Progress in Artificial Intelligence (2021) 10:391–416
https://doi.org/10.1007/s13748-021-00245-3

REGULAR PAPER

When robots contribute to eradicate the COVID-19 spread in a context of containment

Naila Aziza Houacine[1] · Habiba Drias[1]

Received: 29 June 2020 / Accepted: 16 April 2021 / Published online: 11 May 2021

Abstract
In the era of autonomous robots, multi-targets search methods inspired researchers to develop adapted algorithms to robot constraints, and with the rising of Swarm Intelligence approaches, Swarm Robotics became a very popular topic. In this paper, the problem of searching for an exponentially increasing number of targets in a complex and unknown environment is addressed. Our main objective is to propose a Robotic target search strategy based on the Elephants Herding Optimization (EHO) algorithm, namely Robotic-EHO (REHO). The main additions were the collision-free path planning strategy, the velocity limitation, and the extension to the multi-target version in discrete environments. The proposed method has been the subject of many experiments, emulating the search of infected individuals by COVID-19 in a context of containment within complex and unknown random environments, as well as in the real case study of the USA. The particularity of these environments is their increasing targets' number and the dynamic Containment Rate (CR) that we propose. The experimental results show that REHO reacts much better in high CR, early start search mission, and where the robots' speed is higher than the virus spread speed.

Keywords Swarm robotics · Swarm intelligence · COVID-19 · Target detection problem · Containment · Autonomous robots

1 Introduction

After the appearance of the SARS in 2002, A(H1N1) in 2009, MERS in 2012, and Ebola in 2014, viruses continue to emerge and spread in 2019 with the new SARS-CoV-2 called COVID-19. The latter was declared as a pandemic by the World Health Organization (WHO) in March 2020. Subsequently, the focus of recent research has been on modeling approaches to estimate the outbreak growth and the impact of different individual and governmental measures to stop the spread of the COVID-19. Almost all studies noticed the impressive growing number of infected people and agree on the importance of active surveillance, contact tracing, containment, quarantine, and early strong social distancing efforts to stop the transmission of the virus, in addition to the need for massive screening of potentially infected individuals in order to isolate them. This last point can be linked to a problem of searching for targets in a complex and unknown environment, where the targets may be potentially affected individuals. The search will then be guided by contact tracing methods in a containment context. As reported in [1], in the Target Detection Problems (TDP) the objective is to find or detect one or multiple targets in a given environment. They may be tackled with one or multiple sensors. When mobile sensors are exploited, it is referred to as a mobile search, the problem becomes related to path planning. Target search is known as one of the benchmark problems in Swarm Robotics (SRs), which are basically Multi-Robots Systems (MRSs) with some special properties. Mobile robots can operate in complex and unknown environments by removing the need for human intervention. They perform tasks that humans cannot accomplish or are risky for them with reduced cost by the application of Swarm Intelligence (SI) algorithms [2,3].

This paper proposes an extension of Elephants Herding Optimization (EHO) [4,5] to SRs, named as REHO (Robotic-EHO), taking into account real-robots limits and an obstacle avoidance strategy. REHO is a Swarm Intelligence approach adapted for dynamic multi-target search in complex and unknown environments, where dynamicity is expressed by the variable Containment Rate and target's

✉ Naila Aziza Houacine
nhouacine@usthb.dz

Habiba Drias
hdrias@usthb.dz

[1] LRIA, USTHB, BP 32 El-Alia Bab-Ezzouar, Algiers, Algeria

number growing exponentially. The contribution of this paper is summarized in two points: first, based on existing static environments representation for targets' search problem, we propose a representation of dynamic environments to represent the COVID-19 (and other close-contact pathogens) spread and evolution in the time (days). To our knowledge, this modeling for TDP has never been proposed before. Then, we propose a Robotic adaptation of EHO algorithm (REHO) inspired by the herding behavior of elephant groups for the position update. A collision free path planning strategy is then provided for discrete environments (sampling-based). Such a result is exploited in this work to break the transmission chain and eradicate the COVID-19 spread.

The remainder of the paper is organized into six sections. The next Section presents an overview of recent research efforts related to both of the COVID-19 and Swarm Robotics approaches of the targets' detection problem. Section 3 is devoted to the modeling of the dynamic environment to mimic the virus spread. Conversion to a target detection problem is also presented here. In Sect. 4 we introduce our REHO search approach. Experimental results are presented and analyzed in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Related work

From the first apparition of the coronavirus disease (COVID-19) in December 2019 in Wuhan, China, to its global spread around the world, the number of researchers working on the subject has widely increased and it continues to attract attention. Recent studies presented in [6,7] indicate that the COVID-19 is mainly characterized by three most important parameters. First, the initial reproductive number or reproductive rate R_0, which is a measure of transmission that provides the number of persons that one infected person contaminates per day. The second is the initial number of cases before starting applying any interventions, and the last one is the incubation period of the virus.

