Solving the Dynamic Traveling Salesman Problem using a Genetic Algorithm with Trajectory Prediction: An application to Fish Aggregating Devices

Groba, Carlos* Sartal, Antonio† Vázquez, Xosé H.‡

Abstract

The paper addresses the synergies from combining a heuristic method with a predictive technique to solve the Dynamic Traveling Salesman Problem (DTSP). Particularly, we build a genetic algorithm that feeds on Newton's motion equation to show how route optimization can be improved when targets are constantly moving. Our empirical evidence stems from the recovery of fish aggregating devices (FADs) by tuna vessels. Based on historical real data provided by GPS buoys attached to the FADs, we first estimate their trajectories to feed a genetic algorithm that searches for the best route considering their future locations. Our solution, which we name Genetic Algorithm based on Trajectory Prediction (GATP), shows that the distance traveled is significantly shorter than implementing other commonly used methods.

Keywords

Supply Chain Sustainability, Dynamic Traveling Salesman Problem, Genetic Algorithms, Fish Aggregating Devices.

1 Introduction

The Traveling Salesman Problem (TSP) probably represents the most intensive area of research within the wide range of combinatorial optimization problems (Golden et al, 1987; Gutin and Punnen, 2002). Whereas the diverse perspectives and problem-solving methods have helped practitioners and scholars to address a multitude of different problems in different industries (Grötschel and Padberg, 1985; Lawler et al, 1985; Duchenne et al, 2007; Donald, 2010), the literature on TSP is still underdeveloped with regard to moving

*cgroba@uvigo.es; University of Vigo.

†antoniosartal@uvigo.es; University of Girona.

‡Corresponding author: xhvv@uvigo.es; University of Vigo and Pompeu Fabra University; Facultade de Economía. Rúa Leonardo da Vinci. 36310 Vigo. Spain. tel: +34986812479; fax: +34986812401.

targets -such as in the fishing or military industries (Helvig et al, 2003). In this case, the most recent approaches (which can be grouped under the heading "Dynamic Traveling Salesman Problem"-DTSP) work on a real time basis to find the changes between nodes (Pantrigo and Duarte, 2013); nevertheless, they do not anticipate the future movement of targets, so the optimal solution is given only when changes happen and the algorithm is subsequently recalculated.

With this academic background in mind, we faced the problem of tuna vessels that pick up fish aggregating devices (FADs) at sea. When FADs transmit information on how much tuna might be available beneath them, the vessels need to design a route taking into consideration that FADs are constantly moving. They need to minimize distance while recovering the FADs because saving time and fuel determines their competitiveness. Using therefore real data, the paper contributes to the literature by proposing a new approach that combines a heuristic method with a predictive technique. Particularly, we first estimate the trajectories of the FADs to subsequently build a genetic algorithm that uses this information and searches for the best possible route considering their future locations.

From all heuristic methods, we chose GAs for their properties (they are evolutionary, show statistical convergence, and tend to a global optimum with considerable robustness) and because they offer vessels the possibility to reach a solution within an acceptable computational time (Jih and Hsu, 2004; Bjarnadóttir, 2004). On the other hand, we chose Newton's movement equation as a predictive technique (we show a performance comparison with other techniques for illustrative purposes) because it offers vessels a sound and quick forecast of the future position of FADs with very little information. By combining both tools in a single method, which could be named Genetic Algorithm based on Trajectory Prediction (GATP), we reach a global optimization solution with statistically better results than those offered by commonly used methods, such as the Nearest Neighbour (NN) strategy or simple Genetic Algorithms (GA)(Konak et al, 2006; Pérez, 2004).

The following section presents a survey of the relevant literature that guides our approach. Section three describes the tuna vessels FAD recovery problem. Section four compares different prediction models in order to show that the final choice (in our case Newton's motion equation) depends on the specific characteristics of each forecasting initiative. Section five shows the experimental design, section six discusses results and, finally, section seven concludes highlighting the main contributions of the paper and their implications.

2 Literature review

Calculating the optimal route for recovering N moving elements lies within the Traveling Salesman Problem (TSP) (Gutin and Punnen, 2002). Given a list of cities and their pairwise distances, the task is to find the shortest possible route to visit each city only once

and then return home (Applegate et al, 2007). Not surprisingly, the initial applications to real world problems were mainly in transportation and logistics (Dantzig et al, 1954).

Scholars soon perceived, however, that further applications could be feasible if they interchanged the city concept with, for example, soldering points or DNA fragments, and the distance concept with other constraints like traveling times, cost or time windows. Further developments thus appeared in such diverse fields as crystal structure analysis (Bland and Shallcross, 1989), the drilling of printed circuit boards (Grötschel et al, 1991) or even the mapping of a mouse genome (Avner et al, 2001). Certainly, the diverse applications also triggered the development of new problem-solving methods (Tyedmers and Parker, 2012), from exact algorithms to metaheuristics (Blum and Roli, 2003), such as Swarm Intelligence (Bonabeau et al, 1999) or GAs (Donald, 2010).

GAs represent in fact one of the most consolidated approaches to the TSP (Potvin, 1996). They were first introduced by Holland (1975) to generate solutions for optimization problems using techniques inspired by natural evolution (Winter et al, 1996), leading to many theoretical developments over the last thirty years (Reinelt, 1994; Smith and Smith, 2002).

Basically, GAs achieve the optimal solution from a random set of initial solutions called population. Each set comprises an array of numbers where each number represents one of the targets on the route, which are named genes. Hence, each population is evaluated by a fitness measure (in our study, for instance, the measure is determined by the minimal distance between all points on each route), so parents of the next generation are selected probabilistically from the whole population so that the best routes are selected to become the parents of the next generation. The process is regulated by operators reflecting typical gene traits such as *crossover* and *mutation*. GAs repeat this loop until they converge to a near global optimal.

Recently, some scholars have intensified the use of GA to implement theoretical developments in different fields of application such as ship routing with time deadlines (Karlaftis et al, 2009), vehicle routing with time windows (Dumas et al, 1995; Garcia-Najera and Bullinaria, 2011; Wang and Regan, 2002; Baker and Ayechew, 2003) and vehicle routing with loading constraints (Ruan et al, 2013). Despite the progress that implementing GAs brought to the literature on route optimization, however, their potential has not been fully exploited when addressing the TSP with constantly moving targets.

GAs generate near-optimal solutions only when cities are at time $t = 0$; but, in a dynamic scenario, the salesman needs to decide a route for $t = 1$, $t = 2$, etc. The final route that the salesman should follow is therefore necessarily different from the one chosen by a conventional approach to static objectives. This is probably the reason why recent literature has increasingly dealt with dynamic targets, leading to a new line of research in this field since Psaraftis (1988) introduced a first reflection on the Dynamic Traveling Salesman Problem (DTSP).

Some contributions compare DTSP and TSP and reflect on basic issues to solve the

problem, appropriate approaches, or key evaluation criteria (Huang et al, 2001; Zhou et al, 2003). Most of the literature, however, presents specific applications based on well-known metaheuristics such as Ant Colony Optimization (Eyckelhof and Snoek, 2002; Guntsch et al, 2001), Simulated Annealing (Jeong and Kim, 1991), Tabu Search (Fiechter, 1994) and Genetic Algorithms (Moon et al, 2002; Younes et al, 2003; Liu et al, 2009), under which we can also include particular offshoots like inver-over operators (Li et al, 2006; Yan et al, 2007) or CHC Algorithms (Simões and Costa, 2011). All this work represents a generalization of TSP in which targets are not necessarily static and applications are often formulated with time-dependent variable constraints.

