IET Journals
The Institution of Engineering and Technology

# Genetic algorithm-based multiple moving target reaching using a fleet of sailboats

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

*VIEL Christophe*[1], *VAULTIER Ulysse*[1], *WAN Jian*[1], *JAULIN Luc*[2]

[1] *School of Engineering, Computing and Mathematics, University of Plymouth, Plymouth, Devon, UK, e-mail: jian.wan@plymouth.ac.uk*
[2] *Lab STICC, ENSTA Bretagne, Brest, France, e-mail: lucjaulin@gmail.com*

**Abstract:** This paper addresses the problem of Dynamic Traveling Salesman Problem (DTSP) for a multi-agent system using a fleet of sailboats. A genetic algorithm (GA) is proposed which attributes to each agent a varying number of targets to be collected. GA allows to obtain a suboptimal solution in the shortest time possible. Moreover, this paper adapts it to the specific problem involving a fleet of sailboats, which is a challenging task with comparison to Unmanned Surface Vehicle (USV) or motorized vehicles in terms of the propulsion. Therein motors can be flexibly controlled while sailboat movements are constrained by available wind direction and speed. Thus the method takes into account wind conditions at various locations of the sailboat. Simulation results demonstrate the effectiveness of the proposed approach.

**keyword:** Genetic algorithm, multi-agent system, autonomous sailboat.

## 1 Introduction

For the past decades, the development of marine robotics has been stimulated by the needs of oceanography and ocean exploitation. Considerable progresses have been made in developing and using various platforms of autonomous marine systems.

Despite of their limited speed and wind dependence, autonomous sailboats have the potential for marine science as their use of renewable solar and wind energies is fit for long-duration missions. Other potential applications such as surveillance and mapping generate further interest in them. Autonomous sailing is however faced with two inherent difficulties: the sailboat velocity is indeed uncontrollable at certain orientations such as upwind, and there is also an absence of accurate control for the boat velocity in many cases; the maximal velocity of sailboats depends entirely on the wind speed. Thus, while the position and the speed of surface vehicles can be controlled by motors, the limitation on the speed of sailboats and the wind conditions must be taken into account. Many low-level and high-level control systems designed for sailboats can be found in the literature [13, 14, 18, 20, 21, 24, 25] to follow line, avoid obstacles, path tracing or reach a target point. This paper focuses on the problem of a fleet of sailboats to collect data from moving targets (buoys, fish, boats), as an application of the Dynamic Traveling Salesman Problem (DTSP).

The Travelling Salesman Problem (TSP) probably represents the most studied optimization problem. However, the problem involving moving targets, or Dynamic TSP (DTSP), is still an open issue. To solve DTSP, several methods exist using for example Ant Colony Optimization (ACO) [1, 2, 12], Genetic Algorithm (GA) [3, 9], Integer Linear Program (ILP) solvers [22, 23] or Self-Organizing Map (SOM) [7, 26]. ACO obtains good performance for DTSP where cities appear and disappear, but are completely inappropriate for moving targets. Genetic Algorithms (GA) [3, 9] offer a sub-optimal trajectory with the most reduced time or distance to collect targets, but require a long processing time and there lack of approaches in the literature for attributing targets to a multi-agent system (MAS). ILP solvers [22, 23] and SOM [7, 26] allow to obtain results with less calculation time. However, ILP solvers require specific configurations such as targets moving with zero accelerations and following linear paths while SOM's trajectories for moving targets are not optimal.

In this paper, the GA developed in [3] has been improved by adapting it to a multi-agent algorithm for tracking multiple targets simultaneously, and by adapting this algorithm to the particular problem coupled with sailboat dynamics using a method similar to [4]. It allows to control a fleet of pursuers like in [7, 26] but with an optimal trajectory of the shortest time to collect all targets like in [3]. However, in opposite with [4], this work is adapted to sailboat dynamics and various wind conditions. We focus here on wind influence, but other oceanic data, like waves or surface currents, can be added in a similar way.

The main contribution of this paper is twofold: first, the proposal of a genetic algorithm for collecting moving targets using a multi-agent system and coming back home after retrieving their targets; second, the adaptation of this GA to the particular problem involving a fleet of sailboats to collect buoys by taking into account wind conditions changing with time and space.

The potential application of this method is to use sailboats for picking up buoys previously cast off in sea and to bring them back to the base. For that, the paper is divided into two main parts: the first part presents the genetic algorithm that finds the smaller time to collect all the moving targets, to return to the initial position, and to choose the number of targets for each agent; the second part adapts the proposed algorithm to the sailboat problem. A calculation of interception time and position is proposed, using an approximation of the average sailboat velocity to reduce the processing time in the loop of the GA. This average velocity takes into account several varying parameters such as wind direction and speed.

The rest of the paper is organised as follows. The related work is introduced in Section 2. Problem statement is described in Section 3. Genetic algorithm is exposed in Section 4 with its parent selection, crossover and mutation in Sections 4.4, 4.5 and 4.6, respectively. The method of target attribution is explained in Section 4.3. The adaptation of the GA for a fleet of sailboats is described in Section 5 and 6, where a method to evaluate an average sailboat velocity and target interception position are proposed in Section 6.2 and 6.1. Section 7 presents some simulation results. Section 8 concludes the paper.


© The Institution of Engineering and Technology 2015

## 2 Related work

The Traveling Salesman Problem (TSP) probably represents the most studied optimization problem. It can be presented as a sub-problem in many transportation and logistics applications, for example, finding the shortest path for vehicles or the fastest path to collect data using a robot. TSP can be divided into a multitude of different problems [15, 19]. However, the literature on TSP is still rare with regards to moving targets, like in fishing [3], surveillance [11] or in military applications [5, 23]. These research topics are grouped under the title of "Dynamic Traveling Salesman Problem ", DTSP, with two main versions existing in the literature. The first one consists of inserting or deleting cities into a given problem instance [1, 2, 12] and the second keeps constant the number of cities but changes the distance among them [5, 22, 23], addressing for traffic jams, highways, or the collection of data with moving targets.

In [1, 2, 12], an Ant Colony Optimization (ACO) algorithm was proposed to solve DTSP due to their strong adaptation capabilities. When a new city appears, the knowledge of old optimization is transferred to the new scenario, and so as to reduce the calculation time. The proposed algorithm maintains a good balance between the computation time and the quality of the solution through effective local search algorithms. However, ACO can only be used for the first version of the DTSP, and it is not adapted to move targets, which is the focus of this paper.

Another adapted method for DTSP is the Genetic Algorithm (GA). In [9], a GA-based method is proposed based on the prediction of target trajectories for solving DTSP problems. A genetic algorithm is a meta-heuristic method inspired by the process of natural selection to generate high-quality solutions to the optimization problem. GA starts with initializing the population with randomly generated but feasible solutions. Each solution is then evaluated by using a fitness function describing the desired configuration, for example, the shorter time or distance to reach all the targets. A selection of parents is made to generate children through crossover and mutation from parents. Children become the new population and the reproduction cycle continues. In [9], each target is moving in a direction at a constant velocity from an initial position, and the pursuer starts from the origin at a constant velocity. Two different crossovers are used with the guarantee that if the two parents are valid tours, the child is also a valid tour. A most evolved GA is proposed in [3]. This one evaluates its fitness to reduce the final time of a vessel to collect moving buoys. A simple prediction of future locations of the buoys is evaluated using Newton's motion equations based on historical real data provided by GPS fitted on buoys. Comparisons between the proposed method and Nearest Neighbor (NN) or simple Genetic Algorithms GA-TSP method are provided. It is worthy noting that the assumption made therein is that only one pursuer rather than multiple pursuers is engaged to collect the targets. In [4] the same approach to [3] is discussed with the extension to a multi-agent system for fishing boats.