Several models for the COVID-19 outbreak have been proposed to estimate the evolution of the virus taking into account different actions and interventions which aim to stop its spread. The main key to achieve this is to reduce the effective reproduction number, R_t to a value under 1 [6]. For that, the model in [7] considers the individual behavioral reaction and governmental actions, like holiday extension, travel restriction, hospitalization, and containment. The author in [8] focused on the impact of traditional public health outbreak response tactics: isolation, quarantine, social distancing, and community containment. Besides, authors in [9] developed two modeling approaches to infer the growth rate of the outbreak when taking non-pharmaceutical measurements such as contact tracing, quarantine, and social distancing. In the article [10] authors used a mathematical model to assess the virus evolution when applying isolation and contact tracing.

Also, in [11] an epidemiological modeling approach for COVID-19 that take into account the lockdown measures is proposed.

All of these researches show that active surveillance, contact tracing, quarantine, and early strong social distancing efforts are needed to stop transmission of the virus. They all share two main points that are the big impact of containment and the importance of contact tracing. As reported in [12], initial containment measures taken in China proved to reduce human-to-human transmission successfully, which strengthens the forecasting's validity of the previously cited studies.

However, most of these studies do not take into account the effective dynamicity of the containment, which is an extremely important parameter that directly affects the reproduction rate R_t.

Authors of [10] also mentioned that the containment is not enough, due to the long incubation period of the virus, it is necessary to initiate a contact tracing procedure in addition to the containment measures. To solve this issue, many researchers have proposed various methods of contact tracing as mentioned in [13,14]. Currently, after determining the potentially infected people, few governments take care of recovering their phone numbers to contact and notify them. However, this is not sufficient at all, it is necessary to reach them so that they carry out the PCR[1] test, and to support them in case of proven contamination.

A solution to this problem would be to carry out a targeted screening, by considering this task as being a Multi-target search problem in a complex and unknown environment. In order to avoid endangering more people because of the lack of staff to face this pandemic, in addition to its many advantages, Multi-Robots Systems are the most suitable for this task. Numerous researches were undertaken to solve the target research problem. Due to their efficacy, most of the recent resolution methods use Swarm Intelligence algorithms as the cooperative strategy of Swarm Robotics.

On the one hand, we have MSL-PSO (Multi-Swarm PSO with LS) which is a hybrid of modified Particle Swarm Optimization and Local Search on a Multi-robot Search System [15], A-RPSO (Adaptive Robotic Particle Swarm Optimization) [3] that provides an adaptive inertia weight to avoid local optima in addition to the obstacle avoidance strategy of the RPSO [16], and Multi-swarm hybrid FOA-PSO (MFPSO) [17] which is a hybrid of PSO and Fruit Fly Optimization Algorithm enhanced with a Multi-swarm strategy and a Multi-Scale Cooperative Mutation used against the limitation of local optima.

We have found that all these PSO-based approaches have only been applied for the search of a single target where

[1] Polymerase Chain Reaction.

A-RPSO and MFPSO perform well. But when it comes to multi-target search, they experience a loss in diversity and suffer from stagnation after having converged on a first target. Furthermore, these approaches have only been applied in completely static environments and without providing the full path from the robot's initial positions to the target.

On the other hand, we have some approaches that consider multi-target search. In [18], Multi-Target based methods such as Improved Group Explosion Strategy (IGES) for searching multiple targets using Swarm Robotics were proposed. In the Multi-target Search Problem with Environmental Restrictions in Swarm Robotics [19], three algorithms are compared (IGES, GES, and RPSO). But, their modeling of the target search problem is somehow different. In IGES, a target requires many iterations from a robot to be collected or the cooperation of multiple individuals for a reduced time.

Also, Shijie Lu and Yingguang Hao proposed the AO-PSO (Auxiliary Orientation in Particle Swarm Optimization) [2] approach, as a Swarm Robotic cooperative for the TDP. But, AO-PSO uses task assignment and its performance depends on the communication range. These details take their definition of TDP a bit away from what we are looking for.

Furthermore, all of the above algorithms still do not account for the dynamic evolution of targets' number. Also, their fitness function only considers the target's distance which does not correspond to the problem we are seeking to solve.

In this study, we tackled the COVID-19 propagation issue in the TDP's manner with an original approach based on metaheuristics. Our approach is compared to the A-RPSO and MFPSO algorithms, whose application is closest to that sought, and that have the best performance compared with other algorithms.

3 Modeling

As referred in [1], target detection problems consist in finding or detecting a target in a given environment. Depending on the models, the assumptions and the approaches, it may concern one or multiple targets and may be tackled with one or multiple sensors, either with mobile sensors where we refer to it as mobile search, or from fixed static sensors where it is known as static surveillance. TDP are commonly summarized as an environment with no map, where the unknown area is unstructured, complex, and possibly dangerous for humans' interventions. Targets must be detected and processed as quickly as possible, as timely intervention would result in better performances [20].