Taking this background into account, our approach resembles that of the existing literature on DTSP but differs in an important way. Both assume the dynamic nature of targets, but the available DTSP solutions work basically on a real time basis to find the changes between nodes (Zhou et al, 2003; Hajjam et al, 2013). The main DTSP methods (Pantrigo and Duarte, 2013) consist in fact in (i) restarting the search method from start, which entails that problems are dealt with as a series of static optimization problems with no relation to each other, and (ii) starting from the best solutions found before the last event, which has found different methodological alternatives. Garrido and Riff (2010), for instance, use a hyper heuristic approach generating a set of low-level heuristics; Pantrigo and Duarte (2013) rely on a Scatter Search Particle Filter (SSPF) to take advantage of the best solutions obtained in the previous executions; whereas Hajjam et al (2013) employ an intermediate structure in a hybrid method that manipulates the self-organizing map in order to minimize route lengths and customer's waiting time. Hence, although DTSP has allowed us to gain new insights by addressing new problems, the current literature does not anticipate the future movement of targets; optimal solutions are given only when changes happen and the algorithm is subsequently recalculated. By contrast, our approach follows the line of research drawn by the literature on DTSP, but assumes that changes in the localization of targets are small, traceable and time dependent. This allows us to estimate the trajectories of the targets, as explained above, to subsequently build a genetic algorithm that searches for the best possible route considering their future locations.

3 The tuna vessel FAD recovery problem

An FAD is a man-made object used to attract ocean-going pelagic fish, such as tuna, which gathers around it for reasons that are still unclear (Bach et al, 1998). Most purse seiners for tuna therefore release FADs into the sea, letting them float on the water surface following the ocean currents. Some of them are furthermore equipped with echo-sounders that transmit information about their localization and the aggregated biomass beneath them (Castro et al, 2001). Vessels can thus receive new messages from the buoys, and each buoy position is updated at least every 12 hours (latitude and longitude). These messages are transmitted automatically in a predictable and controlled way, communicating in real time via satellite telecommunication systems such as Argos, Inmarsat, Orbcomm and

Iridium (Moreno et al, 2007).
Each tuna vessel can handle a different number of FADs, but the maximum per vessel could be some hundreds. Each of them drifts on a particular course and speed, and these conditions change with time because they depend basically on the sea currents under the FAD and on superficial wind. Their speed can thus range from 0.2 knots[1] to 2 knots when they are in areas with strong currents, but on average they travell at about 1 knot. Most tuna vessels recover their FADs still today following the Nearest Neighbor strategy (NN) (or even without any plan at all). The advantage of this method is that it is easy to implement, as the next target will always be the closest to the vessel. The amount of nautical miles covered by vessels, however, is far from being optimal. In order to calculate the NN with dynamic targets, it is necessary to take the next recovery decision once the previous one has been taken, and once the rest of the targets' movements have been ascertained in order to figure out which will be the closest at time $t + t'$.

Figures 1 and 2 show a real scenario of FADs drifting on the Indian Ocean. Each black point represents a different buoy, whereas the red line represents the past positions of the object sent via satellite. The time difference between positions is generally 12 hours or less.

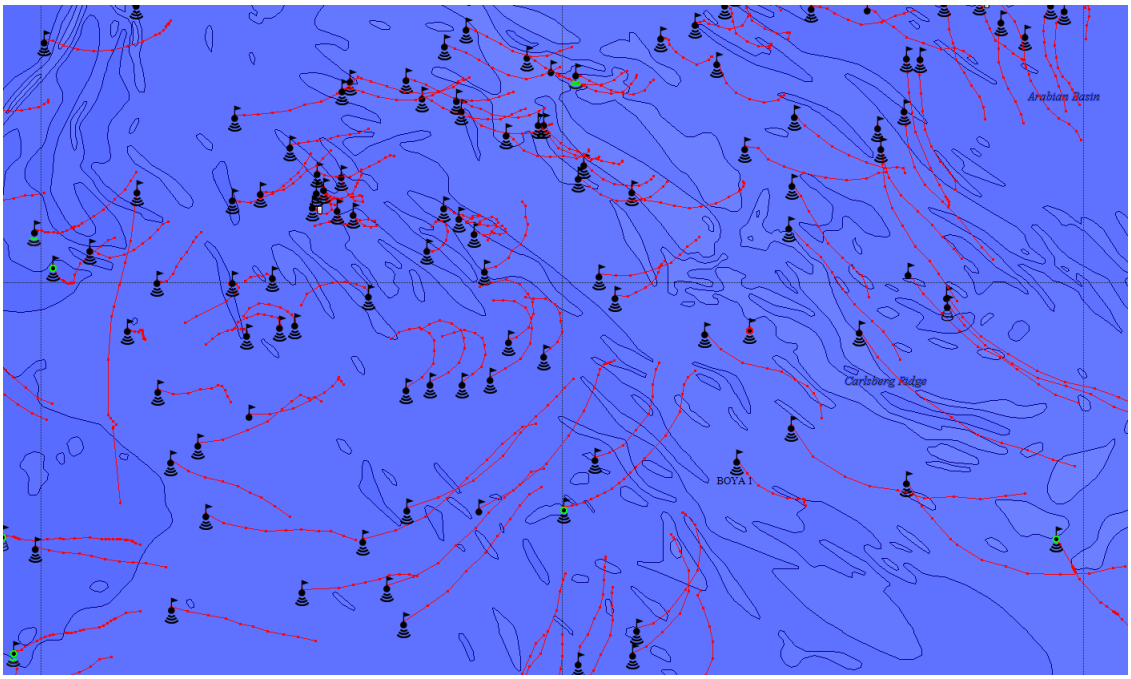

Figure 1: FADs drifting in the Indian Ocean

[1] 1 knot = 1 nautical mile per hour; 1,852 kilometers per hour

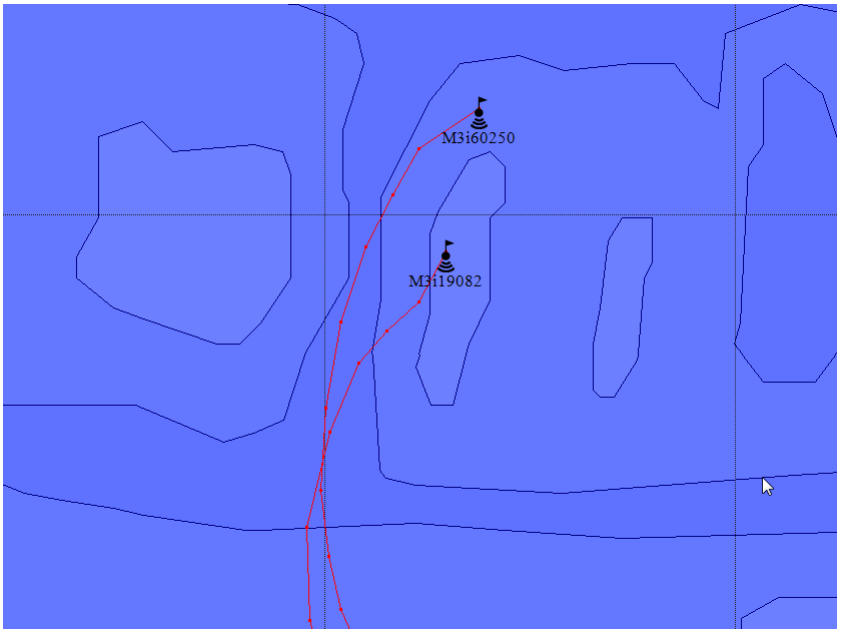

Figure 2: Two FADs drifting in the ocean

4 Drifting object prediction

The literature offers different methods to simulate and predict current movements in the sea (Özgökmen et al, 2000), but all are based on complicated mathematical models which require data that a vessel cannot easily obtain, such as wind data and eddy effects. Furthermore, by contrast with Lagrangian buoys, which have scientific purposes, FADs are made without any standard. They are made by fishermen who use simple materials like wood, string and net, so two FADs cannot be assumed to drift identically under the same conditions. It is not possible therefore to use models based on standard Lagrangian buoys to predict the future positions of FADs in the sea.