[5] is one of the first papers that study DTSP directly rather than the adapted TSP to a single or multiple pursuers. It proposes and proves a geometric optimal solution to minimize the time to collect targets moving with a zero acceleration and following linear path passing through the origin. Each pursuer has to catch one target and come back to the origin before starting the next target. A similar configuration is studied in [22, 23]. Here, the DTSP is studied for the application of multiple weapons to multiple target assignments, where the weapons play the role of the salesmen, and targets play the role of the cities to be visited. The specific constraint here is that the problem needs to be solved in a very short time. The algorithm turns out to be a modern ILP solver, allowing to solve instances of relevant size for a weapon-to-target assignment problem in a reasonable short time. The solutions are often globally optimal. However, the chosen constraints make these two problems limited to a specific configuration.

In [7, 26], a SOM-based approach is proposed to control a team of AUVs to visit moving targets in a dynamic 3-D ocean current environment. This method can deal with complicated cases such as the number of AUVs is smaller or larger than the number of targets. Initially, SOM is used to define the shortest path between static targets, making it useful for energy saving. By updating continuously targets assignment to UAVs and taking into account current perturbations on UAVs and targets, the proposed algorithm adapts itself to reach the moving target and to choose a shortest path. Note the principle is to adapt assignments to current behavior without any effort to predict them, thus the trajectories are not optimal and could be improved if a target prediction becomes available. Moreover, adapting it to obtain the shortest time rather than the shortest distance in the presence of moving targets seems rather complicated.

## 3 Problem statement

In this paper, the problem can be summarized with the following assumptions:

- $N$ sailboats, also called agent, are deployed to collect $N_t$ moving targets with $N \leq N_t$. Each agent is numbered from 1 to $N$ and each target is numbered from 1 to $N_t$;
- Targets must be collected with the least time, $i.e.$ a target is assigned to a particular sailboat if it is able to be at the target at the earliest possible time, even if it needs to travel a longer distance compared to other nearby sailboats;
- Targets need to be recovered once by only one boat;
- Targets cannot be added or removed during the mission;
- Agents start in an initial "home " location and must come back to the "home " to complete its mission after they have reached their targets;
- The number of targets attributed to sailboats is not necessarily equal, but each agent collects at least one target.

Each target is represented by a number. For instance, $[4, 2, 3, 1]$ means that the sailboat has to recover target 4, then target 2, target 3 and finally target 1, before coming back to the home position.

## 4 Genetic algorithm

A genetic algorithm achieves a quasi-optimal solution from a random set of initial feasible solutions called population. From the initial generating population, each solution is then evaluated by using a fitness function describing the desired configuration. In our case it is the shortest time to reach all the moving targets. For that, a prediction for the movement of each target and a calculation for the interception time are required. Once the fitness is evaluated, a selection of the parents is made by organizing a competition between them. Children are generated from crossovers and mutations among parents, and then they become the new population. The reproduction cycle continues until a stopping criteria is reached. The best solution of each iteration is kept in memory to make the final choice.

In this paper, the GA has been built to minimize the distance to be travelled by the agent during the collecting process. The following subsections described the algorithm design. Figure 1 shows the flowchart of the algorithm. Note this algorithm can deal with all kinds of vehicles because time to perform manoeuvres are supposed to be negligible compare to time to reach the target (more details on these hypothesis will be provided in Section 5.1).

### *4.1 Initialization*

As usual in genetic algorithms, the initial population is chosen randomly. Let define the population matrix $L_{pop}$ of size


© The Institution of Engineering and Technology 2015

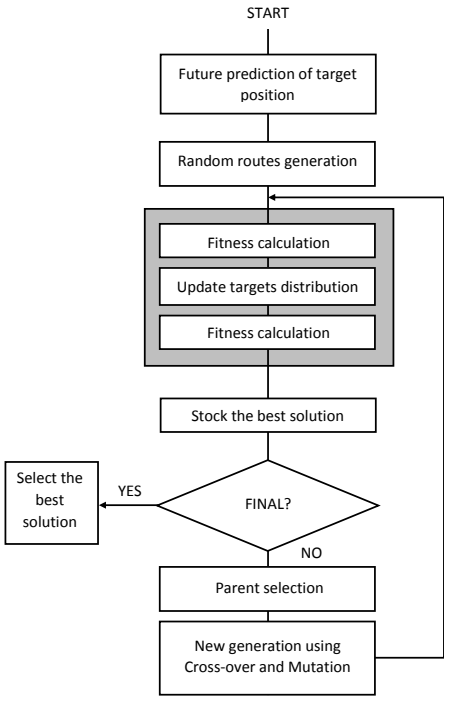



**Fig. 1**: Genetic algorithm flowchart

$N_{pop} \times N_t$ where $N_{pop}$ is the size of the population, and $L_{pop}(i, :)$ is the solution associated to the subject $i$ inside the population. A subject $i$ corresponds to the solution $L_{pop}(i, :)$. A solution associated to subject $i$ is the order in which targets are to be collected by agents. Let also define the randomly split matrix $L_{split}$ of size $N_{pop} \times (N-1)$ associated to the population $L_{pop}$, with $L_{split}(i, :)$ is the split vector associated to the subject $i$. The split list $L_{split}(i, :)$ assigns the $N_t$ targets inside $L_{pop}(i, :)$ to the $N$ agents such that

- Agent 1 has to collect targets $[L_{pop}(i, 1), \ldots, L_{pop}(i, L_{split}(i, 1) - 1)]$,
- Agent $j$ for $j \in [2, \ldots, N-1]$ has to collect targets $[L_{pop}(i, L_{split}(i, j)), \ldots, L_{pop}(i, L_{split}(i, j+1) - 1)]$,
- Agent $N$ has to collect targets $[L_{pop}(i, L_{split}(i, N)), \ldots, L_{pop}(i, \text{end})]$,
- $L_{split}(i, a) < L_{split}(i, b)$ for all $(a, b) \in [1, N-1]^2$ such $a < b$.

Example: Suppose $N = 3$ agents $\{a, b, c\}$, and solution $i$ is associated to the list $L_{pop}(i, :) = [2, 4, 1, 3, 7, 6, 5]$ with the split list $L_{split}(i, :) = [3, 5]$. Thus Agent $a$ pursues $[2, 4]$, agent $b$ pursues $[1, 3]$ and agent $c$ pursues $[7, 6, 5]$.

Let $\mathcal{N}_k$ be the number of targets collected by Agent $k$ such as

- $\mathcal{N}_1 = L_{split}(i, 1) - 1$ ,
- For $j \in [2, \ldots, N-1]$, $\mathcal{N}_j = L_{split}(i, j+1) - L_{split}(i, j)$,
- $\mathcal{N}_N = N + 1 - L_{split}(i, N)$.

The fitness of each solution is evaluated as described in the Section 4.2.

### 4.2 Fitness evaluation and fitness list

For each subject $i$ in the population, let $f_i$ be the fitness of the subject $i$ and $F_i \in \mathbb{R}^{N_t}$ be the list of fitness discrepancy. The fitness $f_i$ of the solution corresponds to the time performed by the slowest agent in the fleet to collect all its targets and to return back home. Let define also the following sum of fitness $\bar{f}_{i,k}$ and $\underline{f}_{i,k}$. $\bar{f}_{i,k}$ is the fitness of an Agent $k$ in the solution $i$: it is to be used for fitness calculation. $\underline{f}_{i,k}$ is the fitness of an Agent $k$ in the solution $i$ when the agent reaches the second last target, and will be used in Section 4.3. One has

$$\bar{f}_{i,k} = \sum_{j=j_{\min,k}}^{j_{\max,k}} F_i(j), \quad \underline{f}_{i,k} = \sum_{j=j_{\min,k}}^{j_{\max,k}-1} F_i(j)$$

where $j_{\min,k}$ and $j_{\max,k}$ are respectively the index of the first and last targets collected by agent $k$ inside list $F_i$. Thus, one has $j_{\max,k} = L_{split}(i, k)$ if $k < N$, $j_{\max,k} = N_t$ else, $j_{\min,k} = L_{split}(i, k-1)$ if $k > 1$, $j_{\min,k} = 1$ else. The fitness evaluation can be expressed with the following Algorithm 1.