The eradication of COVID-19 spread in a context of containment is a particular TDP. We aim to search and find infected individuals in a bounded harsh environment and return a satisfiable path to get them to the hospital in order to take care of them and stop the pandemic. To formally define this problem for Swarm Robotics and adapt it to this particular issue, we made the following assumptions.

3.1 Assumptions about environment

The **search environment** is a grid with a square shape. As schematized in Fig. 1, it is a 2D grid-based representation composed of multiple zones. Each position of coordinate (x, y) has a unique value, positive values within the range $[0 - 1]$ to represent Containment Rate (CR), and negative values (-1) for obstacles. The environment can hold Robots, Targets, and Obstacles. We only consider Complex environments, which consist of a high density of obstacles.

A **zone** is a subarea of the environment where the position squares have one same value that corresponds to the local CR, its values can vary from 0 to 100%. The lower the Containment Rate, the greater the risk that the area will contain infected individuals.

A **Target** represents individuals likely to be infected. The estimated target positions can be defined by susceptible infected individuals by Contact Tracing methods, such as for each confirmed case of COVID-19 a certain number of potential targets are defined. We assume that a target is considered as an (x, y) stationary position of the environment. The number of targets grows depending on the virus spread speed and their positions are randomly generated.

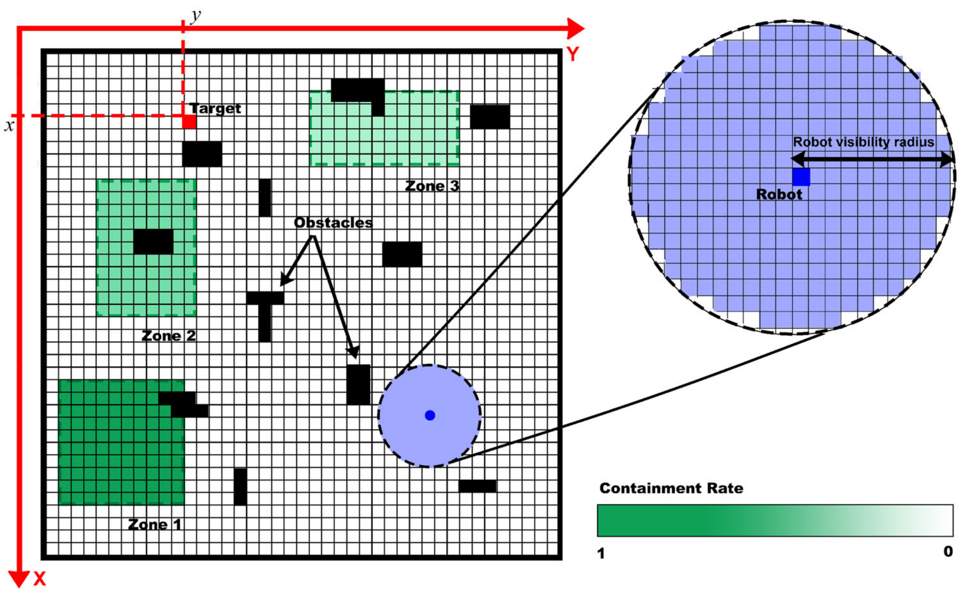
Obstacles are objects that obstruct the movement of robots, they block them from positioning themselves there or cross through. Obstacles are abstracted to rectangular shapes with variable dimensions (depending on the environment size), where all the obstacle area takes the value -1. Obstacles positions are randomly generated, they can overlap each other and form different shapes, but cannot be placed over the position of a target.

3.2 Assumptions about robots

Swarm robotics is constituted of mobile robots. Each robot is defined by its position and performance. It occupies a single position P defined by its coordinates (x, y) of the environment, abstracted to be a square of one unit side length. We assume that each robot can locate its own position relative to the environment in order to update it and share it with other robots according to the searching strategy.

The performance of each robot is evaluated via the **fitness function**. Its values can vary within the range $[0 - 1]$, the greater is the fitness value, the closer to a target the robot is, and when it is equal to 1 it means a target is found. Fitness function depends on both the Containment Rate of each zone

Fig. 1 Representation of an environment's grid with it different components



and the distance between the robot and a target. It is defined by Eq. (1).

$$\text{Fitness}(x, y) = 1 - \text{Min}\left(\frac{\text{CR}(x, y) + 2 * \frac{\text{Dist}(x,y)}{\text{diagSize}}}{3}, \frac{\text{Dist}(x, y)}{\text{diagSize})}\right) \quad (1)$$

where CR*(x,y)* is the Containment Rate of the robot's current position with CR in the range $[0 - 1]$, (x_t, y_t) position of the closest target, diagSize is the length of the environment's grid diagonal, and Dist*(x,y)* is the euclidean distance between a target and the current robot position, it's given by Eq. (2).

$$\text{Dist}(x, y) = \sqrt{(x - x_t)^2 + (y - y_t)^2} \quad (2)$$

Since the shape of the environment grids are squares, diagSize is equal to $\sqrt{2} *$ Size (environment side length).