In this context, we address the drifting FAD prediction problem from another, simpler point of view; one that requires no more data than the last position of each object. As soon as the buoys transmit their subsequent positions, the algorithm will update the last position and will be calculated again to predict the best route, following the current position of the buoy. So if the prediction for a specific FAD is not accurate enough to predict the best route, the solution will be updated when the next message is received, therefore showing a better optimal route if one exists.

Among the most simple prediction methods available, we have selected to compare two easy-to-implement tools: time series forecasting and Newton's motion equation. These prediction methods are valid for any FAD, regardless of the ocean where it is drifting; however, their effectiveness decreases with time because the error has a cumulative effect. Be that as it may, our goal is to predict where the FAD will be in the near future; not

only its next position, but many future positions.

4.1 Time series

A time series is a sequence of data points that are measured at uniform time intervals. Time series forecasting, in turn, refers to a model that predicts future events based on past values (Casdagli, 1989). Among the methods to perform this type of analysis, the autoregressive model (AR) is very often used to predict the future position of objects (Reinelt, 1994; Besse et al, 2000). It predicts the output of a system based on its previous outputs.

There are a number of different notations for time-series analysis, among which one of the most common is the following:

$$Y = \{Y_t : t \in T\}$$

This $AR(p)$ notation indicates an autoregressive model of order p (number of lags). The $AR(p)$ model is defined as:

$$y_{t+1} = c + \sum_{i=1}^{p} (\alpha_i y_{t+1-i}) + \varepsilon_t$$

where $\alpha_1, \ldots, \alpha_p$ are the parameters of the model, c is a constant (often omitted for simplicity) and ε_t is the error. AR models are easy to calculate and are widely used as predictors for time series analysis of, for instance, stock markets, etc. (Marcellino et al, 2006; Zhang and Qi, 2005).

4.2 Newton's motion equation

Motion equations describe the behavior of a system as a function of time. More specifically, the equations of motion describe the behavior of a physical system as a set of mathematical functions in terms of dynamic variables: normally spatial coordinates and time are used. The Newton's motion equation is considered between two points of time: one initial point and one current or final point:

$$y_{t+1} = y_t + v_t \Delta_t + \tfrac{1}{2} a_t \Delta_t^2$$

where:

- y_t: the position at the end of the interval (displacement)
- v_t: the velocity at the end of the interval t
- Δ_t: the time interval between the initial and current states
- a_t: acceleration at time t

Notice that in the rest of the article we will refer to the time variable t as discrete. Although FADs are constantly moving in the sea, the information on their position is given twice a day because of the airtime cost inherent to satellite communications.

Given that the acceleration value depends on the speed of the object, in our discrete case its calculation requires the last two positions of the object:

$$a = \frac{\Delta_v}{\Delta_t}$$

4.3 Model comparison

Twelve real random FADs were selected from all the oceans to compare how well the prediction methods forecast their next position. Each position is a pair of two single elements: latitude and longitude. Independence between latitude and longitude is assumed so, for a single position prediction, the model estimates two independent parameters, latitude and longitude, which are estimated using the same equation.

The first step is to compare the different time series to see what order suits better the real tracking of FADs. Ordinary Least Squares (OLS) was used to determine AR, and the order of AR was selected using MAPE (Mean Absolute Percentage Error).

MAPE expresses accuracy as a percentage of the actual data and is defined by the formula:

$$M = \frac{1}{N} \sum_{t=1}^{N} \left| \frac{(A_t - F_t)}{A_t} \right|$$

where A_t is the actual value, F_t is the predicted value and N the number of fitted points. MAPE therefore provides an intuitive way to asses the importance or errors, since it easily reflects, for instance, that an error of 10 when the actual value is 100 (10% error) is worse than an error of 10 when the actual value is 1000 (1% error).

We have checked that first order time series predicts much better than other time series with more order lags. The rationale is that the speed of FADs ranges from 0,2 to 2 knots, so the best approach to the next position will be close to the last position, which is what the first order equation suggests.

This first order series is very simple to calculate and only requires the last position of each object, but it has one important limitation. It predicts the future rather accurately once the α_1 parameter value has been set for each FAD. This means that, before predicting the next positions, it is necessary to analyze all the past positions, calculate the best parameter (coefficient) and only then can the equation be used to predict future positions for that FAD (but only for that FAD). If we need to predict future positions of other FADs, the optimal coefficient needs to be re-calculated.

Accordingly, in order to estimate the future movement of FADs with a time series model, we will consider two options: The first stems from estimating the average of all the coefficients, which would be close to 1 ($\alpha \simeq 1$). This is named a random walk model,

with the form $y_{t+1} = y_t + \varepsilon_t$, therefore suggesting that the next movement of an FAD is only determined by its last position (which entails there is no movement). The second option estimates the coefficient for each FAD. We will then compare these two time series models with Newton's motion equation.

$$y_{t+1} = y_t \tag{1}$$
$$y_{t+1} = \alpha_1 y_t \tag{2}$$
$$y_{t+1} = y_t + v_t \Delta_t + \tfrac{1}{2} a_t \Delta_t^2 \tag{3}$$

The information available on each object is latitude and longitude in degrees, and the time difference between each position is 12 hours. Each buoy in the study has a minimum of 200 samples, which is enough to establish the coefficients for each buoy and to compare the different methods properly. Table 1 shows some real data from a buoy and how the latitude and longitude change with time. In this case, day 0 refers to the last position, whereas each half-day step is the former position that the buoy sends every 12 hours.

Table 1: Buoy data samples

An example of buoys data		
Days	**Latitude**	**Longitude**
0.00	-8.07450	53.08117
0.50	-8.20417	52.92067
1.00	-8.43233	52.75367
1.50	-8.67017	52.56317
2.00	-8.69100	52.45533
2.50	-8.60883	52.27750
3.00	-8.60100	52.08517
3.50	-8.61000	51.87983
4.00	-8.60050	51.78600
4.50	-8.51550	51.74350
5.00	-8.38250	51.67350
5.50	-8.28683	51.54150
6.00	-8.22217	51.45200
6.50	-8.18550	51.41283
7.00	-8.10300	51.42383
7.50	-8.00983	51.43133
8.00	-7.94450	51.38983
8.50	-7.88600	51.34833
9.00	-7.84233	51.33783
9.50	-7.78750	51.24567
10.00	-7.81867	51.19767
10.50	-7.87417	51.12983
11.00	-7.98883	51.09517
11.50	-8.03833	51.08083
12.00	-8.07300	51.07117
12.50	-8.08317	51.01017
13.00	-8.13817	50.92533
13.50	-8.22083	50.84883
14.00	-8.27383	50.85417
14.50	-8.21600	50.81000

