Enterprise Information Systems
Vol. 00, No. 00, February 2012, 1–22

RESEARCH ARTICLE

Acquisition of Business Intelligence from Human Experience in Route Planning

Gema Bello Orgaz[a*], David F. Barrero[b], María D. R-Moreno[b] and David Camacho [a]

[a]*Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Spain*; [b]*Departamento de Automática, Universidad de Alcalá, Spain*

(*v2.2 released September 2008*)

The logistic sector raises a number of highly challenging problems. Probably one of the most important ones is the shipping planning, i.e., plan the routes that the shippers have to follow to deliver the goods. In this paper we present an AI-based solution that has been designed to help a logistic company to improve its routes planning process. In order to achieve this goal, the solution uses the knowledge acquired by the company drivers to propose optimized routes. Hence, the proposed solution gathers the experience of the drivers, processes it and optimizes the delivery process. The solution uses Data Mining to extract knowledge from the company information systems and prepares it for analysis with a Case-Based Reasoning (CBR) algorithm. The CBR obtains critical business intelligence knowledge from the drivers experience that is needed by the planner. The design of the routes is done by a Genetic Algorithm (GA) that, given the processed information, optimizes the routes following several objectives, such as minimize the distance or time. Experimentation shows that the proposed approach is able to find routes that improve, in average, the routes made by the human experts.

Keywords: Logistics; Business Intelligence; Route optimization; Case-Based Reasoning; Genetic Algorithms; Applied AI; Information Systems

*Corresponding author. Email: david@aut.uah.es

ISSN: 1751-7575 print/ISSN 1751-7583 online

DOI: 10.1080/1751757YYxxxxxxxx
http://www.informaworld.com

1. Introduction

The logistic professionals face challenges of increasing complexity. They must maximize the customer satisfiability, delivering things on time while minimizing the impact on the environment, fuel consumption and, at the same time, maximizing the company profit. This is usually a tough task given the constant increase in travel demands, fueled by economic development, and the ever-growing demands to do more with less. Some examples of those challenges that they face include high employee mobility, poor safety and unreliability. Adding to the challenge is the fact that there are inherently complex systems involving a very large number of components, each having different and often conflicting objectives.

The actual demand of precise and trustable information in the logistic sector has driven the development of complex Geographic Information Systems (GIS), very useful for planning their routes (Sadeghi-Niaraki *et al.* 2011). Those systems are combined with Global Position Systems (GPS) for positioning the different elements involved in the shipping. However, although very useful, those systems cannot take decisions based on, for example, shortcuts in the route that human operators have learned from their experience. At this point, Artificial Intelligence (AI) provides some tools that can boost the utility of decision-support tools (Beheshti and Beheshti 2010, Li 2012). Data Mining (Witten and Frank 2005) is also useful to extract knowledge from data sets in a human-understandable structure. This inferred information can be used later to improve the system efficiency.

To be competitive, optimizing the distribution chain to fulfill the customer requests is one of the most important issues in the logistic sector. In this complex problem, researchers and practitioners from AI have seen a big potential to address some of the mentioned problems, and in this way to improve the efficiency, safety, and environmental-compatibility of the logistic companies (Hanif *et al.* 2011). Some of the areas that best fit in this context are Data Mining, Machine Learning, i.e, Case Based Reasoning (CBR) (Aamodt and Plaza 1994) and Genetic Algorithms (GAs) (Holland 1992, Martínez-Bernabeu *et al.* 2012). The optimization of distribution routing can be considered as a Traveling Salesman Problem (TSP) (LaRusic *et al.* 2012). Depending on the characteristics of the distribution, the problem might be abstracted to some other variations of the basic TSP. One of these potential variations is addressed in (Minis *et al.* 2012), where a TSP is solved taking into account that a vehicle can breakdown.

This paper proposes a solution to the distribution optimization problem. Since the logistic company has different delivery centers, each one with its own information systems storing information about the drivers' experience, a satisfactory solution to this problem requires the use of semantic structures to integrate data from the information systems and Data Mining to identify the drivers' behavior. Once the information has been preprocessed, a CBR is applied to obtain the business intelligence from the drivers experience which is used to feed a GA that optimizes the planning routes.

The main contribution of our approach is the combination of Data Mining, CBR and GA. This hybridization of techniques enables our solution to use knowledge acquired by the company drivers. It is implicitly stored in the information systems in form of routes. This knowledge is made explicit with the CBR and then used to optimize the shippings routes through a GA. To the best of our knowledge, this is the first attempt to introduce implicit expert knowledge in the route optimization problem using a CBR and a GA.

The rest of the paper is structured as follows. It first describes some AI solutions to the route optimization problem. Then, section 3 presents the different components that the proposed solution is subdivided into, with special emphasis in the data processing

and transformation for interoperability. Section 4 describes in detail the CBR architecture while the optimization algorithm based on a GA is introduced in section 5. The description of the solution is complemented by section 6, that contains the experimental evaluation of the solution, including real data provided by a logistic company and a comparison between the routes generated by human-experts and the proposed solution. The paper finishes outlining some conclusions and future research lines that remain open.

2. Related Work

AI has been widely used in the logistic domain, therefore the existence of a large corpus of literature on this topic is not surprising. Over the last years, logistic companies have integrated information systems to manage their activities and to improve the process efficiency (March and Storey 2008, Jung 2011). For example, an adaptive heuristic approach to optimize the order picking efficiency is proposed in (Chiang *et al.* 2011), where a higher service level with shorter distances is achieved. Another example comes from Distributed Artificial Intelligence (DAI), that offers suitable tools to deal with the problems in the transportation domain (Fischer *et al.* 1995).

In logistics, each driver in the company needs a shipping route. In an ideal scenario, routes must maximize user satisfaction while minimize the cost under different perspectives such as time or distance. Several AI-based solutions have been proposed to solve this problem (Pisinger and Ropke 2007, Savelsbergh and Sol 1995), including, for instance, agent technology (Wooldridge and Jennings 1995). Distributed agent-based decision systems have provided satisfactory solutions given the nature of the problem, which is dynamic, unpredictable and distributed. An example of this category of solution is the Living Systems Adaptive Transportation Networks (LS/ATN) (Neagu *et al.* 2006).

Another perspective to address the problem comes from Graph Theory. It can be used to analyze different aspects of the network, such as its structure, behaviour, stability or even community evolution inside the graph (Emmert-streib 2011). The shipping routes can be abstracted as a graph, and therefore Graph Theory can be applied to solve this problem. This approximation has been successfully applied, for instance, a shortest path algorithm of Graph Theory has been used to solve the vehicle routing problem (Liu Xiao-Yan 2010, Fan and Meng 2010).

Other approaches are based on random search algorithms based on an objective function to be optimized. Methods such as Genetic Algorithms (Holland 1992) have been traditionally used in optimization problems with notable success (Changjiang *et al.* 2009, Bin *et al.* 2010, Jiang *et al.* 2009). There is a wide range of applications where GAs have been successful, from optimization (Zhao and Wang 2011), to Machine Learning (Agustín-Blas *et al.* 2012) or even evolutionary art (Romero and Machado 2007). GAs have demonstrated to be robust algorithms able to find satisfactory solutions in highly multidimensional problems with complex relationship among the variables. In this type of problems GAs are able to find a maximum in an extensive search space evaluating a minimum number of points. The Traveling Salesman Problem (TSP) belongs clearly to this type of problems and has attracted much attention in the GA community (Sengoku and Yoshihara 1998, Fogel 1988, Starkweather *et al.* 1991, Bhattacharyya and Bandyopadhyay 2008) and related fields (Buriol *et al.* 2004, Ilie and Bdic 2011). Therefore, using a GA to optimize routes seems a reasonable choice (Ko and Evans 2007).