Robots are characterized by simple computation capacity, reduced memory, power limitation, a limited velocity, and must provide an obstacle avoidance strategy. They totally ignore targets and obstacles positions. However, they are fitted with three kinds of sensors. The first one is used to evaluate the robot's performance, the second one for obstacle perception, and the third sensor is the biological COVID-19 test that allows determining if an individual is infected or not. These sensors have a restricted range of perception: the radius of fitness and obstacles perception range is set to 10 squares around each robot (see Fig. 1), while the biological test is limited to the current robot's position.

3.3 Dynamic environment simulating COVID-19 spread

In a realistic situation, all cities never apply the containment in the same way. This is what gives rise to the formation of zones with different rates of containment. Based on this description, random surfaces (zones) of the environment are generated and every zone is initialized to a random Containment Rate value within the range $[0 - 1]$. Also, the CR is not fixed for a given zone, it changes over the days. CR can increase, decrease, or fluctuate in a random way. This Containment Rate evolution is made via updating the CR interval boundaries by adding or subtracting 5% of the Containment's rate, as shown in Eq. (3).

$$[\text{Min}, \text{Max}] : \begin{cases} \text{Min} = \text{Min} + (5 * Day/100); \text{ if growing CR} \\ [\text{Min}, \text{Max}] \text{ no change; if random CR} \\ \text{Max} = \text{Max} - (5 * \text{Day}/100); \text{ if diminishing CR} \end{cases} \quad (3)$$

The impact of the Containment Rate (CR) on the Reproduction rate (R_{day}) for any day d, in a given zone z, can be expressed with Eq. (4).

$$R_{d,z} = R_0 * (1 - \text{CR}_{d,z}) \quad (4)$$

where R_0 is the estimated Reproduction rate without containment and $\text{CR}_{d,z}$ is the Containment Rate of the zone z in the day d. Thus, R is inversely proportional to CR.

R, in turn, impacts the number of COVID-19 new cases, so the number of targets is updated by Eq. (5).

$$\#Target_d = R_d * \#Target_{d-1} \quad (5)$$

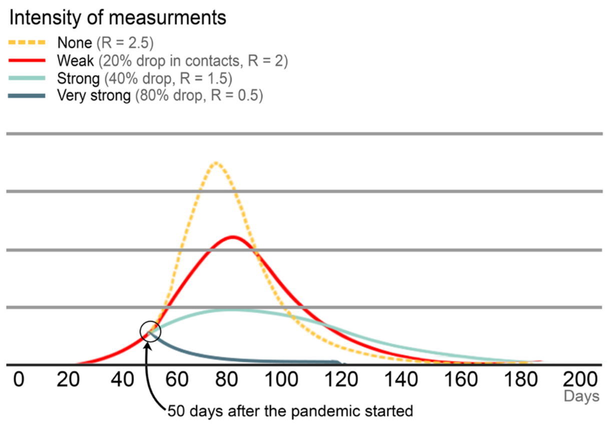


Fig. 2 Containment rate impact on the number of new cases [21]

where $\#Target_d$ and $\#Target_{d-1}$ are, respectively, the current and previous number of targets and R_d the current Reproduction rate.

By logical deduction, CR impacts the evolution of the number of new cases. As shown in Fig. 2 the greater is the Containment Rate, the slower the virus evolves, and the fewer individuals are infected.

According to this, the search space is constantly growing in an exponential way, at each instant d for each target the size of the search space is equal to Size^2. So, the size of the search space for all targets at instant d is Size$^2 * \#Target_d$. If we use Eq. (5) to calculate the size of the search space at successive times, we find Eq. (6).

$$\begin{aligned}
&t=0: \text{Size}^2 * \#Target_0\\
&t=1: \text{Size}^2 * \#Target_1 = \text{Size}^2 * (R_1 * \#Target_0)\\
&t=2: \text{Size}^2 * (R_2 * \#Target_1) = \text{Size}^2 * (R_2 * (R_1 * \#Target_0))\\
&\ldots\\
&t=d-1: \text{Size}^2 * (R_{d-1} * \#Target_{d-2})\\
&\quad = \text{Size}^2 * (R_{d-1} * (R_{d-2} \ldots R_2 * (R_1 * \#Target_0)..))\\
&t=d: \text{Size}^2 * (R_d * \#Target_{d-1})\\
&\quad = \text{Size}^2 * (R_d * (R_{d-1} * (R_{d-2} \ldots R_2 * (R_1 * \#Target_0)..)))
\end{aligned} \tag{6}$$

Then, according to Eq. (4), we know that R_t depends on the initial R_0. If we consider the worst case of 0% of Containment Rate, we obtain: $R_0 = R_1 = R_2 = \cdots = R_{d-1} = R_d$. And when we replace them in Eq. (6) it will turn into Eq. (7).