The results are shown in table 2: Newton's motion equation has less error in average than the other two methods (even if we compare it with the specific $AR(1)$ for each FAD). This is an important result because FAD positions can be therefore estimated without great computational costs. By contrast, it is worth noting that Newton's motion equation only works for short time predictions because the error increases with time. Forecasting the trajectory of an FAD for more than five days would require a more complex prediction

Table 2: MAPE for three prediction methods

Buoys	Lat/Lon	Ramdom walk	AR(1)	Newton
Buoy 1	Latitude	0,84%	0,84%	0,82%
	Longitude	0,22%	0,18%	0,24%
Buoy 2	Latitude	0,46%	0,47%	0,48%
	Longitude	0,11%	0,11%	0,08%
Buoy 3	Latitude	1,31%	1,28%	0,63%
	Longitude	0,27%	0,29%	0,06%
Buoy 4	Latitude	1,69%	1,67%	0,75%
	Longitude	0,25%	0,32%	0,06%
Buoy 5	Latitude	5,33%	5,37%	1,02%
	Longitude	0,07%	0,14%	0,02%
Buoy 6	Latitude	0,30%	0,33%	0,25%
	Longitude	0,05%	0,13%	0,03%
Buoy 7	Latitude	1,27%	1,28%	0,77%
	Longitude	0,15%	0,23%	0,03%
Buoy 8	Latitude	6,28%	6,30%	3,86%
	Longitude	0,39%	0,44%	0,12%
Buoy 9	Latitude	32,37%	32,36%	18,40%
	Longitude	0,10%	0,16%	0,03%
Buoy 10	Latitude	2,81%	2,79%	0,88%
	Longitude	0,25%	0,27%	0,05%
Buoy 11	Latitude	10,04%	10,04%	5,05%
	Longitude	0,21%	0,26%	0,07%
Buoy 12	Latitude	30,16%	30,16%	59,98%
	Longitude	0,69%	0,72%	0,19%

method, probably internalizing chaotic modeling as with ocean currents or weather forecast (Casdagli, 1989). This is not needed here, however, because vessels spend normally less than five days fishing and recover two to three FADs each day. Future research can indeed identify applications where this effort is useful.

5 Methodology

5.1 Data on buoys and vessels

5.1.1 Buoys input

If our problem has N drifting objects, being each b_i an FAD:

$$(b_1, b_2, \ldots, b_N)$$

And for each of these objects we know their current position and the last M positions, then the input we have is an $N \times (M+1)$ matrix:

$$\begin{pmatrix} b_1^t & b_1^{t-1} & b_1^{t-2} & \cdots & b_1^{t-M} \\ b_2^t & b_2^{t-1} & b_2^{t-2} & \cdots & b_2^{t-M} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ b_N^t & b_N^{t-1} & b_N^{t-2} & \cdots & b_N^{t-M} \end{pmatrix}$$

Where the first column are all the objects in the current time $t = t$, the second column

are all the objects in $t = t - 1$, following the progression up to the last column, where we have the objects at $t = t - M$.

Each object b^t has two coordinates, latitude and longitude, to plot the real position on a 2D map:

$$b^t = (latitude^t, longitude^t)$$

Note that M can be different for each object depending on when the object was released in the sea. In our case it is enough to have two positions in the past for each object; that is, $t - 1$ and $t - 2$:

$$\begin{pmatrix} b_1^t & b_1^{t-1} & b_1^{t-2} \\ b_2^t & b_2^{t-1} & b_2^{t-2} \\ \vdots & \vdots & \vdots \\ b_N^t & b_N^{t-1} & b_N^{t-2} \end{pmatrix}$$

5.1.2 Vessel and fishing information input

We need from the vessel the following inputs:

- Vessel speed average (in knots) when traveling from one object to the following one: v_s
- Initial position of the vessel: $v_{init} = (v_{lat}, v_{lon})$
- Time spent by object f_t (fishing time) with two possibilities (our solution can handle both regardless of the choice of the vessel):
  - The same fishing time for all the objects: $f_{t,1} = f_{t,2} = \ldots = f_{t,N}$
  - Different fishing time for each object: $f_{t,1} \neq f_{t,2} \neq \ldots \neq f_{t,N}$

5.1.3 Objects prediction

Once we have the inputs, the first step is to predict the next position of each object using Newton's motion equation. Accordingly, the estimation of the R next positions would be:

$$\begin{aligned} \hat{b}_{t+1} &= b_t + v_t\Delta_t + \frac{1}{2}a_t\Delta_t^2 \\ \hat{b}_{t+2} &= \hat{b}_{t+1} + \hat{v}_{t+1}\Delta_t + \frac{1}{2}\hat{a}_{t+1}\Delta_t^2 \\ &\vdots \\ \hat{b}_{t+R} &= \hat{b}_{t+R-1} + \hat{v}_{t+R-1}\Delta_t + \frac{1}{2}\hat{a}_{t+R-1}\Delta_t^2 \end{aligned}$$

The number of future predictions must be enough to mix with the GA. We will talk about a quantity of R future positions, where typically $R >> N$, depending on v_s and f_t.

After predicting R future positions for each object, the result matrix will be:

$$\begin{pmatrix} b_1^t & \hat{b}_1^{t+1} & \hat{b}_1^{t+2} & \dots & \hat{b}_1^{t+R} \\ b_2^t & \hat{b}_2^{t+1} & \hat{b}_2^{t+2} & \dots & \hat{b}_2^{t+R} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \hat{b}_N^t & \hat{b}_N^{t+1} & \hat{b}_N^{t+2} & \dots & \hat{b}_N^{t+R} \end{pmatrix}$$

Where for example $\hat{b}_N^{t+2}$ is the predicted position of the object b_N at time $t+2$.

5.2 Genetic Algorithm design

As discussed in Section 2, GAs achieve a quasi-optimal solution from a random set of initial solutions called population. In our paper, the specific properties of the GA reflected our concern to minimize the distance traveled by the vessel throughout the recovering process. We summarize them below and provide subsequently in the following subsections a brief report on the algorithm design:

- Population size: 100
- Natural Selection Mechanism: Tournament selection
- Tournament size: 50 couples tournament. Winners are selected
- Crossover type: Greedy crossover. $P = 0,7$
- Mutation type: Simple Mutation between two elements. $P = 0,3$
- Stopping Criteria: 1000 iterations without any fitness improvement

5.2.1 Solution encoding

A solution represents the route that a vessel must follow to recover all the buoys. Each buoy will therefore be a point of the route represented by a number. For instance, (4, 2, 3, 1) means that the vessel has to recover object 4, then object 2, object 3 and finally object 1, which is the end of the route; vessels do not return to their initial point of departure.

5.2.2 Initialization

As is usual in GAs, the initial population was chosen randomly with the aim of covering the entire search space. We particularly used a random set of 100 initial solutions, which perfectly suits the problem we desire to address (Yang, 1997). The fitness of each solution is measured as the total distance traveled by the vessel to recover all the buoys.

5.2.3 Selection

Once the fitness of each random solution has been calculated, the GA works to select a sub-set of routes that becomes the parents of the next generation. We have used here the well-known Tournament Selection Method (TSM) as a selection procedure due to its robustness and simplicity to adjust the genetic pressure, which determines the convergence rate of the GA. Firstly, the TSM chooses a number of couples (tournament size) randomly from the population. We use a 2-Tournament for the mating selection, which entails that we initially selected randomly 50 pairs of routes. Then, each pair competes with each other. The one with the best fitness wins the tournament and becomes a parent for the next generation, named offspring. This selection pressure drives the GA to improve the population fitness through successive generations. In order to do so, nevertheless, the algorithm needs a probability for crossover and mutation. Following standard practices with GAs, the crossover probability has been set at p=0,7; whereas the mutation probability, p', equals 0,3 (p+p' = 1).