Another discipline that is used in our approach is Machine Learning (Bishop 2007). Using it, drivers experience can be extracted, new knowledge inferred, and then improve

the business intelligence (Duan and Xu 2012). In particular, we propose CBR (Aamodt and Plaza 1994, He *et al.* 2009a,b), an AI technique that solves new problems based on the acquired knowledge about how similar problems were solved in the past. Any CBR system can be defined using two main elements: the *cases*, or concepts which are used to represent the problem (their main characteristics and the solution that was found to solve this problem), and the knowledge base that is used to store and retrieve the cases. These techniques have been already applied in the logistic sector. We can mention the work of (Yan *et al.* 2003) to evaluate a logistic application, or using a CBR method to construct a risk assessment model to identify and manage risks in future projects (Kumar and Viswanadham 2007).

The solution that we propose is based on these techniques, a CBR to gather expert knowledge, and a GA that is fed by the CBR output to optimize the company delivery routes. However, in order to implement those algorithms a number of issues raises, and they have to be solved. For this reason the architecture that implements the solution has to increase its complexity. Next section deals with this architecture and the implementation problems that have to be solved.

3. Routes Planner System Architecture

Before introducing the system architecture, we should describe the scenario for which it was designed. Our starting point is a logistic company that has a set of distributors (around $1,200$ in the whole Spanish geography) with PDAs connected by GPRS. Each day, the order in which the delivery should be carried out is decided by the (human) distributor depending on a bunch of factors: Density of the traffic, state of the highway, hour of the day and place of the shipping. For example, it is better to ship in industrial areas early in the morning if the traffic is less dense. The list of tasks for each distributor is supplied each day from a central server.

The daily average number of shippings by each distributor is around 50. That is, we have around $60,000$ physical acts (delivers to carry out) each day, all of them monitored and stored in a log in the head office thanks to GPRS. The PDA of each distributor has a GPS module that allows them to follow the routes. So, experienced distributors can take advantage of these tools for doing better their job, using their knowledge about the sector. But actually, a common characteristic in this sector is the low rate of permanency of the employees. Therefore, the knowledge acquired by former employees about the routes that they followed is lost, and cannot be used by the novice employees. As a consequence, this knowledge is not used to improve the shippings routes and the company loses competitiveness. It is worth mentioning that there are approximately 10% of the deliveries that for some reasons (for instance, unknown recipient) cannot be carried out. From the $60,000$ physical daily acts, about 30% of them are highly reliable since they involve well-known places, time of the delivery or receptors.

The company needed a decision support tool that, from the one hand was able to ease the identification of the best driver behaviors and, on the other hand, learn from them to optimize its delivery process. In order to satisfy these requirements, we have designed a tool that is able to learn from the human drivers and provide this knowledge to future employees. In addition, the system uses this knowledge to plan the routes, as well as to evaluate and compare them under different criteria.

Our tool provides a solution that integrates different AI techniques to extract information and transform it to provide added value to the company. The tool uses the

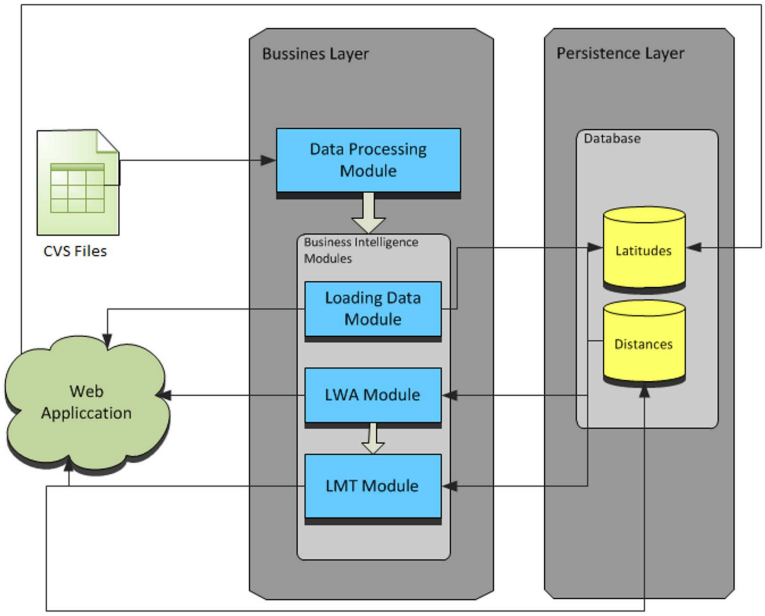

Figure 1. System architecture with its four main modules: Data Processing, Loading Data, Learning Working Areas (LWA) and Learning Manager Task (LMT).

information provided by the drivers about the deliveries, filters this information, and, based on it, defines the delivering zones or working areas for each driver. With all this information, it finally plans the routes. Based on these features, the tool is subdivided into four subsystems, as Figure 1 shows. A brief summary of each subsystem follows.

- The *Data Processing* Module takes the data from the files provided by the logistic company in CSV format (Comma Separated Values). It checks whether the file is a valid one, then loads the information into data structures and finally performs a first statistic analysis of each driver contained in the file.
- The *Loading Data* Module is in charge of obtaining and loading geographic information, in form of latitude and longitude, of each address by using the Google Maps API. In addition, this module computes the distances among the addresses.
- The *Learning Working Areas* (LWA) Module creates zones (or working areas) for each driver using the data loaded and processed in the previous modules. Each working area may contain one or more zip codes. It also allows us to visualize the different working areas where we can appreciate some of their features, such as the location, size or geographic distribution. This subsystem implements a CBR.
- The *Learning Manager Task* (LMT) Module optimizes routes based on the output generated by the LWA Module and the original human-made routes. It generates a certain number of plans as output. It also provides tools to compare the optimized routes to the original plan. This subsystem implements a GA and it is able to optimize plans according to different criteria, depending on which parameter the company is more interested in optimizing. To be more specific, the parameters to optimize include distance, positive rate of LWA, time, etc. The planning algorithm can adapt its answer to the driver behavior and generate plans according to it, if that is the user's desire.

Table 1. Summary of the fields contained in the CSV data files. These files are provided by the logistic company to the route optimization system and store historical data about the drivers activities.

Attribute	***Description***
Driver_id	Represents the identification code of the driver. With this code the logistic company identifies each driver and it is generally a 6 digits number.
State	Describes the state of the shipment, which can be `delivery done`, `pick up done`, `incident`, `login` and `logout`.
ZipCode	The zip code of the shipment address.
Address	City, name and number of the street.
Delivery information	Date and time of the shipment.
Incident	Represents the identification code of an incident, in case there was any.
Confirmation	The type of confirmation received by the client. It can take the values `signature`, `stamp` or `nothing`.
Comments or issues	Any incidence or comment generated in the shipping process: Delays originated by the reception people, comments about the correct place, how to make the shipping, ...

Each one of these modules is described in more detail in the next subsections.

3.1. *Data Processing Module*

This module deals with the data provided by the drivers, being responsible for loading data into the system. To this end, it reads the data from a CSV file, filters its content and loads the valid entries into memory. These files contain information about the packages shipment. CSV files follow the philosophy of a table, that is, the first line represents the name of the columns separated by ";". Then, each line after represents a row of the table with a value for each field, separated again by ";". In our case, each row represents a physical action (deliver, login, pick up, etc) of a driver or drivers during several days. The information contained in these files is summarized in Table 1. *A priori*, loading data from a CSV file is a simple task, however, in this case there is a couple of non-trivial issues that have to be handled.