$$\text{Size}^2 * R_0^d * \#Target_0 \tag{7}$$

Here we conclude that the complexity in the worst case for this problem is exponential[2] and is given by Eq. (8).

$$\text{complexity}(d) = \text{Size}^2 * R_0^d * \#Target_0 = O(R_0^d) \tag{8}$$

3.4 A real case study: the case of the USA

In order to convert a real case study to a TDP, we need detailed data with positions of new COVID-19 cases, so we choose the data provided by the USA in [22]. This Dataset is updated every day with the newly registered cases and deaths, each line has the form <DATE, COUNTRY, STATE, FIPS, CASES, DEATHS>. In this study, we only need three of these information that are <DATE, FIPS, CASES>, where DATE is the date of registering the new cases. FIPS is the Federal Information Processing Standards, it represents a unique state-country codification, and CASES is the number of new cases registered in the day DATE. In what follow we excluded three countries namely: *Alaska*, *Hawaii* and *Puerto Rico* in order to work on a continuous search area. Below is presented an analysis of the concerned data.

Figure 3 illustrates boxplots of the daily new cases in the USA from the first COVID-19 case on 2020-01-21 to 2020-04-05. It shows that the median daily new cases (represented by a red line) is 11.5 and the mean (represented by a red point) is 4424.18. On most days the US registered between 0 and 1657.75 new cases, but there are some days where the number of new cases reached 34,742 new cases.

From the left boxplot of Fig. 3, we can notice that the outliers represent all the days where more than 4144.37 new cases have been registered. But, this threshold depends on the date we picked the data because this dataset is continuously updated. For these reasons, we decided to work with this dataset without replacing the outliers.

As shown in Fig. 4, the USA experienced an irregular increase of new cases of COVID-19 during the first 40 days. Then, the number of infected individuals has rapidly grown to hit a maximum of 34,742 new cases on 2020-04-04 and the total number of cases reached 336,238 cases on 2020-04-05.

We used a second dataset [23] to get the coordinate positions of every new case location. From the many available information in this dataset, we only kept <COUNTRY_FIPS, LAT, LNG>, where COUNTRY_FIPS corresponds to the FIPS of the first dataset, LAT and LNG are the latitude and longitude coordinates, respectively. Based on these two datasets, we obtained the needed information, such as <DATE, LAT, LNG, CASES>, for each new case and of each new day, we now have its real geographic position on the USA territorial.

Figure 5 depicts the geographical distribution of the Coronavirus cases in the USA territorial. The cases are represented

[2] when the value of R_0 is > 1.

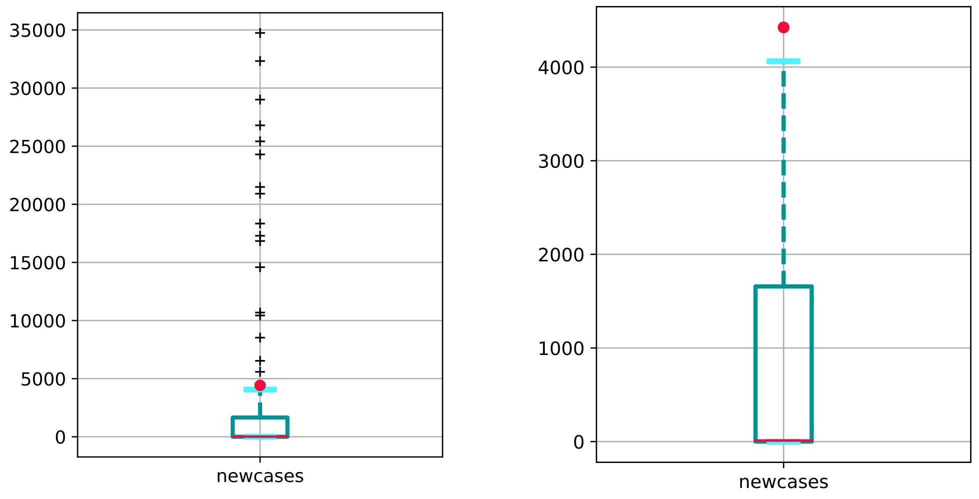


Fig. 3 Boxplot of new COVID-19 cases in the USA (Left: with visible outliers. Right: with hidden outliers)