5.2.4 Crossover

Crossover is a method where the offspring inherits the characteristics from their parents. We chose the Greedy Crossover Method (Yang, 1997) to address the specific characteristics of our problem: the vessel needs to recover all the buoys only once, and we cannot remove any of them or add others. Thus, given two parent routes R1 and R2, the first offspring is built following these rules: we start in a random buoy b, and then check if the edge leading to b or from b is used in both R1 and R2. If this happens, then the common buoy b is chosen. Otherwise the b's right edge is compared in R1 and R2, so the shorter one is chosen unless it is repeated and it introduces a cycle. In this case, the longer path is chosen. The second offspring is built in a similar way but comparing the b's two left side edges instead of the right ones. In order to implement this method, we need to calculate the distance between some buoys to compare the edges and to determine how the offspring is created. This offspring inherits different characteristics from both parents, ensuring that all the buoys of the route are chosen only once.

5.2.5 Mutation

Mutation is used to preserve and introduce the genetic diversity, so it prevents the algorithm to avoid a local minimum when the population is too similar among them. There is always a mutation probability associated to the mutation operator, which as noted above, we fixed at a standard level of 0,3. There are different mutation types; from the simplest where only one chromosome is mutated (bit string mutation) to more complex approximations (Flip bit, Boundary, Gaussian, etc.). Here we use the simplest one, where two elements of the route are exchanged randomly, since it is sufficient to maintain the genetic diversity of our population and ensures proper convergence of the algorithm. For example, if we apply mutation in the route R1= (2, 4, 1, 3, 5) over the

elements 2 and 3 (first and fourth position of the vector), the resultant offspring will be R1'= (3, 4, 1, 2, 5).

5.2.6 Stopping Criteria

So once we have chosen the 50 parents from the initial population, we provoke crossover or mutation. In both cases we will generate two descendants from each of the 50 parents. If the crossover operator is selected, we choose randomly other parent from the other remaining 49 parents, and later apply the Greedy Crossover technique in order to get the two descendants. By contrast, if the mutation operator is selected, we will mutate two genes (buoys) of the parent route. So if we desire to generate two descendants, we need to perform the mutation twice for each parent. Once we repeat this process for all 50 parents, the offspring will double to reach again 100 —improved— solutions. This process finishes when the loop of steps achieves 1000 iterations without any fitness improvement (stopping criteria). The quasi-optimal route is thus obtained reflecting the shortest distance to recover all the buoys from the vessel's initial position.

5.3 GATP final solution: implementing the GA with Trajectory Prediction

Based on the inputs and techniques we have showed previously, this subsection describes how the GA can work together with the prediction technique (in this case Newton's motion equation) to improve the route when targets are constantly moving. The solution, named GATP (Genetic Algorithm based in Trajectory Prediction), will evolve from scratch to a route where the vessel anticipates the future movement of the FADs.

In order to calculate the final route, we will use the predicted positions. Figure 3 shows the block diagram of our GATP solution, whereas figure 4 represents in detail how our method calculates the fitness of each route.

We show below the steps to solve the problem (figure 3):

1. Calculation of the R future positions of each object: $(\hat{b}^{t+1}, \hat{b}^{t+2}, \ldots, \hat{b}^{t+R})$.
2. Random solutions are calculated as follows:
   Being $(b_1, b_2, \ldots, b_N)$ the objects to recover, we will select random solutions to have the first generation of solutions to the problem. Each route is a sorted list of the objects: $(r_1, r_2, \ldots, r_N)$, where each r can be whatever object to recover $(b_1, \ldots, b_N)$.
3. Calculation of the fitness of each route:
   (a) $[t = 0]$ and $[d = 0]$. Time and distance equal to zero.
   (b) v_p = initial vessel position.

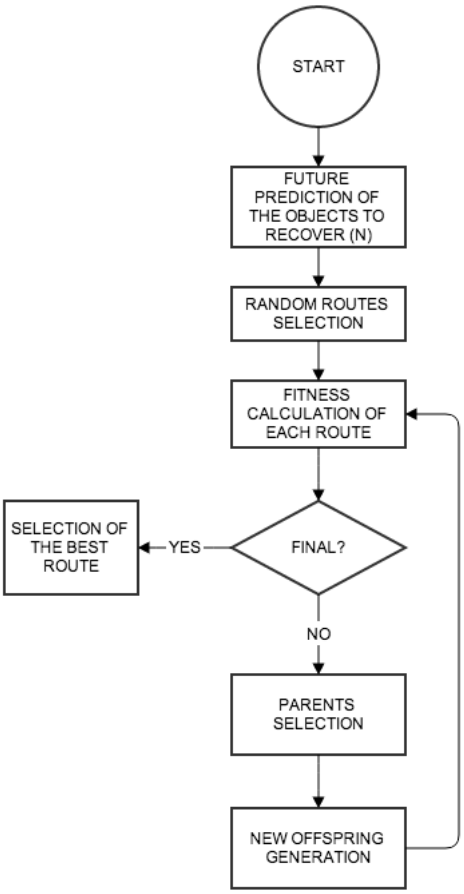

Figure 3: GATP Block Diagram

(c) From $i = 1$ to N

i. Calculation of the time needed by the vessel to go from the current position (v_p) to the next object (r_i: next object in the selected route). This time it will be stored as b_i. Also, the fishing time of this object will be taken into account as f_i.
The position of the object r_i will be $\hat{r}_i^t \equiv$ predicted position of that object i at time t.

ii. Calculation of the time spent to recover and fish in the object r_i

$$t_i = b_i + f_i$$

iii. Vessel position (v_p) is updated to the position of the last recovered object r_i at time t.

iv. Calculation of the distance traveled by the vessel to recover the object i

$$d_i = v_s \cdot b_i$$

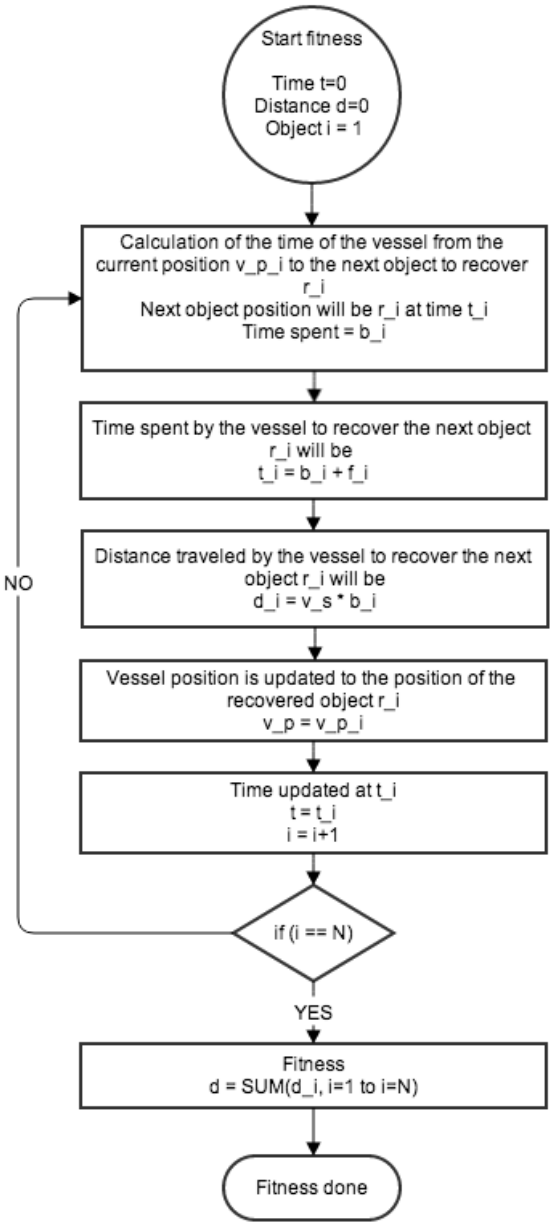

Figure 4: GATP Fitness calculation

v. Current time t is updated to $t = \sum_{j=1}^{i} t_j \equiv$ current time spent.