In our information system, the percentage of misspellings in the addresses (streets name or zip codes) is rather high, around 45%, being the bottleneck of our application and a difficulty that has to be addressed. For this reason, the Data Processing Module includes a preprocessing stage that aims to correct the addresses. After some basic preprocessing the percentage drops to 20%. But this is still unacceptable, so it is necessary to introduce more complex techniques in the system, in particular, the Data Processing Module uses Data Mining algorithms to detect and correct address inconsistencies.

Another issue that has to be solved is the existence of equivalent entries in the data. They have to be detected and eliminated. Automatic detection of similar (or duplicate) entities is a critical problem in a wide range of applications from Web learning to protein sequence identification (Fellegi and Sunter. 1969, Newcombe *et al.* 1959). The identification of similar entities, or Record Linkage (RL), is necessary to merge databases with complementary information. Then, there is a need to solve these problems algorithmically given the costs of using human expertise.

The problem of RL was defined as the process of matching equivalent records that differ syntactically (Newcombe *et al.* 1959). In (Fellegi and Sunter. 1969), the entity matching

issue is formulated as a classification problem, where the basic goal is to classify entity pairs as matching or non-matching, based on statistical unsupervised methods. Edit distances, such as Levenshtein (Levenshtein 1966) or Jaro-Winkler (Winkler 1999), have been primarily used to solve this problem in the past. A string edit distance is a metric that can be used to determine how close two strings are. This value is obtained in terms of the edit distance. A brief description of the methods used in our solution follows:

(1) The *Levenshtein distance* between two strings (Levenshtein 1966) is defined as the minimum number of edits needed to transform one string into another one, with the edit operations being insertion, deletion and substitution of a single character.
(2) The *Jaro-Winkler distance* (Jaro 1995) is a measure of similarity between two strings. It is a variant of the Jaro distance metric and mainly used in the area of RL. The higher the Jaro-Winkler distance for two strings is, the more similar the strings are. The score is normalized such that 0 is no similarity at all and 1 is an exact match.
(3) The *block distance*, also known as *Manhattan distance* (Craw 2010), represents distances between points in a city road grid. In the case of strings, it examines the absolute differences between characters of a pair of strings.
(4) The *Cosine distance* (Qian *et al.* 2004) is a measure of similarity between two vectors using the Cosine's angle between them. The result of the Cosine function is equal to 1 when the angle is 0, and it is less than 1 when the angle is of any other value. The value of the angles allows us to determine whether two vectors/strings are pointing in roughly the same direction which means they are the same string.
(5) The *Dice distance* (Anuar and Sultan 2010) groups two written letters (bigrams) of the 2 strings we want to compare. Once we have the set of bigrams for each string, then the Dice distance is the result of dividing the double of the number of character bigrams found in both strings by the number of bigrams in the first string plus the number of bigrams in the second string.
(6) The *Monge-Elkan distance* (Monge and Elkan 1997) uses variable costs depending on the substring gaps between the words, this distance can be considered as an hybrid distance because it can accommodate multiple static string distances using several parameters.

In order to merge entries in the data files, *upper* (0.8) and *lower* (0.6) thresholds have been applied. When the value returned by the algorithm is greater than the upper threshold value, the wrong street is automatically replaced by the string with the highest value found by the algorithm. For values between the upper and lower threshold, the different alternatives are returned (we can set that number, by the default is 3) and the user has to decide which one is the correct one. If the value returned by the algorithm is smaller than the lower threshold value, the row is marked and no solutions are given. The user can change the lower threshold value or manually correct the street.

In addition to the preprocessing of data, this subsystem also gathers some basic statistical information about the drivers. One log is given for each driver. It stores all the shipping information for a complete month and it can be analyzed by its individual days. Then, some statistics about the driver are generated by this module, including the number of working days, number of physical acts, average of physical acts performed, or average of the type of confirmation received by the client.

Once the data has been preprocessed by this module by filtering and storing it in a database, it is prepared to be processed by the next subsystem, the Loading Data

Module.

3.2. *Loading Data Module*

The Loading Data Module is in charge of obtaining and loading the geographic information, in form of latitude and longitude, of each address. This information is provided thanks to the connection to the Google Maps API. It is subdivided into two subsystems: the geoposition and the distance modules. The *geoposition module* is dedicated to obtain the coordinates of each delivery address. These coordinates will be used for the zones analysis and representation using the Google Web application. The *distance module*, on the contrary, generates the distance between all pair of addresses for a given date. This information will be used by the planner to generate the routes.

Throughout the process of computing the coordinates, the module groups the shippings that belong to the same address, date and driver into one to simplify the problem. This is because a driver can deliver several packages in the same address; from the perspective of the route planner those packages can be envisioned as an unity, so they are grouped into what we name *act*. So, we define an act as a set of packages and pick ups that are delivered/collected at the same address and time by the same distributor. As a consequence, the addresses in the database are loaded by the Loading Data Module as acts, and an extra field represents the number of deliveries and pick ups for that act.

Finally, this module is also in charge of subdividing and loading into the database the *Address* field structure of the previous module into 3 fields: type of street, name of the street and number of the street. With the data correctly preprocessed and loaded, the system is ready to use it in order to identify the working areas, task that is implemented by the next module.

3.3. *Learning Working Areas (LWA) Module*

This module generates the working zones that divide the space of shipments for the drivers. To generate that subdivision we have used the zip code as the baseline. So, a zone or a working area can be represented by a zip code or more than one. To take that decision, we have considered the following statements:

- A zone should be large enough to contain different zip codes, in case that the zip codes represent small areas.
- We consider the zip codes format used in Spain. It contains geographic and population parameters useful for the natural subdivision in zones.
- In case of error in the addresses contained in the files -which is rather common-, it is easier to identify and correct an error in the zip code.

This module also provides a zone analysis of each driver, allowing the comparison of working areas that belong to different drivers. This feature is useful to identify overlapping areas among the drivers. The information can be visualized on the graphical interface provided by the Google Maps application. The working areas are the result of applying the CBR algorithm, it will be presented in detail in the section 4.

3.4. *Learning Manager Task (LMT) Module*

The LMT module plans and optimizes routes for a given date and driver. In order to perform this task the module implements a GA. Through the interface we can set the main parameters required by the GA, such as the fitness function used in the algorithm or the number of different plans we want to generate. This subsystem also eases the comparison of the optimized routes with the original ones performed by the driver. Of course, the critical element in this module is the GA, which is described in detail in section 5. In order to get all the information needed by the GA for the optimization process, it is necessary to identify the working zones. The algorithm that performs this task is described in the next section.

4. Business Intelligence Modelling and Learning Approach

The LWR module uses CBR (Aamodt and Plaza 1994) to learn areas and therefore to acquire the business intelligence of the logistic distributors. In our case, the knowledge base is extracted from the drivers experience and later it is reused for optimization and logistic issues. This knowledge is implicitly contained in the files that are handled by the Data Processing Module. The objective of the CBR is to explicit this knowledge and put it at disposal of the routes optimization algorithm.