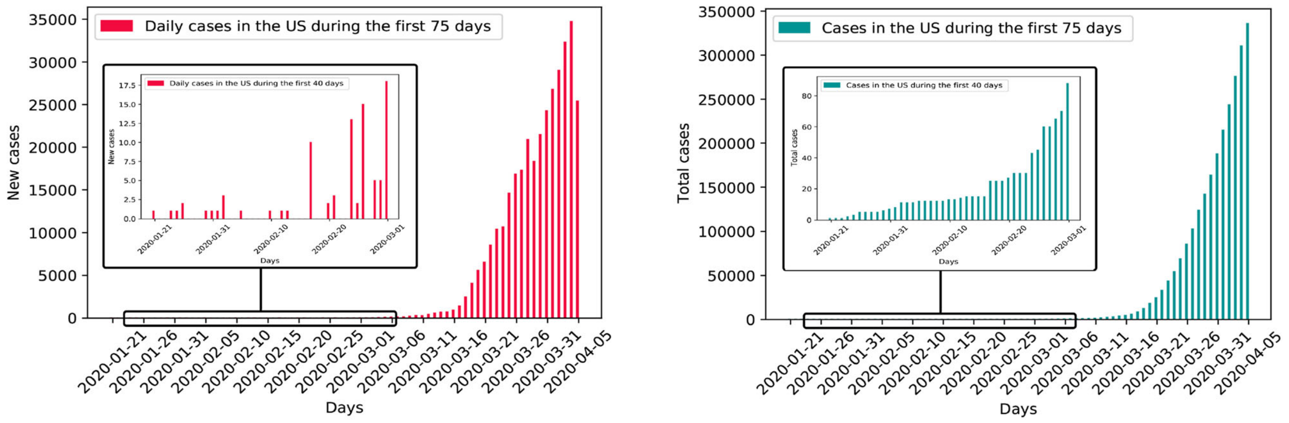


Fig. 4 Evolution of new and total Coronavirus cases per day in the USA

based on their coordinates on 2020-03-05, 2020-03-10, 2020-03-15, and 2020-03-20. We can remark the rapid evolution of the virus spread especially from 2020-03-01 to 2020-03-15 and then from 2020-03-15 to 2020-03-20. The number of cases in each city is somehow clustered and represented by a single point, and the color of each point indicates the number of cases in the concerned city.

In order to work on a more realistic situation, we opted for declustering the COVID-19 cases. Every single point that represents a certain number of cases in a specified day and city is flared into several points (targets). As shown in Fig. 6, from a single point we first define a radius of 20 squares around that initial point. Then, we randomly represent as many cases as necessary inside that radius.

Normalization of the coordinate positions to fit into a 5000×5000 squares representation requires the following steps. First, we get the USA Latitude [25, 50] and Longitude [−125, −65], to adapt them in a square environment. We choose the largest interval from the two (Longitude in this case) and define the representable USA surface in a grid with a square shape, with upper and lower positive values [MinLat, MaxLat] = [23, 88] and [MinLng, MaxLgn] = [55, 115]. Then, we define our target height and width boundaries as [min, max] = [0, 5000]. Finally, to get normalized coordinate according to our environment representation and axis orientation, we used Eq. (9) to define targets and borders positions.

$$\begin{cases} \text{NormalizedLat} = \dfrac{(90 - \text{originalLat} - \text{MinLat}) * (\max - \min)}{(\text{MaxLat} - \text{MinLat}) + \min} \\ \text{NormalizedLng} = \dfrac{(180 + \text{originalLng} - \text{MinLng}) * (\max - \min)}{(\text{MaxLng} - \text{MinLng}) + \min} \end{cases} \tag{9}$$

where 90 value refers to latitude max value that varies from 0° at the equator to 90° at the poles (−90° at the south pole and 90° at the north pole). And 180 refers to the longitude max value, which varies from 0° on the Greenwich meridian to 180° (−180° in the west and +180° in the east).