vi. Total distance traveled:

$$d = \sum_{j=1}^{i} d_j$$

(d) Fitness of the route:

$$d = \sum_{i=1}^{N} d_i$$

4. Check if the termination condition has been reached. In this case the result is the best route; if not, go to step 5.

5. Parents selection.

6. New offspring generation (crossover and mutation).

7. Go to step 3.

Note that we ignore the movement of the FAD in the first movement of the vessel, which means that we ignore the movement of the FAD when the vessel is traveling to recover it. This only happens, however, for the first object. The rationale of this mathematical simplification is to avoid the calculation of the collision vector from the vessel to the object when it is moving (alternatively, the only challenge has to do with the time calculation of the algorithm). This evolving process makes the routes selected in each generation converge on the route that minimizes the real distance from the vessel to all the FADs. Algorithm 1 shows the pseudocode of the GATP solution.

initialization;
while $i < N$ **do**
    calculates future positions of $object_i$;
    $i = i + 1$;
end
random routes selection;
i = 0;
for $i \leftarrow 1$ **to** N **do**
    calculation of time from current vessel position to $object_i \rightarrow b_i$;
    calculation of time spent by the vessel to recover the next $object_i \rightarrow t_i$;
    calculation of distance traveled by the vessel to recover next $object_i \rightarrow d_i$;
    update vessel position: $v_p = v_{p_i}$;
    update time spent: t = $\sum_{j=1}^{i} t_j$;
    $i = i + 1$;
end
fitness calculation: d = $\sum_{i=1}^{N} d_i$;
if *stop condition* **then**
    final route = best fitness route calculated;
    exit;
else
    parents selection from routes;
    new offspring generation;
    $i = 0$;
    go back to the for section;
end

Algorithm 1: GATP algorithm

Figure 5 shows the rationale of the GATP solution and how the route is calculated from the initial position of the vessel (represented by a square) to each object, considering each trajectory is time dependent. We can see that the first vector goes directly where the first buoy is; however, the second finishes where the buoy is expected to be at time t_{i+1}. The same procedure holds for the rest of the buoys.

The most significant difference between our GATP method and the GA-TSP approach is the restriction used to calculate the costs of traveling from one point to the other. The GA-TSP method does not use any prediction technique and it is based in the static assumption of the objects. For this reason the fitness function is different in each case. The costs are totally dependent on the distance traveled by the vessel before measuring where the next object will be at time t. The distance between the vessel (previous object at time t) and the next object at time t is subsequently calculated. This fitness function makes our GATP solution evolve towards the route that minimizes the distance traveled by a vessel.

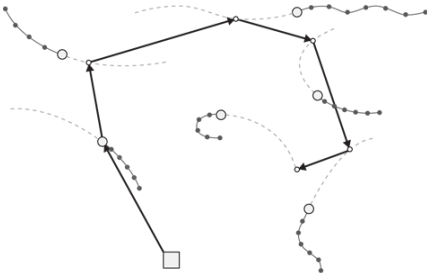

Figure 5: Using GA based combining prediction methods

Figure 6 shows our solution with an example of the implementation of the three methods.

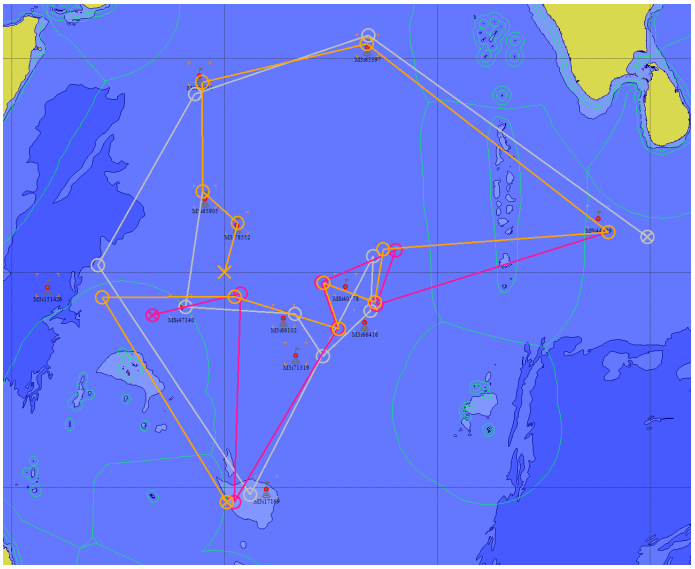

Figure 6: NN, GA-TSP and GATP: A graphic example for 12 FADs

Each route has a different color when they deviate from the rest:

- NN: grey color
- GA-TSP: pink color
- GATP: orange color

The FADs are represented by the dots in red. In this example the orange cross represents the starting position of the vessel, whereas each circle reflects an FAD recovery. The final point of each route (last FAD recovered) is marked with a cross inside a circle.

The graphic shows that the circles are close for the first FADs recovered, but as the time goes by, the distance increases. Particularly, the three methods start similarly: all start picking the two FADs to the north of the orange cross (vessel initial position). This is why we can only see one trajectory in orange. Then, the NN solution leads the vessel to a different route; we can see how the grey line goes south, whereas the orange line keeps heading north. The GA-TSP and GATP trajectories thus continue the route in the same order during the recovery of 3 more FADs; one to the north, the following to the east and the third to the south-east. After picking the fifth FAD, however, the routes deviate and the pink trajectory reflecting GA-TSP appears.

6 Results and discussion

In this section we discuss the improvement achieved by addressing DTSP with the method proposed in this paper: GAs based on Trajectory Prediction (GATP). Initially, and just with informative purposes, we compare the Nearest Neighbor (NN) strategy, which is the method normally used by tuna vessels, with TSP solved by GAs (GA-TSP). This second method consists on applying a simple GA to the TSP problem. Then, we compare the performance of our GATP method with both NN and GA-TSP. It is worth recalling that we use real data offered by tuna companies for the last quarter of 2013. We have made the following assumptions:

- Average vessel speed = 12 knots
- Recovery time = 3 hours for all the objects
- Number of objects to recover = 6, 9 and 12
- Buoys have different speed, which can go from 0.2 knots up to 2 knots
- Distance between buoys is also variable, it can go from 100 nm up to $1,500$ nm

The results (average, standard deviation and the improvement percentage achieved between each two methods) are shown in Table 3. We can observe that our GATP method is always better than the NN and GA-TSP for recovering 6, 9 and 12 buoys (normal working range for these vessels).