Algorithm 1: fitness evaluation

**Require:** Initially, let $F_i = 0_{N_t \times 1}$ be the current fitness, $t = 0$ be the current time instant, $n = 1$ be the current agent index and $p_n = [0, 0]$ be its initial position.
**for** $j = 1 : N_t$ **do**
  **if** $j = L_{split}(i, n)$, the next agent is selected: **then**
    take $t = 0$, $t^{old} = 0$ and $n = \min(n+1, N-1)$.
  **end if**
  ∘ Take the target $k = L_{pop}(i, j)$ and evaluate its current position $p_k^t(t)$ (using for example Section 5.3).
  ∘ Using $p_k^t(t)$ and current Agent $n$ position $p_n(t)$, evaluate their interception point $\bar{p}(k, n)$ and interception time $t_{k,n} > t$ (using for example Section 6.1).
  ∘ Update Agent $n$ position, fitness and time : take $t^{old} = t$, $F_i(j) = t_{k,n} - t^{old}$, $t = t_{k,n}$, and $p_n(t_{k,n}) = \bar{p}(k, n)$.
  **if** $j + 1 = L_{split}(i, n)$ or $j = N_t$, **then**
    ∘ Agent $n$ has reached all its target. Evaluate time $t_{0,n}$ to come back to home.
    ∘ Take $F_i(j) = t_{0,n} - t^{old}$.
  **end if**
  ∘ Fitness $\bar{f}_{i,k}$ $\forall k \in [1, \ldots, N]$ are times required by each Agent $k$ to collect all its target and back home. The fitness $f_i$ of the solution corresponds to the longest time performed in the fleet, thus $f_i = \max_{k \in [1, \ldots, N]} (\bar{f}_{i,k})$.
**end for**

Let define the stock fitness matrix $Lf$ of size $N_{pop} \times 1$ such $Lf(i) = f_i$ and the stock matrix of increasing fitness $LF$ of size $N_{pop} \times N_t$ such $LF(i, :) = F_i$. These two matrices are to be used to update the target distribution and to select parents of the new generation.

### 4.3 Update target distribution

To balance the time performed by agents, $L_{split}$ needs to be updated. An equal distribution of the targets is not always a good option because some targets can be collected quickly by one agent while others can be very far from the group requiring one dedicated agent to collect it. The target attribution is chosen to reduce the global fitness $f_i$ by balancing the individual fitness $\bar{f}_{i,k}$. For that, fitness $\bar{f}_{i,k}$ after Agent $k$ coming back home is compared with fitness of other agents $\ell$ reaching their second last target $\underline{f}_{i,\ell}$. If this fitness is lower, one target of a slower agent is attributed to Agent $k$. The second last target is chosen to compare fitness. Remind that each agent needs to collect at least one target.

Since the fitness of an agent must be re-evaluated when its number of targets has been modified. Only one modification by agents is allowed in the following approach. Let $q = 0_{N_t}$ be the binary test variable to manage it. The method is described in Algorithm 2.

If $\exists j \in [1, \ldots, N]$ $q(j) = 1$, the modification has been made on $L_{split}$. A new evaluation of the fitness $f_i$ is required before conducting the parent's selection exposed in Section 4.4.

### 4.4 Parent's selection

Once the fitness of each solution has been calculated, a selection of these solutions is performed to become the parent of the next generation. $N_{parent} = \frac{N_{pop}}{2}$ are generated. For that, a Tournament Selection Method (TSM) is used as the selection procedure. TSM is a robust and simple method for selecting priority solutions with the best fitness, but with a chance to


© The Institution of Engineering and Technology 2015

Algorithm 2: update fitness list

**for** $j = 1 : (N-1)$ **do**
  **for** $k = (j+1) : N$, **do**
    **if** $q(j) = 1$, **then**
      break : a modification has already be made for Agent $j$.
    **else**
      **if** $\left(\bar{f}_{i,j} < \underline{f}_{i,k}\right) \& \left(\mathcal{N}_k > 1\right) \& \left(q(k) = 0\right)$, *i.e.* Agent $k$ has at least two targets, it has not been modified yet and its fitness $\underline{f}_{i,k}$ is larger than the fitness $\bar{f}_{i,j}$ of Agent $j$, **then**
        Agent $j$ takes a target to Agent $k$ to balance their tasks
        **for** $p = j : (k-1)$ **do**
          $L_{split}(i,p) = L_{split}(i,p) + 1$
        **end for**
        put $q(j) = 1$ and $q(k) = 1$.
      **end if**
      **if** $\left(\bar{f}_{i,k} < \underline{f}_{i,j}\right) \& \left(\mathcal{N}_j > 1\right) \& \left(q(k) = 0\right)$, *i.e.* Agent $k$ has not been modified yet, Agent $j$ has at least two targets and its fitness $\underline{f}_{i,j}$ is larger than the fitness $\bar{f}_{i,k}$ of Agent $k$, **then**
        Agent $k$ takes a target to Agent $j$ to balance their tasks
        **for** $p = j : (k-1)$ **do**
          $L_{split}(i,p) = L_{split}(i,p) - 1$
        **end for**
        put $q(j) = 1$ and $q(k) = 1$.
      **end if**
    **end if**
  **end for**
**end for**

select also a solution with bad fitness to keep genetic diversity for avoiding the convergence to a local minimum. TSM can be described as follows:

1.Select randomly two subjects $(i,j)$ inside $N_{parent}$ with their associated fitness $\left(f_i, f_j\right)$ and split list $\left(L_{split}(i,:), L_{split}(j,:)\right)$.
2.Define the best subject using $f_i$ and $f_j$.

We take here $P_T = 1$ to ensure the best solution. As the tournament is performed by selecting randomly a couple of solutions, the diversity is not lost because two solutions with a large fitness can also be selected. A loop is performed to select the $N_{parent}$. Solution $L_{pop}(i,:)$, split list $L_{split}(i,:)$ and fitness list $F(i,:)$ of a chosen parent $i$ are stocked inside the lists $L_{parent}$, $L_{parent,\,split}$ and $F_{parent}$.

Once the $N_{parent}$ parents from the initial population have been chosen, a crossover or mutation is provoked for each parent with a probability of $P_{cross}$ and $P_{mut} = 1 - P_{cross}$. Two descendants from each of the $N_{parent}$ parents are generated to recreate the new generation of $N_{pop}$ subjects. If the crossover is selected for parent 1, another parent 2 is chosen randomly from the other remaining $N_{parent} - 1$ parents. Two children are generated from each parent. Children are stocked inside the list $L_{child}$ of size $N_{pop} \times N_t$ and $L_{child,\,split}$ of size $N_{pop} \times (N-1)$.

### 4.5 Crossover

Crossover is a method where the child inherits the characteristics of two parents for obtaining a better result. Crossover operator is chosen with probability $P_{cross}$, taken here $P_{cross} = 0.7$. In case of crossover, the second parent is selected randomly. A modified crossover inspired by the Greedy Crossover Method and adapted for our specific problem is proposed here.

For a couple of parents $(i,j) \in [1,\,\ldots,\,N_{parent}]^2$, let define the parent 1 with the route $R_1 = L_{parent}(i,:)$, fitness list $F_1 = F_{parent}(i,:)$ and slip list $S_1 = L_{parent,\,split}(i,:)$. Similarly, let define $R_2$, $F_2$ and $S_2$ for the parent 2. Finally, let $R_{child,\,1}$,$S_{child,\,1}$ and $R_{child,\,2}$, $S_{child,\,2}$ be the attributes of children.