The information loaded in the database can be used in a straight way (i.e. address and time delivery can be used to estimate how much time is necessary to complete a particular ship), or it can be used to indirectly learn from the distributors experience. They know which areas have high traffic hour by hour, so the driver can take alternative paths to reduce time and gas consumption. The drivers also know the gas stations and the right moment to stop for gas to minimize the impact to the route to follow. During the working day, the driver may stop in places not related to the shipping to rest or eat. These data could be extracted from the path variations in specific hours, and although it can build solutions theoretically worse than optimal (from a distance perspective), it can be better in time but waste other resources such as gas.

However, handling this information also generates some problems, mainly caused by the deficiency and irregularity of the data. The information has a lot of noise: there are many mistakes in the zip codes and the street names that are hard to correct *and t*he shipping time is not reliable since sometimes the drivers annotate the hour after they have done some deliveries. These deficiencies can be classified in three categories:

- Information noise: There is a high number of mistakes made by the drivers when they introduce information in the system. These mistakes are hard to correct, if not impossible.
- Information gaps: Quite often many shippings do not have the address fully specified, so it is not possible to analyze the route followed by the driver.
- Time inconsistencies: The shipping time is not reliable since sometimes the drivers annotate the hour after they have done the deliveries or pick ups.

For those reasons, the CBR uses this information in a simplified way. It prevents the effects of that noise. The generation of the final CBR information, such as the working areas of the drivers, are carried out using statistical considerations, specifically the average of behaviors followed by all the analyzed drivers, to minimize the noise. The next section describes the CBR used to learn the working areas.

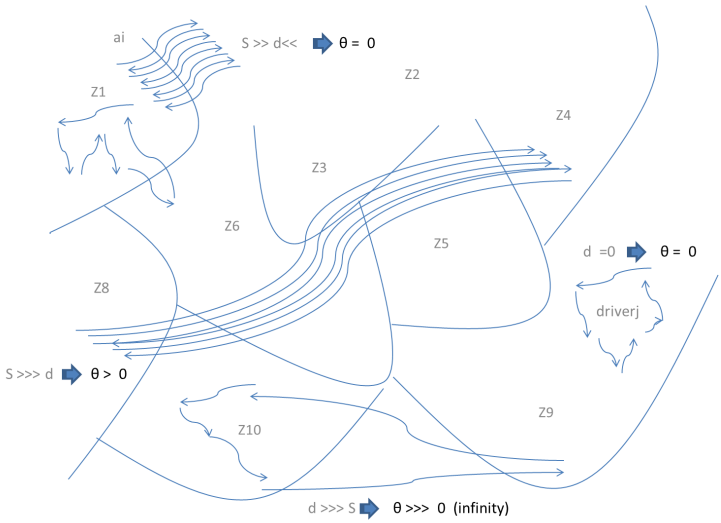

Figure 2. Representation of the grid of zip codes as well as drivers behavior. It shows drivers moving within a zip code and also drivers moving through several zip codes.

4.1. *CBR to automatically learn the working areas*

A critical issue for a logistic company is the concept of the *working zone*, or *working area* (WA). It represents a particular zone in a city, or in a country, where one or several drivers will work delivering or picking up objects, i.e., it is a zone that groups one or more acts. Usually, the logistic companies define these WAs using human experts who assign WAs to a set of drivers. Minimizing the overlapping of WAs among different drivers is essential to minimize the number of drivers and the deliver time for a set of acts. Therefore, this issue is critical for the company competitiveness. The automatic generation of WAs is the goal of the CBR.

The algorithm starts defining a grid that represents all the available postal codes $(Z_i, i \in [1..n])$ for a city or country. These postal codes are used to fix a geographical centroid so we can calculate, using any standard algorithm based on GPS coordinates, the distance between two postal codes Z_i and Z_j, $dist(Z_i, Z_j)$. On the other hand, the information from drivers' log is used to calculate how many acts belong to the same postal code and how many acts occur between adjacent postal codes. It is needed because a driver can work in a particular postal code, but he can also work in several postal codes. This information is represented using a parameter s_i, which is calculated as $s_i = \sum(acts(Z_i \rightarrow Z_j) + acts(Z_j \rightarrow Z_i))$, where $acts(Z_i \rightarrow Z_j)$ represents the number of jumps from postal code Z_i to postal code Z_j. In this way s_i represents the jumps between two postal codes. It seems reasonable to expect that if several acts are interleaved, and they are placed in different postal codes, these should be close enough to maintain the shipping cost low. Figure 2 shows a representation of the grid of zip codes and the drivers behavior.

Let us know consider a particular driver's log as a list of m ordered acts $(act_k, k \in [1..m])$. A new value $\theta = \sum(d_k/s_k)$, is calculated for each log:

$$d_k = dist(Z_k, Z_{k+1}). \quad (1)$$

Where Z_k represents the postal code for act_k and Z_{k+1} the postal code for act_{k+1}. This new parameter θ has the following characteristics:

(1) if $d = 0$ then $\theta = 0$; so these acts must be grouped in the same zone. Therefore, the acts that belong to the same zip codes should be grouped in the same zone.
(2) if $s = 0$ or $s \approx 0$, then $\theta \approx \infty$; therefore those zip codes are not grouped in the same zone. It tries to avoid grouping in the same LWA those acts that belong to different postal codes and have not been interleaved by the driver. If he does not consider useful to jump between these codes, it should be considered by the algorithm.
(3) if $s >> d$ then $\theta \approx 0$, therefore these zip codes will be grouped in a community. It represents a high connectivity between close geographical areas.
(4) if $d >> s$ then $\theta >> 0$, therefore these codes are not grouped in the same community, this represents some non usual acts in the shipping process that can appear as punctual requests from clients that will not (probably) be repeated in the future. As a consequence, this driver behavior should not be learned.

Using the zip codes stored in each log, the system generates a set of LWA learning from the human experts using the following algorithm:

(1) For each working day and act selected from a given log and driver, the θ value is computed. Then, depending on the value of θ, it is grouped in the same LWA if $\theta < \delta$, we typed this kind of strong zones as LWA_1. Therefore two acts act_i and act_j are grouped if $\theta(act_i, act_j) < \delta$. The value of δ was obtained from an empirical evaluation of several drivers and working days randomly selected. This value was selected analyzing the LWA_1 generated in the first execution of the algorithm.
(2) Once the selected day has been completely analyzed, usually there are still some acts remaining that have not been grouped in a LWA. These non-grouped acts are now selected and the same procedure is executed, but now higher δ values are used. Two new kinds of zones are defined, LWA_2 if $\theta(act_i, act_j) < 2{*}\delta$ and LWA_3 if $\theta(act_i, act_j) < 3 * \delta$. These new zones can be considered as *weakly connected* in comparison to the previous one because the θ value has been relaxed to allow their grouping.
(3) If there are still non-grouped acts, we define a new type of zone, let us name it LWA_4, that is strictly based on the distance between zip codes. It is computed for all the analyzed acts as the mean of the distances between the zip codes that in this moment has been grouped. The remaining non-grouped acts, those ones whose distances are lower than the previous distance, are selected and grouped in a new LWA.
(4) Finally we define some special LWA zones for those acts that have not been grouped in the previous algorithm steps. We define the LWA_5 to include the community isolated acts; additionally the LWA_6 represents acts with no information, they usually are acts with wrong zip codes that cannot be solved.