These results are supported statistically. The comparisons have been tested through a Repeated Measures ANOVA, given that the same subjects (vessels) are used for each treatment (method). We thus find significant differences among methods and among the interaction of methods with a different number of FADs (Sig. $= 0.000 < 0.05$) (Table 4). Results are consistent since the most frequent multivariate tests used in ANOVA (Phillai's trace, Wilks' Lambda, etc.) show a very high significance. If we now deepen into which of the means for the three methods are significantly different from the others, the pairwise comparisons support our descriptive analysis: average results by GATP are

Table 3: Results comparison

			Total distance traveled (nautical miles)			Improvement Comparison		
Experiment	**Nº buoys**		**NN**	**GA-TSP**	**GATP**	**GA-TSP vs NN**	**GATP vs NN**	**GATP vs GA-TSP**
1	6 buoys	$\bar{x}$	1735.1	1645.0	1615.6	4.4%	6.2%	1.9%
		σ	566.6	498.8	501.2	5.6%	5.6%	2.2%
2	9 buoys	$\bar{x}$	4277.0	4185.4	3953.6	1.7%	7.3%	5.5%
		σ	807.3	726.8	702.0	7.5%	4.7%	3.5%
3	12 buoys	$\bar{x}$	4069.4	3817.0	3734.1	5.6%	7.5%	2.1%
		σ	725.8	559.6	556.3	5.6%	8.3%	4.4%

Table 4: Repeated Measures ANOVA: multivariate tests

Effect		**Value**	**F**	**Hipothesis df**	**Error df**	**Sig.**
Method	Pillaiís Trace	0.552	34.470	2.000	56.000	0.000
	Wilksí Lambda	0.448	34.470	2.000	56.000	0.000
	Hotellingís Trace	1.231	34.470	2.000	56.000	0.000
	Royís Largest Root	1.231	34.470	2.000	56.000	0.000
Method · Nº of Buoys	Pillaiís Trace	0.371	6.482	4.000	114.000	0.000
	Wilksí Lambda	0.653	6.647	4.000	112.000	0.000
	Hotellingís Trace	0.495	6.805	4.000	110.000	0.000
	Royís Largest Root	0.405	11.556	2.000	57.000	0.000

Table 5: Repeated Measures ANOVA: Pairwise Comparison

Pairwise Comparisons						
					95% Confidence Interval for Differences	
(I) Method	**(J) Method**	**Mean Difference (I-J)**	**Std. Error**	**Sig.**	**Lower Bound**	**Upper Bound**
NN	GA-TSP	144.711	32.937	0.000	63.467	225.956
	GATP	259.415	35.151	0.000	172.709	346.121
GA-TSP	NN	-144.711	32.937	0.000	-225.956	-63.467
	GATP	114.703	17.826	0.000	70.733	158.674
GATP	NN	-259.415	35.151	0.000	-346.121	-172.709
	GA-TSP	-114.703	17.826	0.000	-158.674	-70.733

statistically different (distances are lower) from those obtained by NN and GA-TSP for 6, 9 and 12 FADs (Table 5).

In short, GATP yields better results than other common optimizing strategies when addressing routes for moving targets in the short term. The sophistication of the prediction method, however, must be adapted to the specific characteristics of the exercise. For instance, improving forecasting accuracy for FADs trajectories in the long term requires a prediction method that considers the chaotic nature of its currents and internalizes Eddy effects, temperature or altimetry. As mentioned above, however, this is not a concern in the specific case of tuna fishing because a typical vessel spends normally less than five days fishing and recovers two sometimes 3- buoys each day.

Finally, although the execution time of the GATP algorithm is an important variable indeed, tuna vessels do not require a real time computation because it takes at least three hours (normally between 8 and 12 hours) to fish and recover each buoy. The skipper will therefore run the solution to plan a week of work, finding out which is the best buoy to start with and then run the algorithm with information updates on the buoys position to keep on. Even if users had to wait several minutes for the algorithm execution in each buoy, it would not accordingly represent a problem. Our experiments with real data show, anyhow, that the execution time for the 3 methods is much lower:

- NN: 10 - 20 milliseconds for the recovery of 6 up to 12 buoys
- GA: 0,5 - 1,2 seconds for the recovery of 6 up to 12 buoys
- GATP: 0,6 - 2 seconds for the recovery of 6 up to 12 buoys

7 Conclusions

We have addressed the Traveling Salesman Problem with GA assuming that targets change their position with time. Our contribution is a new way of solving the dynamic route optimization problem using a simple prediction method that, combined with the power of GAs, makes the implemented algorithm evolve towards the near-optimal route.

The comparative analysis between GATP and other commonly used methods like NN or GA-TSP reveals the benefits of internalizing predictive methods within GAs. However, given the chaotic nature of ocean currents and regardless of the sophistication of the forecasting method, we can expect that GATP adds less value if long term predictions were needed.

In practical terms, the GATP algorithm's execution time allows new, better routes to be recalculated easily when the FADs new positions are updated, also showing a better real time route possibility where it exists. We could accordingly conclude that GATP allows tuna vessels and any other agent pursuing moving targets in the short term -like military airplanes- to minimize the distance traveled, which would impact directly on such relevant variables as the time employed, fuel consumption or CO_2 emissions to the

atmosphere. It is important to emphasize here that, for a given speed, the distance saved is equivalent to fuel savings. This is extremely relevant not only to reduce costs but also to increase the storage space.

Furthermore, the development of more sustainable fishing with FADs may benefit from further research. To begin with, the algorithm is totally flexible and open to future improvements adding new restrictions, such as prioritizing the recovery of some targets that have more fish beneath them (using buoys with echo-sounder information), working with time-windows for the recovery of FADs (the vessels can't get fish during the night, for example), implementing a multiple vessel FAD recovery strategy or finding the optimal vessel speed in order to save more fuel. From a more general perspective and beyond the specific tools employed in this paper, our results also reflect the value of mixing an heuristic method with a predictive technique, regardless of the specific choice in any of the two. A quasi-optimal solution could be consequently found using other heuristic methods if they were combined with prediction techniques in a proper way.

Finally, from a theoretical point of view, it is worth noting that GATP is a generalization of the classic methods to solve the TSP with GA. When targets are not moving the predicted next position by GATP will be the same offered by GA-TSP, so this solution will evolve as classic methods for GA-TSP do. However, when targets start moving, the proposed solution is different because GATP evolves in order to continue optimizing the total route traveled, assuming the future movement of each object and therefore achieving better results (depending on the prediction period). Our solution can therefore be used in a generic way; for static, dynamic and mixed scenarios, being a more flexible and more adaptable solution to estimate near-optimal routes in general.

Acknowledgements

The authors wish to thank Marine Instruments for the buoys data. We also thank Iago Paz, Gabriel Rosón, Marcos Álvarez, Diego Piñeiro, Francisco Pino, Bruno Lema and Daniel Mosquera for several useful suggestions. Regarding financial support, this research benefited from the "Programme for the Consolidation and Structuring of Competitive Research Units" co-funded by the European Regional Development Fund (10SEC300036PR), as well as from the support of the Spanish Ministry of Economics and Competitiveness through project ECO2013-45706-R.