Given the two parent routes $R_1$ and $R_2$, the first offspring is built with the following steps:

**A)** Choose a random number $b$ inside $[2,\,\ldots,N_t-1]$.

**B)** Check if the route leading from 1 to $b$ or from $b$ to the $N_t$ is the same in both $R_1$ and $R_2$, *i.e.* $R_1(1:b) = R_2(1:b)$ or $R_1(b:N_t) = R_2(b:N_t)$. If this happens, then

1.If $R_1(1:b) = R_2(1:b)$:
 (a)**Update children route:** $R_{child,\,1}(1:b) = R_1(1:b)$ and $R_{child,\,2}(1:b) = R_1(1:b)$.
 (b)**Define children split list:** Evaluate $f_1 = \sum_{j=1}^{b} F_1(j)$ and $f_2 = \sum_{j=1}^{b} F_2(j)$. If $f_1 \le f_2$, take $S_{child,\,1} = S_1$ and $S_{child,\,2} = S_1$. Else, take $S_{child,\,1} = S_2$ and $S_{child,\,2} = S_2$.
2.If $R_1(b:\text{end}) = R_2(b:\text{end})$,
 (a)**Update children route:** $R_{child,\,1}(b:N_t) = R_1(b:N_t)$ and $R_{child,\,2}(b:N_t) = R_1(1:b)$.
 (b)**Define children split list:** Evaluate $f_1 = \sum_{j=b}^{N_t} F_1(j)$ and $f_2 = \sum_{j=b}^{N_t} F_2(j)$. If $f_1 \le f_2$, take $S_{child,\,1} = S_1$ and $S_{child,\,2} = S_1$. Else, take $S_{child,\,1} = S_2$ and $S_{child,\,2} = S_2$.

**C)** Otherwise

1.the fitness of $R_1$ and $R_2$ of the $b$'s right edge are compared, *i.e.* $f_1 = \sum_{j=1}^{b} F_1(j)$ and $f_2 = \sum_{j=1}^{b} F_2(j)$. The shorter one is chosen as base for first children route, *i.e.* if $f_1 \le f_2$, take $R_{child,\,1}(1:b) = R_1(1:b)$ and $S_{child,\,1} = S_1$, else take $R_{child,\,1}(1:b) = R_2(1:b)$ and $S_{child,\,1} = S_2$.
2.the $b$'s left edge is compared in $R_1$ and $R_2$, *i.e.* $f_1 = \sum_{j=b}^{N_t} F_1(j)$ and $f_2 = \sum_{j=b}^{N_t} F_2(j)$. The shorter one is chosen as base for the second children route, *i.e.* if $f_1 \le f_2$, take $R_{child,\,2}(b:N_t) = R_1(b:N_t)$ and $S_{child,\,2} = S_1$, else take $R_{child,\,2}(b:N_t) = R_2(b:N_t)$ and $S_{child,\,2} = S_2$.

**D)** To complete $S_{child,\,1}$ and $S_{child,\,2}$, build a new route using the fitness of the both parents or each one

 1.For $S_{child,\,1}$(or $S_{child,\,2}$) , if $S_{child,\,1}(1:b)$ has already be defined, thus for $p$ from $b+1$ to $N_t$:
  (a)Put $a = S_{child,\,1}(p)$ and find the ranks $n_1$ and $n_2$ of element $a$ inside the list $S_1$ and $S_2$, *i.e.* find $S_1(n_1) = a$ and $S_2(n_2) = a$.
  (b)Check
   (1)If elements $S_1(n_1+1)$ and $S_2(n_2+1)$ exist ($S_1(n_1)$ and $S_2(n_2)$ are not the last element of their lists) and are not already inside of the list $S_{child,\,1}$, *i.e.* $\left(S_1(n_1+1) \notin S_{child,\,1}(1:p)\right)$ $\& \left(S_2(n_2+1) \notin S_{child,\,1}(1:p)\right)$, does
    •If $F_1(n_1+1) < F_1(n_2+1)$, take $S_{child,\,1}(p+1) = S_1(n_1+1)$,
    •Else, take $S_{child,\,1}(p+1) = S_2(n_2+1)$.
   (2)If $S_1(n_1+1)$ exists and $\left(S_1(n_1+1) \notin S_{child,\,1}(1:p)\right)$ $\& \left(S_2(n_2+1) \in S_{child,\,1}(1:p)\right)$, take $S_{child,\,1}(p+1) = S_1(n_1+1)$.
   (3)If $S_2(n_2+1)$ exists and $\left(S_1(n_1+1) \in S_{child,\,1}(1:p)\right)$ $\& \left(S_2(n_2+1) \notin S_{child,\,1}(1:p)\right)$, take $S_{child,\,1}(p+1) = S_2(n_2+1)$.
   (4)Else, pick a random element not already inside of the list $S_{child,\,1}$ to complete $S_{child,\,1}(p+1)$.
 2.If $S_{child,\,1}(b,:N_t)$ has already be defined, performed the same steps than in (a) for $p$ from $b-1$ to 1 and by tacking $S_1(n_1-1)$ and $S_2(n_2-1)$ instead of $S_1(n_1+1)$ and $S_2(n_2+1)$.

Two children have been created using this method. Stock $R_1$ and $R_2$ inside the list $L_{child}$ and $S_1$ and $S_2$ inside the list $L_{child,\,split}$.

### 4.6 Mutation

Mutation is used to preserve and introduce genetic diversity. It avoids the algorithm to converge to a local minimum when the population is similar. Mutation operator is associated with a mutation probability $P_{mut}$ such $P_{mut} = 1 - P_{cross}$, taken here $P_{mut} = 0.3$. Different mutation types exist (Flip-bit, Boundary, Gaussian, etc.), but the simplest one where only one modification is made with an equal probability for each element is chosen here. First, take one parent $L_{parent}(i,:)$ with


© The Institution of Engineering and Technology 2015

$L_{split}\,(i,:)$ and give its information to the child such as $R_1 = L_{parent}\,(i,:)$ and $L_1 = L_{split}\,(i,:)$. Two elements of the list $R_1$ are exchanged randomly following a uniform distribution. It allows mixing targets inside an agent list and/or exchange targets between two agents in the same operation. However, the split list $L_1$ stays unchanged by the mutation. This operation maintains the genetic diversity of our population and ensures proper convergence of the algorithm.

For example, suppose 3 agents $\{a,\, b,\, c\}$ associate to the list $R_1 = [2, 4, 1, 3, 5, 6]$ and the split list $L_1 = [3, 5]$, *i.e.* agent $a$ pursues $[2, 4]$, agent $b$ pursues $[1, 3]$ and agent $c$ pursues $[5, 6]$. We mutate $R_1$ by exchanging the element 4 and 3 (second and fourth position of the vector) to obtain $R_1 = [2, 3, 1, 4, 5, 6]$. Note agent $a$ and agent $b$ targets have changed to become $[2, 3]$ and $[1, 4]$, but agent $c$ targets stayed unchanged.

Perform this operation twice with the same parent to obtain two children $R_1$ and $R_2$. Stock $R_1$ and $R_2$ inside the list $L_{child}$ and $S_1$and $S_2$ inside the list $L_{child,\,split}$.

### 4.7 *New generation*

Once the $N_{pop}$ children have been generated from the initial population, the children become the new population such $L_{pop} = L_{child}$ and $L_{split} = L_{child,\,split}$, and cycle from Section 4.2 to Section 4.7 continues. Before updating the population, the solution $S_{min}$ of the current population with the best fitness $f_{min}$ is compared to the best solution of all previous loops noted $S_{opt}$, $L_{split,opt}$, $f_{opt}$. If $f_{min} \leq f_{opt}$, the solution is stocked to become the new optimal solution, *i.e.* $S_{opt} = S_{min}$, $L_{split,opt} = L_{split,min}$, $f_{opt} = f_{min}$.

### 4.8 *Stopping criteria*

This process finishes when the number of loops achieves a chosen $N_{\max} = 1000$ iterations without any fitness improvement, or when 10000 iterations have been performed for avoiding an infinite loop if the solution does not converge. The solution $S_{opt}$, $L_{split,opt}$ is thus considered as the quasi-optimal route. This one reflects the shortest time to recover all the targets from the agents' initial position.