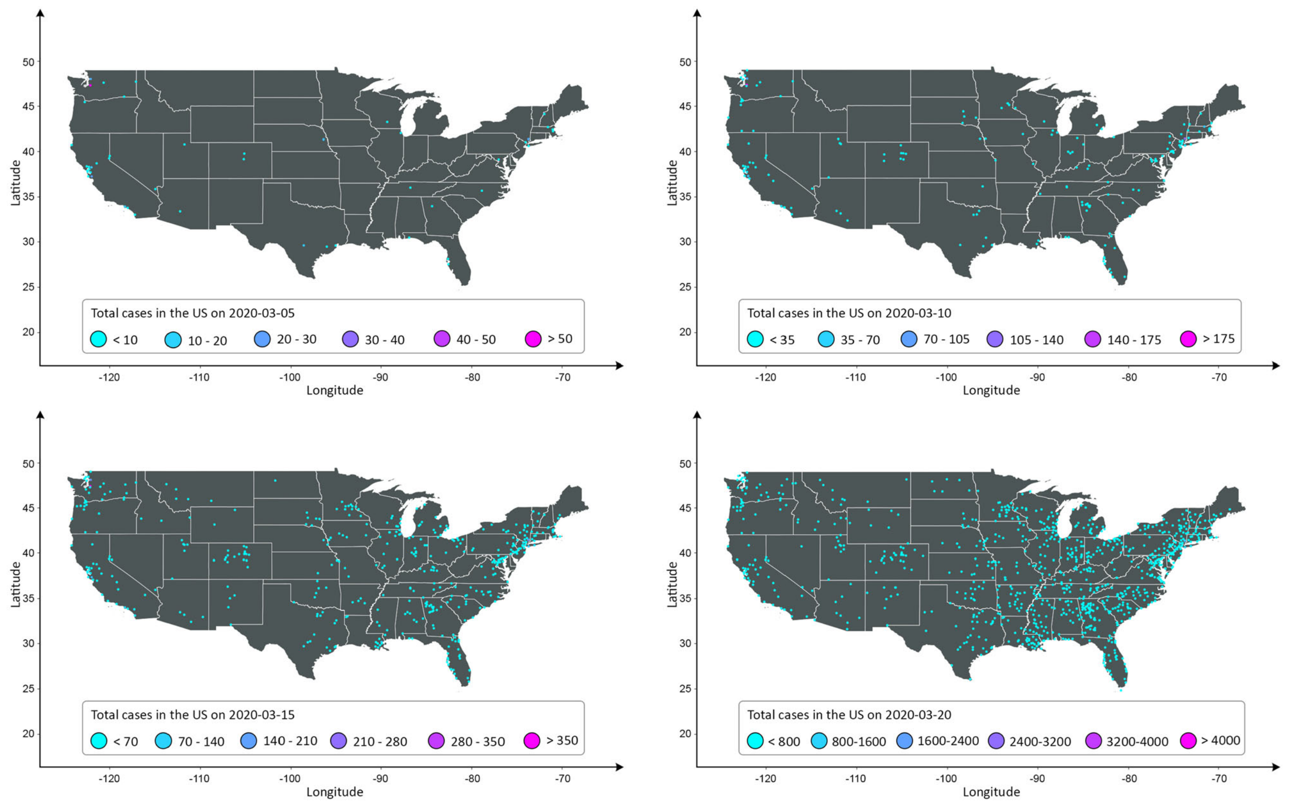


Fig. 5 Geographic distribution of Coronavirus cases in the USA at different times

Finally, Fig. 7 depicts the result[3] obtained from this conversion. The black rectangles are obstacles. The red circles represent some targets, and the blue ones illustrate robots with their range of perception. Also, the blue line is a robot's path solutions liking between a founded target and their clan's start position, which is colored in green. The environment on the Left represents a randomly generated environment with a 5000 × 5000 squares size, in which dashed green rectangles are used to represent zones with different Containment Rate. It should be noted that the zones are created one after the other. So, the CR of the most recent zone overwrites the small area where there is overlap. The environment on the right illustrates the real case study, the USA territorial is represented surrounded by obstacles to limit the search area.

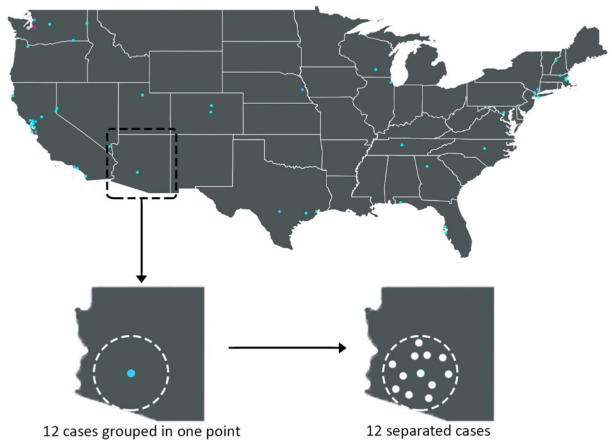


Fig. 6 Declustering of the represented Coronavirus cases in the USA

4 Robotic EHO (REHO) for eradicating COVID-19

4.1 Original EHO algorithm

Elephants Herding Optimization (EHO) [4,5] is a Swarm Intelligence based method inspired by the herding behavior of elephants and proposed to solve optimization problems. In nature, elephants are social animals that live in groups, each group is composed of several clans where each clan live under the leadership of a matriarch. Females live in family groups, while male elephants leave their group once they grow up. In EHO approach, the elephant population is composed of some clans, and each clan has a fixed number of elephants. A certain number of male elephants will leave their family group at each generation. Elephants of a clan live together

[3] The target and robot circles are enlarged to be more perceptible in the figure.