References

Applegate D, Bixby R, Chvatal V, Cook WJ (2007) The traveling salesman problem: a computational study. Princeton University Press

Avner P, Bruls T, Poras I, Eley L, Gas S, Ruiz P, Wiles M, Sousa-Nunes R, Kettleborough R, Rana A, et al (2001) A radiation hybrid transcript map of the mouse genome. Nature genetics 29(2):194–200

Bach P, Dagorn L, Josse E, Bard F, Abbes R, Bertrand A, Misselis C (1998) Experimental research and fish aggregating devices (fads) in french polynesia. SPC Fish Aggregating Device Information Bulletin 3:3–18

Baker BM, Ayechew M (2003) A genetic algorithm for the vehicle routing problem. Computers & Operations Research 30(5):787–800

Besse PC, Cardot H, Stephenson DB (2000) Autoregressive forecasting of some functional climatic variations. Scandinavian Journal of Statistics 27(4):673–687

Bjarnadóttir ÁS (2004) Solving the vehicle routing problem with genetic algorithms. PhD Thesis, Technical University of Denmark, Denmark

Bland RG, Shallcross DF (1989) Large travelling salesman problems arising from experiments in x-ray crystallography: a preliminary report on computation. Operations Research Letters 8(3):125–128

Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput Surv 35(3):268–308

Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: From natural to artificial systems. Oxford University Press, Oxford, UK

Casdagli M (1989) Nonlinear prediction of chaotic time series. Physica D: Nonlinear Phenomena 35(3):335–356

Castro JJ, Santiago JA, Santana-Ortega AT (2001) A general theory on fish aggregation to floating objects: an alternative to the meeting point hypothesis. Reviews in fish biology and fisheries 11(3):255–277

Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. Journal of the operations research society of America pp 393–410

Donald D (2010) Traveling Salesman Problem, Theory and Applications. Intech

Duchenne É, Laporte G, Semet F (2007) The undirected m-peripatetic salesman problem: Polyhedral results and new algorithms. Operations research 55(5):949–965

Dumas Y, Desrosiers J, Gelinas E, Solomon MM (1995) An optimal algorithm for the traveling salesman problem with time windows. Operations research 43(2):367–371

Eyckelhof CJ, Snoek M (2002) Ant systems for a dynamic tsp: Ants caught in a traffic jam. In Ant Algorithms pp 88–99

Fiechter CN (1994) A parallel tabu search algorithm for large traveling salesman problems. Discrete Appl Math 51(3):243–267

Garcia-Najera A, Bullinaria JA (2011) An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. Computers & Operations Research 38(1):287–300

Garrido P, Riff MC (2010) Dvrp: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. Journal of Heuristics 16(6):795, 834

Golden BL, Levy L, Vohra R (1987) The orienteering problem. Naval Research Logistics (NRL) 34(3):307–318

Grötschel M, Padberg MW (1985) Polyhedral theory. The traveling salesman problem pp 251–305

Grötschel M, Jünger M, Reinelt G (1991) Optimal control of plotting and drilling machines: a case study. Mathematical Methods of Operations Research 35(1):61–84

Guntsch M, Middendorf M, Schmeck H (2001) An ant colony optimization approach to dynamic tsp

Gutin G, Punnen AP (2002) The traveling salesman problem and its variations, vol 12. Springer

Hajjam A, Créput JC, Koukam A (2013) From the tsp to the dynamic vrp: An application of neural networks in population based metaheuristic. In Metaheuristics for Dynamic Optimization pp 309–339

Helvig CS, Robins G, Zelikovsky A (2003) The moving-target traveling salesman problem. Journal of Algorithms 49(1):153–174

Holland J (1975) Adaptation in natural and artificial systems, university of michigan press. Ann Arbor, MI 1(97):5

Huang ZC, Hu XL, Chen SD (2001) Dynamic traveling salesman problem based on evolutionary computation. In: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, IEEE, vol 2, pp 1283–1288

Jeong CS, Kim MH (1991) Paper: Fast parallel simulated annealing for traveling salesman problem on simd machines with linear interconnections. Parallel Comput 17(2-3):221–228

Jih WR, Hsu Y (2004) A family competition genetic algorithm for the pickup and delivery problems with time window. Bulletin of the College of Engineering, National Taiwan University 90:121–130

Karlaftis MG, Kepaptsoglou K, Sambracos E (2009) Containership routing with time deadlines and simultaneous deliveries and pick-ups. Transportation Research Part E: Logistics and Transportation Review 45(1):210–221

Konak A, Coit DW, Smith AE (2006) Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety 91(9):992–1007

Lawler EL, Lenstra JK, Kan R, Shmoys DB (1985) The traveling salesman problem: a guided tour of combinatorial optimization, vol 3. Wiley New York

Li C, Yang M, Kang L (2006) A new approach to solving dynamic traveling salesman problems. Simulated Evolution and Learning pp 236–243

Liu L, Wang D, Yang S (2009) An immune system based genetic algorithm using permutation-based dualism for dynamic traveling salesman problems. In Applications of Evolutionary Computing pp 725–734

Marcellino M, Stock JH, Watson MW (2006) A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series. Journal of Econometrics 135(1):499–526

Moon C, Kim J, Hur S (2002) Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. Computers & Industrial Engineering 43(12):331–349

Moreno G, Dagorn L, Sancho G, Itano D (2007) Fish behaviour from fishers' knowledge: the case study of tropical tuna around drifting fish aggregating devices (dfads). Canadian Journal of Fisheries and Aquatic Sciences 64(11):1517–1528

Özgökmen TM, Griffa A, Mariano AJ, Piterbarg LI (2000) On the predictability of lagrangian trajectories in the ocean. Journal of Atmospheric and Oceanic Technology 17(3):366–383

Pantrigo JJ, Duarte A (2013) Low-level hybdridization of scatter search and particle filter for dynamic tsp solving. Metaheuristic for Dynamic Optimization pp 291–308

Potvin JY (1996) Genetic algorithms for the traveling salesman problem. Annals of Operations Research 63(3):337–370

Psaraftis HN (1988) Dynamic vehicle routing problems. Vehicle Routing: Methods and Studies (16):223–248

Pérez J (2004) A hybrid approach for a constraint routing problem. In: Proceedings of the Fourth International Conference on Hybrid Intelligent Systems, Kitakyushu, Japan

Reinelt G (1994) The traveling salesman: computational solutions for TSP applications. Springer-Verlag

Ruan Q, Zhang Z, Miao L, Shen H (2013) A hybrid approach for the vehicle routing problem with three-dimensional loading constraints. Computers & Operations Research 40(6):1579–1589

Simões A, Costa E (2011) Chc-based algorithms for the dynamic traveling salesman problem. Applications of Evolutionary Computation pp 354–363

Smith GC, Smith S (2002) An enhanced genetic algorithm for automated assembly planning. Robotics and Computer-Integrated Manufacturing 18(5):355–364

Tyedmers P, Parker R (2012) Fuel consumption and greenhouse gas emissions from global tuna fisheries: A preliminary assessment. Tech. rep., ISSF Technical Report 2012-03. International Seafood Sustainability Foundation, McLean, Virginia, USA

Wang X, Regan AC (2002) Local truckload pickup and delivery with hard time window constraints. Transportation Research Part B: Methodological 36(2):97–112

Winter G, Periaux J, Galan M, Cuesta P (1996) Genetic algorithms in engineering and computer science. John Wiley & Sons, Inc.

Yan XS, Liu HM, Yan J, Wu QH (2007) A fast evolutionary algorithm for traveling salesman problem. In Proceedings of the 3rd International Conference on Natural Computation, 2007 ICNC 2007 4:85–90

Yang R (1997) Solving large travelling salesman problems with small populations. Second International Conference on Genetic Algorithms in Engineering Systems pp 157–162

Younes A, Basir O, Calamai P (2003) A benchmark generator for dynamic optimization. In Proceedings of the 3rd WSEAS International Conference on Soft Computing, Optimization, Simulation & Manufacturing Systems, Malta

Zhang GP, Qi M (2005) Neural network forecasting for seasonal and trend time series. European journal of operational research 160(2):501–514

Zhou A, Kang L, Yan Z (2003) Solving dynamic tsp with evolutionary approach in real time. In: Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, IEEE, vol 2, pp 951–957