Finally, each learned LWA is stored as a new case in the knowledge base. The algorithm is run for each driver and as a result the output is a set of areas. A graphical representation

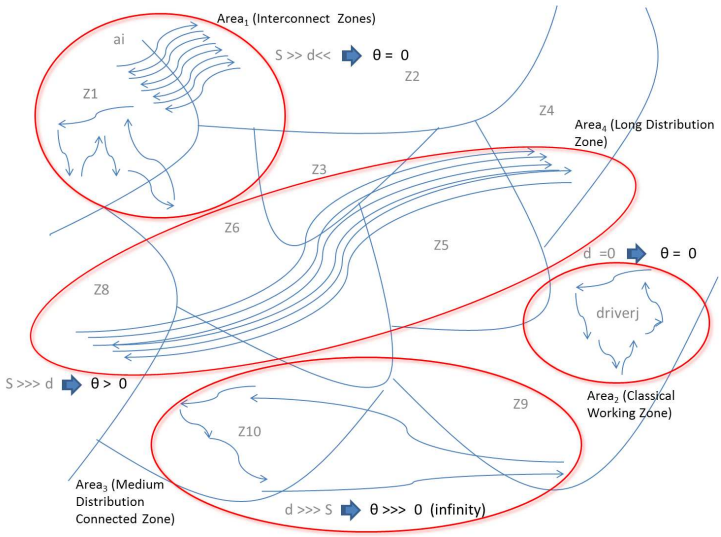

Figure 3. Example of learned WAs generated through the algorithm, those areas are represented with red circles.

of the algorithm output is shown in Figure 3. Once the areas have been learned, sometimes it is quite convenient to visualize them, that is the goal of the LWA representation subsystem.

4.2. *LWA graphical representation*

Although the main goal of the CBR-based algorithm is to learn from the behavior shown by the drivers to use it in the route planning, human supervision and analysis of the results is useful. In order to ease this task, we have developed two different types of representation.

The first representation shows a statistical analysis of a given driver, including information about the number of the LWA considered, type of the zone ($LWA_i, i \in [1..6]$), number of acts belonging to this zone, percentage of acts that belong to this zone, extension (in Km^2) and density of acts ($numberacts/Km^2$).

The second representation is graphical, to ease the analysis of the data. Three different graphical representations have been included:

(1) *Box-based representation.* This is the default representation. The area is represented by a box with basic overlapped information such as driver ID, number of acts, and zone number.
(2) *Marker-based representation.* This representation places a marker for each driver's act. Drivers are represented with a color, so it is easy to compare the acts that belong to two or more drivers. The zone number is shown inside the marker. If any marker is selected, the whole information related to this act is shown.
(3) *Shadow-based representation.* A square is drawn for each act. This square is semi-transparent and zones with higher number of acts are represented with opaque squares. This representation shows in a simple graphical way both, the physical

motions of the drivers and the overlapping zones between two or more drivers.

Once the areas have been learnt through the CBR algorithm, the application has all the components to optimize the routes. How this is done is described in the next section.

5. Route optimization using GA

One of the main features of the proposed system is its capacity to suggest efficient routes. The term efficient in this context should be interpreted as relative to a set of routes designed by a human expert, and they are provided to the system as input. So, the goal of the described system is to improve a given set of routes rather than generate complete new routes. This characteristic is used by the optimization engine to guide its search. The definition of the zones is provided by the CBR previously described, so it does not have to handle this task and thus, the route optimization can be considered as a variation of the well-known TSP. The selection of the optimization algorithm is critical to the success of the system. Evolutionary Computation (EC) provides a set of tools that are well suited to this scenario, in particular, the optimization engine is based on a GA.

5.1. *Using a GA to solve the TSP*

The GA implemented in the application is based on Sengoku's work described in (Sengoku and Yoshihara 1998). This GA encodes the solution using a classical permutation codification (Bäck *et al.* 1984), where the acts are coded as integer numbers between one and the number of acts in a fixed length chromosome. Since no act can be visited twice, integers in the chromosome are not allowed to be repeated. Indeed, the valuable information in this codification is placed in the adjacency rather than in the alleles, the absolute position of the integer and its value does not contain relevant information. In this way, the chromosome $G_1 = \{4, 3, 2, 5, 6, 1\}$ represents the path $4 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 1$, and it is equivalent to another chromosome $G_2 = \{5, 6, 1, 4, 3, 2\}$ representing $5 \rightarrow 6 \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow 2$. The route that they code is the same because the genes have the same adjacency, as can be seen in Figure 4.

GAs require a mechanism to evaluate the quality of individuals, i.e., the fitness function. This is a key subject in any GA design that may determine its success or failure. This is a very basic mechanism used with a selection procedure to push the population to evolve towards fitter positions in the search space. The fitness function must have a direct relationship with the value that the solution provides to the user. The higher fitness value, the better the solution should be for the user. There are several criteria that can be used to optimize routes; users may prefer, for instance, to save time instead of fuel. The design of the fitness function should keep this fact in mind, and should be flexible enough to let the user change the optimization criteria.

In our case, we have used an aggregated scalar fitness function which evaluates the route based on different criteria whose influence can be changed by the user. The fitness function is fed with a human-made route and the LWA provided by the CBR. The assessment of the chromosomes is done comparing the route coded by the individual with the reference route. This comparison uses four criteria or califications: Time, Distance, Zone and Acts. Using these califications we can evaluate different aspects related to the route, such as time or distance. In order to provide an aggregated estimation of the chromosome quality, these califications are aggregated in a linear combination that

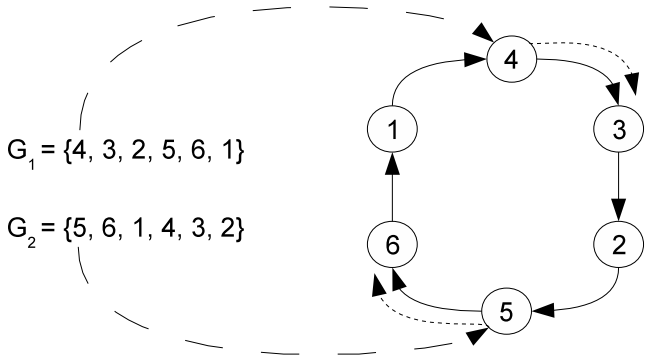

Figure 4. Two different chromosomes (G_1, G_2) coding the same route $4 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 1$.

conforms the fitness function, as is expressed in equation 2.

$$f(A) = \omega_d \Delta_d(A) + \omega_t \Delta_t(A) + \omega_z \Delta_z(A) + \omega_a \Delta_a(A) \tag{2}$$

Where $f(A)$ denotes the fitness of chromosome A, ω_i are coefficients associated to distance (ω_d), time (ω_t), zones (ω_z) and acts (ω_a). These coefficients are used to weight the contribution of each category to the fitness. It is possible to change the califications weight just modifying the coefficients ω_i. The function $\Delta_i(A)$ returns the relative difference between the route codified in the chromosome A and the reference route for each one of the described categories. By default, ω_d and ω_i are set to 0.35 while ω_z and ω_a are both set to 0.15. Based on our experience, we found that those values give good results.

The TSP is a NP-hard problem, and thus finding the optimal solution in the general case -eventually with many acts- is not viable. Therefore, it is not possible to know when a global maximum is achieved and a convergence criteria is required to stop the execution. In this case, the GA is supposed to have converged if (a) the 40% of the last generations have a fitness improvement less than 5% or (b) 20% of the generations has improved the fitness 1% or less. The initial population is generated almost randomly, avoiding repeated individuals.

