:

   (a) Millán Ruiz, D., Hidalgo, J.I.: *Algoritmo memético paralelo para la distribución de esfuerzo en centros de llamadas dinámicos multi-agente y multi-tarea.* Proceedings of the 7th Spanish Conference on Meta-heuristics, Evolutionary Algorithms and Bioinspired Algorithms (MAEB 2010), Valencia, Spain, September, September 8-10, 2010.

Áreas de Investigación Futura

Por último, se proponen algunas directrices para el trabajo futuro. En primer lugar, se recomienda el análisis de más juegos de datos, especialmente si queremos generalizar ciertas conclusiones a otros ámbitos y otros entornos multi-agente.

Otra posible mejora implica la selección de más atributos para realizar las predicciones, así como técnicas más sofisticadas para la selección de dichos atributos. En este sentido, se deberían de analizar otras técnicas no lineales, ya que análisis de componentes principales es una técnica que realiza componentes lineales de un conjunto de variables de entrada.

También se podría hacer un estudio más profundo del tratamiento de restricciones que permita añadir nuevas reglas y restricciones sin impactar mucho en la función de evaluación.

Para los niveles de dinamismo, también podríamos tener una aproximación continua (sin niveles) para aquellos sistemas multi-agente dinámicos donde los agentes no son seres humanos de forma que no hay que preocuparse por sus derechos o restricciones físicas. Por ejemplo, podríamos cambiar sus perfiles de habilidad en cualquier momento sin ningún tipo de restricción legal.

Para concluir, proponemos ampliar este estudio mediante técnicas multi-objetivo como SPEA-II o NSGA-II, dada nuestra formulación del problema.