## 5 Dynamic model

The GA exposed in Section 4 can be performed for different agents and targets dynamics. In this paper, the problem is focused on targets, for example buoys, pursued by sailboats, and so the specific constraints of sailboat dynamics. In opposite to motorboats and USV that can control their position and velocity flexibly, the position and the velocity of sailboats depend on wind direction and speed. Moreover, autonomous sailboat has difficulty in accelerating or decelerating rapidly. Thus, the following sections expose the strategy for implementing the GA with sailboats. First, the dynamics of the sailboat and an approximation of its average velocity for variable wind conditions are to be exposed. Second, a method to evaluate the interception time with a target is provided, using the approximation of the sailboat velocity. This interception time is used in the GA.

### 5.1 *Sailboat dynamics*

The dynamic model of a sailboat is complex and it requires the knowledge of parameters difficult to obtain in practice. However, for a specific wind direction and speed, it is possible to create a diagram of the average velocity of a sailboat. Since the distance between two targets is supposed to be large here, the dynamic model can be simplified to assume a constant sailboat velocity for a defined orientation and wind speed. Moreover, the time to perform maneuvers are much smaller than the time to reach the target. This simple dynamic model allows to implement the algorithm with a modest processing time.

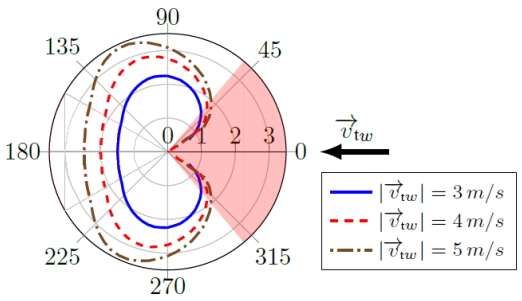



**Fig. 2**: Polar diagram of sailboat velocity function of wind velocity $v_{tw}$. Red areas correspond to sailing direction to avoid (dead area). [21]

Line following control for a sailboat has been studied in [8, 10, 17, 18]. The sailboat $k$ dynamics is approximated the following model:

$$p_{s,k}\,(t_0 + T) = p_{s,k}\,(t_0) + v_{s,k}\,(t_0)\,T \tag{1}$$

where $T \geq 0$ a chosen time, $p_{s,k} = \left[x_{s,k},\, y_{s,k}\right]$, $v_{s,k} = V_{s,k}\left[\cos\left(\theta_{s,k}\right),\, \sin\left(\theta_{s,k}\right)\right]$ are the sailboat $k$ Cartesian coordinate and velocity, with $\theta_{s,k}$ the constant sailboat heading during $t \in [t_0,\, t_0 + T]$ chosen to intercept sailboat target as exposed in Section 6, and $V_{s,k}$ is the average velocity of the sailboat during $t \in [t_0,\, t_0 + T]$, function of $\theta_{s,k}$ and wind parameters as exposed in Section 5.2.

### 5.2 *Current sailboat velocity*

Sailboats' velocity depends on the apparent wind direction $\psi_{aw}$ and the apparent wind speed $v_{aw}$, which can be evaluated from the current sailboat direction and speed and the true wind direction $\psi_{tw}$ and the wind speed $v_{tw}$. As shown in [6, 21], a diagram of the average velocity of a sailboat with regards to the wind direction can be plotted using only the true wind, see Figure 2. Thus, one may write $V_{s,k} = f\left(\theta_{s,k}, \psi_{tw}, v_{tw}\right)$ where $f$ is a known function corresponding to the polar diagram of the sailboat velocity.

Let $[\psi_{tw} + \pi - \delta,\, \psi_{tw} + \pi + \delta]$ with the hauled angle $\delta$ be the dead area where sailboat is considered as upwind (pink area in Figure 2), with $\delta$ is taken equal to $\frac{\pi}{4}$. When a desired orientation $\bar{\theta}_{s,k}$ is inside the dead area, the sailboat needs to perform tacks at $45^\circ$ around $\bar{\theta}_{s,k}$: the distance performed to be multiplied by two, which is equivalent to the sailboat travelling in the direction $\theta_{s,k} = \bar{\theta}_{s,k}$ with a velocity divided by two. Thus, the sailboat velocity model can be approximated by

- If $\cos\left(\psi_{tw}\,(t) - \theta_{s,k}\,(t)\right) + \cos\left(\delta\right) > 0$

$$V_{s,k}\,(t) = v_{\max}\left(v_{tw}\,(t)\right) \tag{2}$$

- Else $V_{s,k}\,(t) = \frac{v_{\max}(v_{tw}(t))}{2}$ where

$$v_{\max}\left(v_{tw}\,(t)\right) = \max_{\substack{\theta_{s,k} \in [-\pi,\, \pi] \\ \psi_{tw} \in [-\pi,\, \pi]}} \left(f\left(\theta_{s,k},\, \psi_{tw},\, v_{tw}\,(t)\right)\right), \tag{3}$$

where $f$ is a known function corresponding to the polar diagram of sailboat velocity, for example, Figure 2. In the same way, one may define

$$v_{\min}\left(v_{tw}\,(t)\right) = \frac{v_{\max}\left(v_{tw}\,(t)\right)}{2}, \tag{4}$$

which is the velocity of the sailboat when it follows a tacking trajectory. These notations are to be used in the next sections.


© The Institution of Engineering and Technology 2015

### 5.3 Target moving prediction

The literature offers different methods to simulate and predict current movements in the sea [16], but all are based on complex models and require the values of those parameters that are difficult to obtain. Thus a simple model is used here to reduce the processing time for the GA.

The Newton's motion equation is considered to evaluate target dynamics:

$$p_{t,k}\left(t_0+T\right)=p_{t,k}\left(t_0\right)+v_{t,k}\left(t_0\right)T+\frac{1}{2}a_{t,k}\left(t_0\right)T^2 \quad (5)$$

where $p_{t,k}$ , $v_{t,k}=V_{t,k}\left[\cos\left(\theta_{t,k}\right),\sin\left(\theta_{t,k}\right)\right]$ and $a_{t,k}$ are the target $k$ coordinate, velocity and acceleration, and $T$ a discrete time. Suppose we have access to $v_{t,k}$ and $a_{t,k}$ using a method like Time series, Newton's motion equation or by tacking them constant. The comparison of these three methods is provided in [3]. Note in these methods $a_{t,k}$ is very small. In [3] for example, future locations of the buoys are evaluated using Newton's motion equations based on historical real data provided by GPS.

## 6 Target interception

To define the fitness in the GA, the interception instant and the position between the sailboat and its target is required. The following sections explain how to find them with different constraints due to the wind conditions.

### 6.1 Interception time calculation

Consider the first sailboat velocity, noted $V_s^*\left(t_0\right)$, at the initial instant $t=t_0$ can be evaluated using Section 6.2 with the first evaluation of the sailboat orientation

$$\theta_s^*=\text{atan2}\left(y_t*-y_s\left(t_0\right),x_t*-x_s\left(t_0\right)\right) \quad (6)$$

where $p_t^*=[x_t^*,\,y_t^*]=[x_s\left(t_0\right),\,y_s\left(t_0\right)]$,
$p_s\left(t_0\right)=[x_s\left(t_0\right),\,y_s\left(t_0\right)]$ and $p_s\left(t_f\right)=\left[x_s\left(t_f\right),\,y_s\left(t_f\right)\right]$.
For initial position of the sailboat $p_s\left(t_0\right)$ and the target $p_t\left(t_0\right)$, the interception time $t_f$ can be expressed as $t_f=t_0+T$ where $T$ can be found by solving

$$0=AT^4+BT^3+CT^2+DT+E \quad (7)$$

with

$$A=\frac{1}{4}a_t\left(t_0\right)^2$$
$$B=v_{t,x}\left(t_0\right)a_{t,x}\left(t_0\right)+v_{t,y}\left(t_0\right)a_{t,y}\left(t_0\right)$$
$$C=\left(x_t\left(t_0\right)-x_s\left(t_0\right)\right)a_{t,x}\left(t_0\right)+\left(y_t\left(t_0\right)-y_s\left(t_0\right)\right)a_{t,y}\left(t_0\right)+\left(V_t\left(t_0\right)^2-V_s^*\left(t_0\right)^2\right)$$
$$D=\left(x_t\left(t_0\right)-x_s\left(t_0\right)\right)v_{t,x}\left(t_0\right)+\left(y_t\left(t_0\right)-y_s\left(t_0\right)\right)v_{t,y}\left(t_0\right)$$
$$E=\left(x_t\left(t_0\right)-x_s\left(t_0\right)\right)^2+\left(y_t\left(t_0\right)-y_s\left(t_0\right)\right)^2$$

and $v_{t,x}=V_t\left(t_0\right)\cos\left(\theta_t\left(t_0\right)\right)$, $v_{t,y}=V_t\left(t_0\right)\sin\left(\theta_t\left(t_0\right)\right)$, similarly for $a_{t,x}$ and $a_{t,y}$.