Our GA uses the same elitist strategy as (Sengoku and Yoshihara 1998). Given a population M of routes in generation i, the N best routes are copied to the generation $i+1$. In an attempt to avoid a premature convergence due to the loose of genetic diversity, those routes whose fitness have a difference less than ϵ are removed. To maintain constant the number of individuals in each generation, $M-N$ individuals are generated by means of a Greedy Subtour Crossover (Sengoku and Yoshihara 1998). This operator takes two random individuals, G_1 and G_2 from the population, and selects one random act. The offspring is constructed taking acts from G_1 to the right and from G_2 to the left until a repeated act is found, then this process is stopped and the remaining acts are filled randomly.

Greedy Subtour Crossover can be better understood with an example. Figure 5 shows the recombination of two chromosomes, $G_1 = \{1, 2, 4, 3, 5\}$ and $G_2 = \{1, 4, 5, 2, 3\}$. The

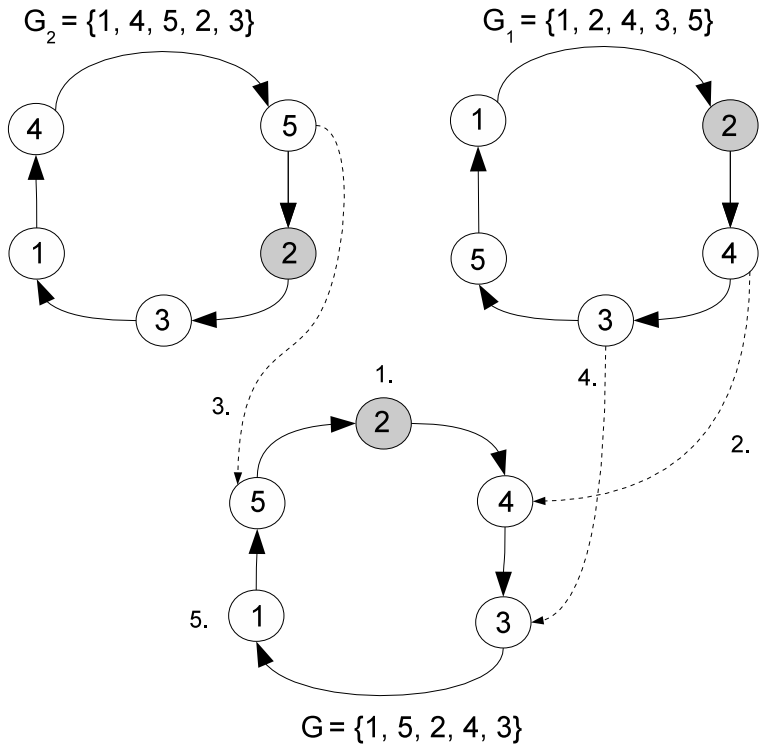

Figure 5. Greedy Subtour Crossover example with chromosomes $G_1 = \{1, 2, 4, 3, 5\}$ and $G_2 = \{1, 4, 5, 2, 3\}$.

sequence of events is the following one:

(1) A random act is selected and copied to the offspring. In this case we start with the act number 2.
(2) The first act to the *right* of 2 in G_1 is copied to the offspring.
(3) The first act to the *left* of 2 in G_2 is copied to the offspring.
(4) The second act to the *right* of 2 in G_1 is copied to the offspring. The second act to the *right* of 2 in G_2, 4, is already present in the offspring, so we stop the iteration.
(5) Remaining acts in the offspring are randomly added. In this case there is only one act left, so it is added to the offspring route.

5.2. *Local search extensions*

A local search has been added to the GA in order to improve its performance. This search has been implemented as an "intelligent" mutation operator called *2opt* and it is applied to a random individual with probability p_m. This genetic operator mutates swapping all non-adjacent pairs of acts in the chromosome, each route built in this way is evaluated and the fittest one is kept. Figure 6 shows an example of how the chromosomes $3, 4, 6, 1, 5, 7, 2$ (left) can be mutated swapping acts (right). This method entails a large number of evaluations, which potentially may consume much computational resources, so the evaluation of the mutation effects should be carefully analyzed.

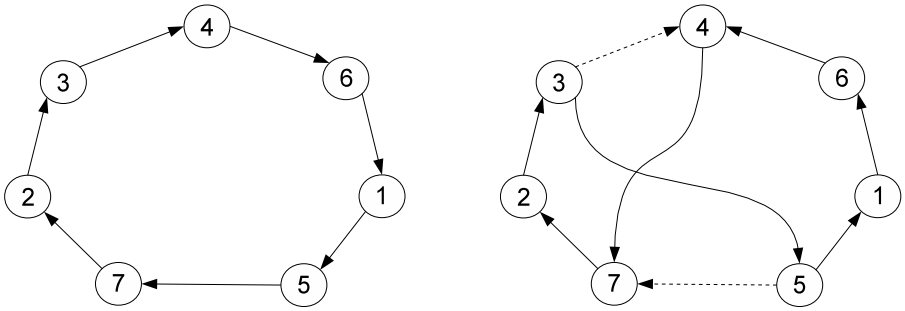

Figure 6. Example of *2opt* mutation. All non-adjacent acts are swapped, however only one in shown.

Depending on the evaluation of the mutation, we define two variations of *2opt* operator that we name light and hard. *Light 2opt* only performs an assessment if the distance of the resulting route is less than the original one, meanwhile *hard 2opt* computes the whole fitness of the route, which is a computationally expensive operation. In case that *2opt* did not generate a fitter individual, it is not modified.

In summary, the algorithm that we have followed in the GA to plan routes is composed by the following steps:

(1) *Generate initial population.* A random initial population with no repetitions is generated.
(2) *Evolve the population.* Once the initial population is created, the following steps will be repeated until the convergence criteria is satisfied.
   a) Selection. The N fittest individuals are copied to the next generation. Individuals whose fitness is equal or less than a threshold are removed.
   b) Crossover. $N-M$ pairs of individuals are randomly selected and recombined using Greedy Subtour Crossover.
   c) Mutation. If the user requires it, *light* or *hard 2opt* mutation is applied with probability p_m.
(3) Results. After a number of iterations, the best planning routes are presented to the user.

6. Experimental results

In this section we report the experimental results collected for each one of the described modules. The dataset used in the experimentation was provided by a logistic company with real historical data. A summary of the attributes dataset was shown in Table 1, and its characteristics were described in subsection 3.1. As a brief summary of the dataset, we can mention that it involves 14 drivers, with 218 average routes per day and driver, with a overall number of ships that equals 10.149. In the following, we assess the two main modules of the system.

Table 2. Percentage of matches for each cleaning algorithm.

	Matches
Block distance	52%
Cosine distance	52%
Dice distance	52%
Jaro-Winkler distance	90%
Levensthein distance	94%
Monge-Elkan distance	40%

Table 3. GA parameters used in the experimentation.

Parameter	Value
Population	60
Elitism size	30
Crossover	Greedy Subtour Crossover
Crossover selection	Random

6.1. *Preprocessing*

One of the main problems we found was the high ratio of mistakes in the input files, mainly the addresses. Without any pre-processing the number of errors in the street names or zip codes is around 35%. After some preprocessing (i.e. removing special characters) the percentage decreased to 20% what still required further processing. After applying the Data Mining algorithms described in subsection 3.1, the number of incorrect streets decreased to only 4%. The remaining mistakes had to be manually corrected.

Different distance metrics yielded different capabilities. Table 2 shows a summary of the capabilities of each method under study. It can be seen that Block, Cosine and Dice distances achieve 52% of matches, which is not a remarkable result. Monge-Elkan is even worse, with only 40%. Therefore the only algorithms whose performance makes them suitable for our purposes are Jaro-Winkler and Levensthein. To be more specific, they match, respectively, 90% and 94%.