The corresponding proof is provided in 9.1. The value of $T$ is the smallest positive real part of these solutions. If no solution corresponds to this criterion, it means that the sailboat cannot reach the target. Using $T$, the intersection time $t_f=T+t_0$ and the interception position $p_t\left(t_f\right)=p_t\left(t_0\right)+v_t\left(t_0\right)T+\frac{1}{2}a_{t,k}\left(t_0\right)T^2$ can be defined. From it, one may find the desired orientation of the sailboat $\forall t\in\left[t_0,\,t_f\right]$

$$\theta_s\left(t\right)=\text{atan2}\left(y_t\left(t_f\right)-y_s\left(t_0\right),x_t\left(t_f\right)-x_s\left(t_0\right)\right)$$

The evaluated orientation $\theta_s\left(t_0\right)$ induces a new evaluation of the sailboat velocity $V_s\left(t_0\right)$ using $\theta_s^*\left(t_0\right)$ and taking $p_t^*=p_t\left(t_f\right)=\left[y_t\left(t_f\right),\,y_t\left(t_f\right)\right]$. If $V_s^*\left(t_0\right)\neq V_s\left(t_0\right)$, a second evaluation of $T$ is required using $V_s^*\left(t_0\right)=V_s\left(t_0\right)$ and $p_t^*=p_t\left(t_f\right)$.

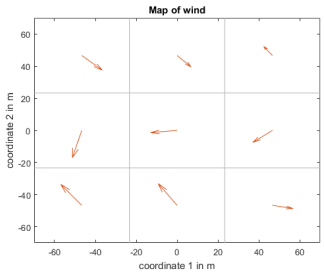



**Fig. 3**: Map of wind distribution

### 6.2 Average sailboat velocity during interception

In practice when vessels sail on a large area, the wind direction and speed are not identical at all positions and they are changing with the time, influencing the sailboat velocity and so the interception point. An average value of the sailboat velocity $V_s$ must be defined by taking into account these conditions. However, since evaluating the true average sailboat velocity by taking into account variations is complicated and time consuming, an approximation of this average value is performed by taking a fixed number of points $N_{diff}$ between the initial position of the sailboat and its target. These ones are defined at an equal distance of each other, and their number is proportional to the number of changes. This method allows performing the velocity approximation in a short time for the GA.

*Wind matrices* Let first define the wind distribution map $\mathcal{M}_w$ divided in a grid of size $N_h\times N_v$. The wind orientation and velocity are supposed to be identical inside a square as shown in Figure 3.

Let $M_\psi\left(t\right)$ and $M_v\left(t\right)$ be matrices of wind orientation $\psi_{tw}$ and velocity $v_{tw}$ which contain the different values of $\psi_{tw}$, $v_{tw}$ corresponding to the wind map distribution. For example, $M_\psi\left(1,1\right)\left(t\right)=\frac{-\pi}{4}$ corresponds to the first case in the map of the Figure 3. Assume there exist a duration $\Delta t_c$ such $M_\psi\left(t\right)$ and $M_v\left(t\right)$ can be considered constant during this interval $I=[t,\,t+\Delta t_c]$, *i.e.* $\forall\left(t_1,t_2\right)\in I\;\; M_\psi\left(t_1\right)\approx M_\psi\left(t_2\right)$ and $M_v\left(t_1\right)\approx M_v\left(t_2\right)$. The wind data are supposed to be collected using systems like NOAA or CMEMs.

Let define the index coordinate $P_t\left(t\right)\in[1:\,N_h]\times[1:\,N_v]$ of the target coordinate $p_t$ inside $\mathcal{M}_w$. In the same way, let define $P_s$ for the sailboat coordinate $p_s$. Notation $M_\psi\left(P_s\right)\left(t\right)$ corresponds to the value of the wind orientation for the sailboat at time $t$.

*Choice of $N_{diff}$* Let define the distance $d_{st}=\sqrt{\left(x_t^*-x_s\left(t_0\right)\right)^2+\left(y_t^*-y_s\left(t_0\right)\right)^2}$ where $[x_t^*,\,y_t^*]$ are target value defined in Section 6.1, and $\bar{t}=\frac{d_{st}}{v_{\min}\left(M_v\left(P_t\right)\left(t_0\right)\right)}$ a first rude approximation of the time required to reach target position with the minimum velocity of the sailboat. Then, one may define

$$N_{diff}=\left|P_t\left(1\right)\left(t\right)-P_s\left(1\right)\left(t\right)\right|+\left|P_t\left(2\right)\left(t\right)-P_s\left(2\right)\left(t\right)\right|+\left\lfloor\frac{\bar{t}}{\Delta t_c}\right\rfloor+2 \quad (8)$$

where $\lfloor x\rfloor$ is the whole part of $x$.

One may observe:

- $N_{diff}=2$ if $P_t=P_s$, *i.e.* the sailboat and the target are in the same area with similar wind conditions, and if $\frac{\bar{t}}{\Delta t_c}<1$,


© The Institution of Engineering and Technology 2015

which means the sailboat catches its target before wind conditions have changed significantly. Thus, only two points, the initial position and the final position, are used to measure the wind conditions and to evaluate the sailboat velocity.

- $N_{diff} = 3$ if
  - $P_t$ and $P_s$ are two adjacent areas with two different wind conditions. Thus, take the initial position, the final position and a middle point to know in which area the sailboat travels during the longest time (for example, if the target/sailboat is close to an edge of an area, the distances inside areas are not equal).
  - $P_t$ and $P_s$ are in the same area but the traveling time is enough for wind conditions to change significantly. Thus, take the initial position, the final position and a middle point to know in which wind condition the sailboat travels during the longest time.
- $N_{diff} > 3$ if
  - $P_t$ and $P_s$ are in two distant areas. $N_{diff}$ is chosen to be large enough to take at least one measurement in each crossed area plus one. For example, if $P_s = (1, 1)$ and $P_t = (1, 4)$, the sailboat needs to cross 4 areas, thus $N_{diff} = |1 - 4| + 2 = 5$, the 4 areas plus one.
  - $P_t$ and $P_s$ are in the same area, but the sailboat moves slowly with comparison to the change of wind conditions. Similarly with the previous points, $N_{diff}$ is chosen to take a measurement for each wind condition.
  - Combination of both previous cases.

*Average velocity calculation* Initialize $t = t_0, \bar{p}_s = p_s(t_0), \theta_s = \text{atan2}(y_t^* - y_s(t_0), x_t^* - x_s(t_0))$ where $p_t^* = [x_t^*, y_t^*]$ are target value defined in Section 6.1,

$d_{st} = \sqrt{(x_t^* - x_s(t_0))^2 + (y_t^* - y_s(t_0))^2}$ and $V_s^* = 0$.

- For $i = 1 : (N_{diff} - 1)$,
  - Evaluate the index coordinate $\bar{P}_s(t)$ associated to $\bar{p}_s$
  - Take $\psi_{tw} = M_\psi\left(\bar{P}_s(t)\right)(t)$ and $v_{tw} = M_v\left(\bar{P}_s(t)\right)(t)$.
  - Evaluate $\bar{v}$ using $\psi_{tw}$ , $v_{tw}$ and $\theta_s$ with the method exposed in Section 5.2
  - Put $v = v + \frac{\bar{v}}{N_{diff}}$, $t = t + \frac{d_{st}}{\bar{v}(N_{diff}-1)}$ and $\bar{p}_s = \bar{p}_s + \frac{d_{st}}{(N_{diff}-1)}[\cos(\theta_s), \sin(\theta_s)]$.
- Evaluate $\bar{v}$ using method exposed in Section 5.2 by taking $\psi_{tw} = M_\psi(P_t(t))(t)$, $v_{tw} = M_v(P_t(t))(t)$ and $\theta_s$. Take $v = v + \frac{\bar{v}}{N_{diff}}$

Choose $V_s^*(t_0) = v$ (or $V_s(t_0) = v$ respectively) in Section 6.1.

## 7 Simulation

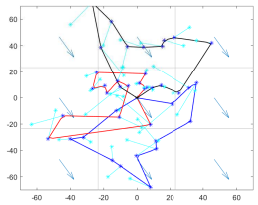
In this section, the result of the proposed GA algorithm is compared with the performance of the algorithm exposed in [4]. Since [4] studied fishing motor boat and not sailboats, the new prediction of the sailboat velocity and targets interceptions exposed in Section 6 are used in the existing GA algorithm to obtain a fair comparison between these two methods. Moreover, "home" has been added as a final target to reach in the previous GA to come back at the initial position. Major difference between these two algorithms is the second part of the crossover which manages the number of targets assigned to each agent. In [4], an asexual crossover is employed.

Let $\mathcal{N} = \{\mathcal{N}_1, \ldots, \mathcal{N}_N\}$ be the set of number of targets collected by each agents and let $N_{tour}$ be the number of iterations made by the GA to find a solution.

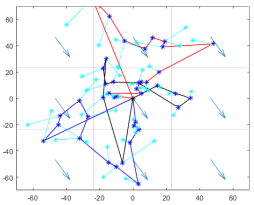
Figures 4 (a),(b),(c) and (d) compare the results of these two methods. Target positions and orientations are chosen randomly. In Figure 4 (a) and (b), paths selected by both approaches are evaluated without taking into account the wind direction. In Figure 4 (c) and (d) where the effects of the wind direction are added. The trajectory evaluated by the GA allows the sailboat to collect targets while avoiding certain configurations such as going upwind (bold lines).

In Figure 5, a configuration in cross is chosen to obtain a simpler configuration to understand. Two different velocities of the target are chosen, $V_t = 1$ and $V_t = 0.5$, and the maximal sailboat velocity is $V_s = 8$. For $V_t = 0.5$ in Figure 5 (a) and (b), one may observe that the solutions found by the proposed GA and the existing GA are close. However, when $V_t = 1$ in Figure 5 (c) and (d), the targets are sparse. In this configuration, the tasks for collection become more challenging. The random selection of the population at the beginning of the GA makes the optimal solution difficult to find. A SOM like [7, 26] could obtain a better performance than the GA approaches for this particular configuration while these two GA methods can be more efficient than the SOM in other complex configurations.

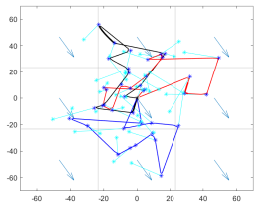
It can be observed that the proposed GA fitness is smaller than the existing GA fitness while the existing GA converges faster to its final solution. It can be explained by the second part of the crossover, which makes the number of targets assigned to the agents less stable and so the evaluation of its fitness. This allows to better balance the mission time between sailboats.



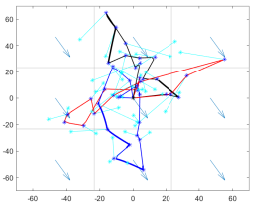
GA. Fitness: 32.49 without considering wind orientation. $\mathcal{N} = \{14, 13, 13\}$. $N_{tour} = 3865$



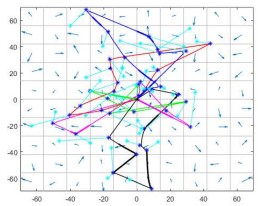
Old GA. Fitness: 35.24 without considering wind orientation. $\mathcal{N} = \{16, 13, 11\}$. $N_{tour} = 2738$.



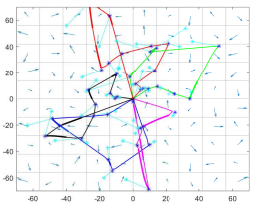
GA. Fitness: 32.488 with wind orientation. $\mathcal{N} = \{16, 12, 12\}$. $N_{tour} = 4422$



Old GA. Fitness: 35.09 with wind orientation. $\mathcal{N} = \{13, 14, 13\}$. $N_{tour} = 4536$



GA. Predicted fitness: 32.05. $\mathcal{N} = \{10, 9, 9, 6, 6\}$. $N_{tour} = 3013$



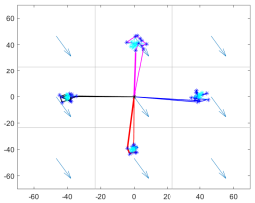
Old GA. Fitness: 34.5. $\mathcal{N} = \{10, 10, 5, 7, 8\}$. $N_{tour} = 2956$

**Fig. 4**: Comparison GA and SOM with different wind directions. $N = 3$ agents following $N_t = 40$ targets. Red, blue, black and magenta lines: $N$ sailboat trajectories. Cyan lines: target trajectories from their initial position (cyan stars) to their final position when they are collected (blue stars). Bold lines correspond to a configuration where the sailboat is upwind.

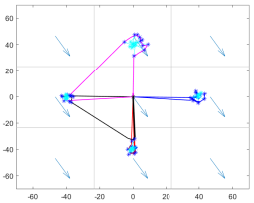
## 8 Conclusion

In this paper, a genetic algorithm for collecting multiple moving targets using a multi-agent system is provided. This proposed algorithm finds the shortest time to collect all the moving targets and come back to the initial position. A new method to choose

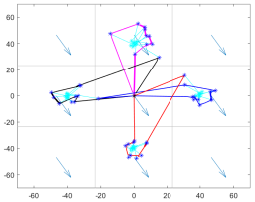

© The Institution of Engineering and Technology 2015



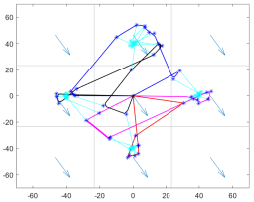
GA. Fitness: 18.03. $\mathcal{N} = \{10, 10, 10, 10\}$. $N_{tour} = 1351$, target velocity $V_t = 0.5$.



Old GA. Fitness: 21.25. $\mathcal{N} = \{8, 10, 11, 11\}$, target velocity $V_t = 0.5$. $N_{tour} = 1351$.



GA. Fitness: 22.97. $\mathcal{N} = \{9, 10, 11, 10\}$, target velocity $V_t = 1$. $N_{tour} = 3825$.



Old GA. Fitness: 27.99. $\mathcal{N} = \{8, 10, 11, 11\}$, target velocity $V_t = 1$. $N_{tour} = 2531$.

**Fig. 5**: Comparison of GA and SOM with cross configuration. $N = 5$ agents following $N_t = 40$ targets. Red, blue, black and magenta lines: $N$ sailboat trajectories. Cyan lines: target trajectories from their initial position (cyan stars) to their final position when they are collected (blue stars).

the number of targets to attribute to each agent has been developed so as to reduce the overall time of the mission. This method can be easily adapted to different kinds of vehicles.