6.2. *Route optimization with GAs*

Once the input has been cleaned and the zones delimited with the CBR, the GA may optimize the routes. In this section we aim to study the behaviour of the GA with a special emphasis on the influence of the *2opt* operator in the performance of the algorithm due to its computational cost. All the experiments shown in this section are average values of 20 runs. The main parameters used by the GA are shown in Table 3.

Since *2opt* may be a computationally expensive algorithm, it is worth studying it in terms of its cost. We used time instead of a hardware-independent measure, such as the number of generations because the latter does not compute the resources needed to execute the algorithm in one generation. So, convergence time seems to be a fair comparative measure. It takes into account the resources consumed by the algorithm, as

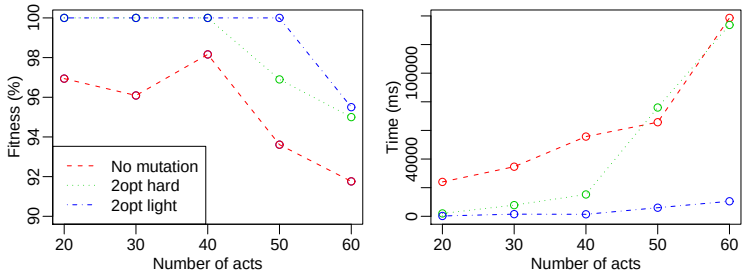

Figure 7. Effects of the 2opt mutation operator: Fitness (left) and run-time (right). The 2opt light operator reduces substantially the computation time, being more evident with a high number of acts. Using any of these two mutation operators enhance the quality of the solutions.

well as the time saving introduced by the operator.

Figure 7 (right) compares the time required by the GA to converge with the mutation operators under study. It can be seen that *2opt light* is the method that requires less time to converge. Interestingly, it found solutions consuming less time using *2opt light* that not using any mutation operator at all. For 40 acts or less, there is a significant difference of the convergence time between *2opt hard* and no mutation, it is clear that using *2opt hard* mutation reduces the convergence time in that case, however the performance improvement disappears when there are more than 40 acts. The execution time saved by *2opt hard* does not pay off the computational cost overrun for any number of acts.

We can observe the quality of the solutions found by the GA in the Figure 7 (left). From the quality point of view, the mutation operators introduce a substantial improvement: Not using mutation yields worse solutions, as can be seen in the figure. However, the difference between *2opt hard* and *light* is not so evident. They show similar performance for any number of acts, except when there are 50 acts, in that case the *2opt light* operator slightly outperforms *2opt hard*, but this difference does not seems to be statistically significant.

So, we conclude that the *2opt* mutation operator in any of its versions improves the quality of the solution. As a side effect, the computational overhead introduced by the *2opt light* is compensated by a substantial improvement in the search process that, in overall terms, reduces the computation time required to find a solution.

6.3. *The application from the user perspective*

It is interesting to compare the plans generated by a human expert and the system described in this paper. We have observed that plans made by humans used to be more efficient at the short-term, while in the last part of the shipping the planned route beats the original human-made plan. The reason behind this fact could be explained, to some extent, by a tendency to begin the route close to the starting point, i.e., humans tend to be more efficient optimizing the beginning of the route. On the contrary, the GA tends to optimize all the route instead of just its begining. This fact is better illustrated with a route example.

Figure 8 represents the distance of a route planned by an human expert (red line) and

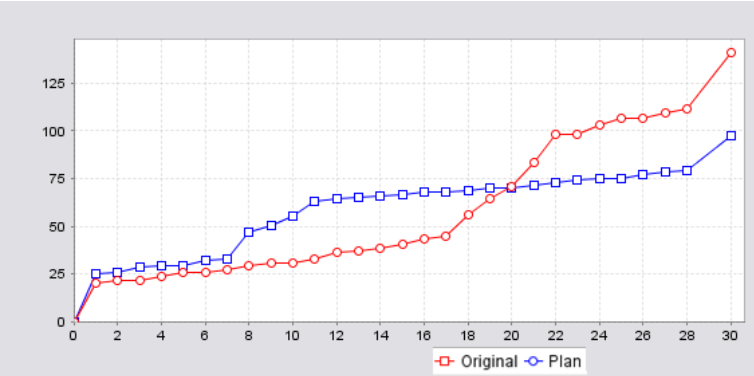

Figure 8. Distance comparison between the human-made route (red) and the GA-planned route (blue). The distance of the GA-planned route (97Km) is shorter than the human-made one (140 Km).

that route optimized by the GA (blue line). Initially, the distance of the human-made route is less than the GA-planned route, for instance 64.167 Km against 70.111 Km for 20 acts. But afterwards that point, the GA-planned itinerary obtains lower distance and is able to shorten the total route distance (140.729 Km against 97.041 Km). The tendency of humans to optimize the short-term might be an explanation of this result.

Another tendency that can be observed in routes generated by human experts is related to how shipping are placed within a street, humans tends to place those shipping together, even when that is suboptimal. Although it seems a logical reasoning, some streets are long enough to make it wrong, and that happens in practice, as our experiments have shown.

7. Conclusions

In this paper we have introduced an AI-based solution that has been designed to help a logistic company to improve its routes generation processes. The application uses some classical AI methods such as CBR, Data Mining and GAs. Using this bunch of algorithms the solution is able to extract the human knowledge that is implicitly stored in the company historical data about past routes and use it in the long-term, improving the efficiency of the routes.

Data Mining has been applied to extract knowledge from the data saved through the information system, and to store it in a manageable structure by the other modules of the tool. Using CBR we have been able to obtain critical business intelligence knowledge from the drivers experience that later can be used by the own company to optimize their business processes. Based on that knowledge, it is possible to detect, find and delimit the best working zones and identify daily incidents. These can be used to generate new routes, avoiding non-desirable routes and taking important advantages using the available data that is regularly updated in the system. This knowledge has been adequately adapted to be used by the optimization GA algorithm which is able to generate non-standard solutions according to the behaviour of the human drivers. Finally, the experimental results obtained show that the proposed approach is able to find routes that improve, in average, the routes made by human experts.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Innovation under the projects ABANT (TIN 2010-19872) and by Jobssy.com company under Project FUAM-076913.

References

Aamodt, A. and Plaza, E., 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications*, 7 (1), 39–52.

Agustín-Blas, L.E., *et al.*, 2012. A new grouping genetic algorithm for clustering problems. *Expert Systems with Applications*, 39 (10), 9695–9703.

Anuar, N. and Sultan, A.B.M., 2010. Validate Conference Paper using Dice Coefficient. *Computer and Information Science*, 3 (3), 139–145.

Bäck, T., Fogel, D.B., and Michalewicz, Z., 1984. Permutations. *In*: *Evolutionary Computation 1. Basic Algorithms and Operators.*, 139–149 Institute of Physics Publishing.

Beheshti, H.M. and Beheshti, C.M., 2010. Improving productivity and firm performance with enterprise resource planning. *Enterprise Information Systems*, 4 (4), 445–472.

Bhattacharyya, M. and Bandyopadhyay, A.K., 2008. Comparative study of some solution methods for traveling salesman problem using genetic algorithms. *Cybernetics and Systems*, 40 (1), 1–24.

Bin, X., *et al.*, 2010. Improved Genetic Algorithm Research for Route Optimization of Logistic Distribution. *In*: *Proceedings of the 2010 International Conference on Computational and Information Sciences*, ICCIS '10 Washington, DC, USA: IEEE Computer Society, 1087–1090.