Adaptation of this genetic algorithm for a fleet of sailboats is also provided. A calculation of interception instant and position is proposed. This proposed algorithm uses an approximation of the average sailboat velocity to reduce the processing time in the loop of the GA. This average velocity takes into account several parameters such as wind direction and speed, which change with the time and the localization of the sailboats. Simulation results show the effectiveness of the proposed method with comparison to a existing GA.

In future work, the attribution of targets to agents is to be explored so as to guarantee an optimal configuration for some particular scenarios, for example, when the targets are grouped in a small stack.

## Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of the U.K. under Grant EP/R005532/1.

## 9 Appendix

### *9.1 Evaluation target interception time*

Let $d_s$ be the distance performed by the sailboat between its initial solution $p_s(t_0) = [x_s(t_0), y_s(t_0)]$ and its final position $p_s(t_f) = [x_s(t_f), y_s(t_f)]$. Since (1), this distance is equal to

$$v_s(t_0)^2 T^2 = d_s^2 \tag{9}$$

where $v_s(t_0)$ is the sailboat velocity. From (9), one may write

$$v_s(t_0)^2 T^2 = (x_s(t_f) - x_s(t_0))^2 + (y_s(t_f) - y_s(t_0))^2$$

and since the final position of the sailboat is the same to the target position, *i.e.* $p_s(t_f) = p_t(t_f)$, one has

$$v_s(t_0)^2 T^2 = (x_t(t_f) - x_s(t_0))^2 + (y_t(t_f) - y_s(t_0))^2$$

Using (5), one may write

$$v_s(t_0)^2 T^2 = \left((x_t(t_0) - x_s(t_0)) + v_{t,x}(t_0) T + \frac{1}{2} a_{t,x}(t_0) T^2\right)^2 + \left((y_t(t_0) - y_s(t_0)) + v_{t,y}(t_0) T + \frac{1}{2} a_{t,y}(t_0) T^2\right)^2$$

and formatted as

$$0 = 2\left[(x_t(t_0) - x_s(t_0)) v_{t,x}(t_0) + (y_t(t_0) - y_s(t_0)) v_{t,y}(t_0)\right] T + \left[(x_t(t_0) - x_s(t_0)) a_{t,x}(t_0) + (y_t(t_0) - y_s(t_0)) a_{t,y}(t_0)\right] + \left(v_t(t_0)^2 - v_s(t_0)^2\right)\Big] T^2 + d_{ts}(t_0)^2 + \left[v_{t,x}(t_0) a_{t,x}(t_0) + v_{t,y}(t_0) a_{t,y}(t_0)\right] T^3 + \frac{1}{4} a_t(t_0)^2 T^4$$

We put $A$, $B$, $C$, $D$ and $E$ as described in Section 6.1 and one get $0 = AT^4 + BT^3 + CT^2 + DT + E$, which can be solved to obtain different values of $T$.

## *10* References


1 A. Baykasoğlu and Z. DU Durmuşoğlu. A multi-agent based approach to modeling and solving dynamic generalized travelling salesman problem. *Journal of Intelligent & Fuzzy Systems*, 31(1):77–90, 2016.

2 H. Chen, G. Tan, G. Qian, and R. Chen. Ant colony optimization with tabu table to solve tsp problem. In *In Proc Chinese Control Conference (CCC)*, volume 1, pages 2523–2527, 2018.

3 Carlos G., Antonio S., and Xosé H V. Solving the dynamic traveling salesman problem using a genetic algorithm with trajectory prediction: An application to fish aggregating devices. *In Proc. Computers & Operations Research*, 56:22–32, 2015.

4 C. Groba, A. Sartal, and X. H. Vázquez. Integrating forecasting in meta-heuristic methods to solve dynamic routing problems: Evidence from the logistic processes of tuna vessels. *Engineering Applications of Artificial Intelligence*, 76:55–66, 2018.

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

6 P. Herrero, L. Jaulin, J. Vehí, and M.A. Sainz. Guaranteed set-point computation with application to the control of a sailboat. *International journal of control, automation and systems*, 8(1):1–7, 2010.

7 H. Huang, D. Zhu, and F. Ding. Dynamic task assignment and path planning for multi-auv system in variable ocean current environment. *Journal of intelligent & robotic systems*, 74(3-4):999–1012, 2014.

8 L. Jaulin and F. Le Bars. A simple controller for line following of sailboats. In *Robotic Sailing 2012*, pages 117–129. 2013.

9 Q Jiang, R Sarker, and H Abbass. Tracking moving targets and the non-stationary traveling salesman problem. *Complexity International*, 11(2005):171–179, 2005.

10 F. Le Bars and L. Jaulin. An experimental validation of a robust controller with the vaimos autonomous sailboat. In *Robotic Sailing 2012*, pages 73–84. 2013.

11 D. Marlow, P. Kilby, and GN Mercer. The travelling salesman problem in maritime surveillance–techniques, algorithms and analysis. In *In Proc of the International Congress on Modelling and Simulation*, pages 684–690, 2007.

12 M. Mavrovouniotis, F. M. Müller, and S. Yang. Ant colony optimization with local search for dynamic traveling salesman problems. *In Proc. IEEE transactions on cybernetics*, 47(7):1743–1756, 2017.

13 J. Melin. Modeling, control and state-estimation for an autonomous sailboat, 2015.

14 J. Melin, K. Dahl, and M. Waller. Modeling and control for an autonomous sailboat: a case study. In *Robotic Sailing 2015*, pages 137–149. 2016.

15 J. Monnot and S. Toulouse. The traveling salesman problem and its variations. *Paradigms of Combinatorial Optimization: Problems and New Approaches*, pages 173–214, 2014.

16 T. M. Özgökmen, A. Griffa, A. J. Mariano, and L. I. Piterbarg. On the predictability of lagrangian trajectories in the ocean. *Journal of Atmospheric and Oceanic Technology*, 17(3):366–383, 2000.

17 C. Pêtrès, M-A Romero-Ramirez, and F. Plumet. A potential field approach for reactive navigation of autonomous sailboats. *Robotics and Autonomous Systems*, 60(12):1520–1527, 2012.

18 F. Plumet, H. Saoud, and M-D Hua. Line following for an autonomous sailboat using potential fields method. In *Proc MTS/IEEE, OCEANS-Bergen*, pages 1–6, 2013.



© The Institution of Engineering and Technology 2015

19 A. P. Punnen. The traveling salesman problem: Applications, formulations and variations. In *The traveling salesman problem and its variations*, pages 1–28. 2007.
20 H. Saoud, M-D Hua, F. Plumet, and F Amar. Modeling and control design of a robotic sailboat. In *Robotic Sailing 2013*, pages 95–110. 2014.
21 Hadi Saoud, Minh-Duc Hua, Frédéric Plumet, and Faiz Ben Amar. Routing and course control of an autonomous sailboat. In *Proc IEEE ECMR*, pages 1–6, 2015.
22 A. Stieber and A Fügenschuh. The multiple traveling salesmen problem with moving targets and nonlinear trajectories. In *In Proc Operations Research*, pages 489–494, 2018.
23 A. Stieber, A. Fügenschuh, M. Epp, M. Knapp, and H. Rothe. The multiple traveling salesmen problem with moving targets. *Optimization Letters*, 9(8):1569–1583, 2015.
24 C. Viel, U. Vautier, J. Wan, and L. Jaulin. Position keeping control of an autonomous sailboat. *In Proc. IFAC CAMS*, 51(29):14–19, 2018.
25 L. Xiao and J. Jouffroy. Modeling and nonlinear heading control of sailing yachts. *In Proc IEEE Journal of Oceanic engineering*, 39(2):256–268, 2014.
26 D. Zhu, H. Huang, and S. X. Yang. Dynamic task assignment and path planning of multi-auv system based on an improved self-organizing map and velocity synthesis method in three-dimensional underwater workspace. *In Proc. IEEE Transactions on Cybernetics*, 43(2):504–514, 2013.


© The Institution of Engineering and Technology 2015