Bishop, C.M., 2007. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1st ed. 2006. Corr. 2nd printing Springer.

Buriol, L., Frana, P.M., and Moscato, P., 2004. A New Memetic Algorithm for the Asymmetric Traveling Salesman Problem. *Journal of Heuristics*, 10, 483–506.

Changjiang, Z., *et al.*, 2009. GA for New Logistics Distribution Routing Optimization Model with Practical Application. *In*: *Proceedings of the 2009 WRI Global Congress on Intelligent Systems - Volume 02*, GCIS '09 Washington, DC, USA: IEEE Computer Society, 25–29.

Chiang, D.M.H., Lin, C.P., and Chen, M.C., 2011. The adaptive approach for storage assignment by mining data of warehouse management system for distribution centres. *Enterprise Information Systems*, 5 (2), 219–234.

Craw, S., 2010. Manhattan Distance. *In*: C. Sammut and G.I. Webb, eds. *Encyclopedia of Machine Learning*. Springer, p. 639.

Duan, L. and Xu, L.D., 2012. Business Intelligence for Enterprise Systems: A Survey. *Industrial Informatics, IEEE Transactions on*, 8 (3), 679 –687.

Emmert-streib, F., 2011. Structural Analysis of Complex Networks. *Analysis*, 3 (c), 1–9.

Fan, L. and Meng, H., 2010. Application Research on Optimal Path Based on Genetic Algorithm.. *JDCTA*, 4 (8), 199–202.

Fellegi, I.P. and Sunter., A.B., 1969. A theory for record linkage. *Journal of the American Statistical Association*.

Fischer, K., Mller, J.P., and Pischel, M., 1995. Cooperative Transportation Scheduling: an Application Domain for DAI. *Journal of Applied Artificial Intelligence*, 10, 1–33.

Fogel, D.B., 1988. An Evolutionary Approach to the Traveling Salesman Problems. *Biological Cybernetics*, 60, 139–144.

Hanif, S., *et al.*, 2011. Delegate MAS for Large Scale and Dynamic PDP: A Case Study.. *In*: F.M.T. Brazier, K. Nieuwenhuis, G. Pavlin, M. Warnier and C. Badica, eds. *IDC*, Vol. 382 of *Studies in Computational Intelligence* Springer, 23–33.

He, W., *et al.*, 2009a. Insight into interface design of web-based case-based reasoning retrieval systems. *Expert Syst. Appl.*, 36 (3), 7280–7287.

He, W., *et al.*, 2009b. Integrating web 2.0 with the case-based reasoning cycle: A systems approach. *Systems Research and Behavioral Science*, 26 (6), 717–728.

Holland, J.H., 1992. *Adaptation in natural and artificial systems.* Cambridge, MA, USA: MIT Press.

Ilie, S. and Bdic, C., 2011. Multi-agent approach to distributed ant colony optimization. *Science of Computer Programming*, (0), In press.

Jaro, M., 1995. Probabilistic Linkage of Large Public Health Data Files. *Statistics in Medicine*, 14 (5-7), March–April.

Jiang, Y., *et al.*, 2009. Influencing factors for predicting financial performance based on genetic algorithms. *Systems Research and Behavioral Science*, 26 (6), 661–673.

Jung, J.J., 2011. Service chain-based business alliance formation in service-oriented architecture. *Expert Systems with Applications*, 38 (3), 2206–2211.

Ko, H. and Evans, G., 2007. A genetic algorithm-based heuristics for the dynamic integrated forward/reverse logistics network for 3PLS. *Computers and Operations Research*, 34, 346–366.

Kumar, V. and Viswanadham, N., 2007. A CBR-based Decision Support System Framework for Construction Supply Chain Risk Management. *In*: *Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering. Scottsdale, AZ, USA*.

LaRusic, J., Punnen, A., and Aubanel, E., 2012. Experimental analysis of heuristics for the bottleneck traveling salesman problem. *Journal of Heuristics*, 18, 473–503.

Levenshtein, V., 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady.*

Li, L., 2012. Effects of enterprise technology on supply chain collaboration: analysis of China-linked supply chain.. *Enterprise IS*, 6 (1), 55–77.

Liu Xiao-Yan, C.Y.L., 2010. Application of Dijkstra Algorithm in Logistics Distribution Lines. *In*: *Proceedings of the Third International Symposium on Computer Science and Computational Technology*, Jiaozuo, P. R. China, 48–50.

March, B.S.T. and Storey, V.C., 2008. Design science in the information systems discipline: an introduction to the special issue on design science research. *MIS Quarterly*, 32 (4), 725–730.

Martínez-Bernabeu, L., Flórez-Revuelta, F., and Casado-Díaz, J.M., 2012. Grouping genetic operators for the delineation of functional areas based on spatial interaction. *Expert Systems with Applications*, 39 (8), 6754 – 6766.

Minis, I., Mamasis, K., and Zeimpekis, V., 2012. Real-time management of vehicle breakdowns in urban freight distribution. *Journal of Heuristics*, 18, 375–400.

Monge, A. and Elkan, C., 1997. An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. *In*: *Proceedings of the SIGMOD Workshop Data Mining and Knowledge Discovery* ACM, 267–270.

Neagu, N., *et al.*, 2006. LS/ATN: Reporting on a Successful Agent-Based Solution for Transport Logistics Optimization. *In*: *Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications* Washington,

DC, USA: IEEE Computer Society, 213–218.

Newcombe, H., *et al.*, 1959. Automatic linkage of vital records.. *Science*.

Pisinger, D. and Ropke, S., 2007. A general heuristic for vehicle routing problems. *Comput. Oper. Res.*, 34 (8), 2403–2435.

Qian, G., *et al.*, 2004. Similarity between Euclidean and cosine angle distance for nearest neighbor queries. *In*: *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, Nicosia, Cyprus New York, NY, USA: ACM, 1232–1237.

Romero, J. and Machado, P., eds. , 2007. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Natural Computing Series Springer Berlin Heidelberg.

Sadeghi-Niaraki, A., *et al.*, 2011. Real world representation of a road network for route planning in GIS. *Expert Syst. Appl.*, 38 (10), 11999–12008.

Savelsbergh, M.W.P. and Sol, M., 1995. The General Pickup and Delivery Problem. *TRANSPORTATION SCIENCE*, 29 (1), 17–29.

Sengoku, H. and Yoshihara, I., 1998. A Fast TSP Solver Using GA on JAVA. *In*: *3rd AROB*.

Starkweather, T., *et al.*, 1991. A Comparison of Genetic Sequencing Operators. *In*: *Proceedings of the fourth International Conference on Genetic Algorithms* Morgan Kaufmann, 69–76.

Winkler, W., 1999. The state of record linkage and current research problems. *Statistics of Income Division, Internal Revenue Service*.

Witten, I.H. and Frank, E., 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd The Morgan Kaufmann Series in Data Management Systems San Francisco, CA: Morgan Kaufmann Publishers.

Wooldridge, M. and Jennings, N.R., 1995. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10 (2), 115–152.

Yan, J., Chaudhry, P., and Chaudhri, S., 2003. A model of a decision support system based on Case-Based Reasoning for third party logistics evaluation. *Expert Systems*, 20 (4), 196–207.

Zhao, J.q. and Wang, L., 2011. Center Based Genetic Algorithm and its application to the stiffness equivalence of the aircraft wing. *Expert Systems with Applications*, 38, 6254–6261.