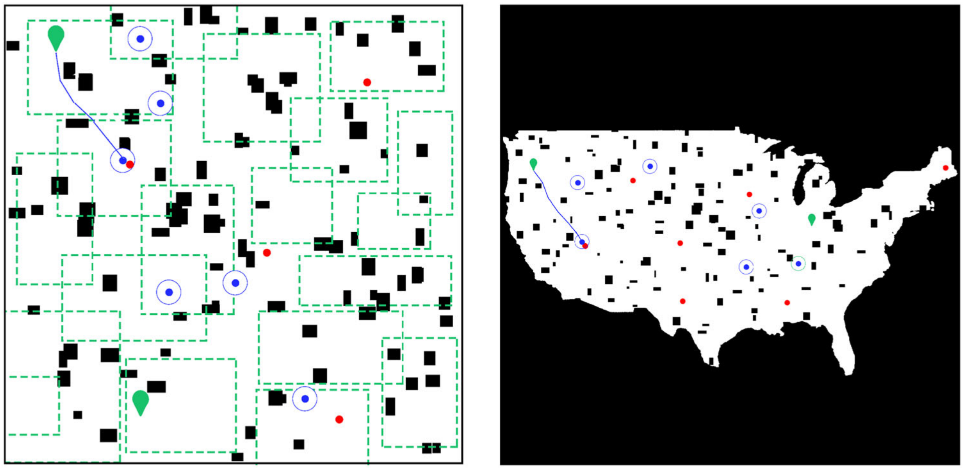


Fig. 7 Left: Random dataset. Right: USA data representation for the TDP

under the leadership of a matriarch. The matriarch of each clan is considered as the fittest elephant of its clan and the male elephants as the worst ones.

The EHO algorithm follows the below few steps:
Step 1 Initialize the generation counter $t = 1$; initialize the population; the empirical parameters: α, β, $\#Clans$, $\#Elephants$, and MaxGen.
Step 2 Sort all the elephants according to their fitness.
Step 3 Clan updating operator using Eq. (10) for all elephants in each clan.

$$X_{\text{new},i,j} = X_{c,i} + \alpha * (X_{\text{best},i} - X_{i,j}) * r \tag{10}$$

And for the fittest elephants (the matriarch) using Eqs. (11) and (12).

$$X_{\text{best},i} = \beta * X_{\text{center},i} \tag{11}$$

$$X_{\text{center},i} = \frac{1}{\#Elephants} * \sum X_{i,j} \tag{12}$$

Step 4 Separating operator to the worst elephant in each clan using Eq. (13).

$$X_{\text{Worst},i} = X_{\min} + (X_{\max} - X_{\min}) * \text{rand} \tag{13}$$

Step 5 Evaluate the new positions of the population; $t = t+1$.
Step 6 If $t == MaxGen$, return the best solution, else go to Step 2.
where:
α: the influence rate of the matriarch on elephants [0, 1].
β: the influence rate of the clan's gravity center on the matriarch [0, 1].
$\#Clans$, $\#Elephants$: respectively, the number of clans and number of elephants per clan.
MaxGen: the maximum generation.
C_i: the ith clan and $X_{\text{center},i}$: the gravity center of clan C_i.
$X_{i,j}$,$X_{\text{new},i,j}$: respectively, the actual and newly updated positions for elephant j in clan C_i.
$X_{\text{best},i}$,$X_{\text{worst},i}$: respectively, the best and the worst elephant in clan C_i.
$X_{\max}$,$X_{\min}$: respectively, the upper and lower bound of the position of elephant individual.
rand, r: random numbers between 0 and 1.

4.2 Proposed REHO algorithm

Robotics Elephant Herding Optimization (REHO) is an adapted version of the EHO Swarm intelligence-based approach on a Multi-Robot System, which become a Swarm Robotics approach. The main difference between REHO and EHO resides in the implementation of real-world scenarios with MRSs characteristics, where robots must have a range of perception and a maximum velocity limit, in addition to providing an obstacle avoidance and path planning strategies.

In this paper, we propose the REHO method for the particular targets' search problem. It consists of a number of $\#Clans$ clans (or groups) and each clan has $\#Elephants$ robots (elephants). These robots cooperate in the search space (environment) in the targets' search task. A target has a position (*x, y*) represented as a vector of 2 values. The objective of the REHO algorithm is to find positions of all targets (infected people) and return a short, safe and satisfying path from every target's position to the start position of clans (hospitals).

Robots' organization and positions initialization Clan positions are randomly initialized in the environment, it represents the start point of the clan which can be a hospital or a suitable place to receive infected individuals. Then, robots of the same clan are initialized in a region within a radius of 20 squares around the clan position.

Robots' positions update The method of calculating new robot positions in REHO is similar to that of EHO, it is computationally simple (only simple basic arithmetic operations), it needs a very small memory, and a reduced need for communication between robots. The positions being on 2 Dimensions, the equations are applied on the two components (respectively, x-coordinate and y-coordinate) of each position. Each robot calculates its next position by Eq. (14), which is originally based on Eq. (10):

$$\begin{cases} x_{\text{new},i,j} = x_{c,i} + \alpha * (x_{\text{best},i} - x_{i,j}) * r \\ y_{\text{new},i,j} = y_{c,i} + \alpha * (y_{\text{best},i} - y_{i,j}) * r \end{cases} \tag{14}$$

Similarly to Eq. (11), the fittest robot of each clan estimates its future position with Eq. (15).

$$\begin{cases} x_{\text{best},i,j} = \dfrac{\beta}{\#Elephants} * \sum x_{i,j} \\ y_{\text{best},i,j} = \dfrac{\beta}{\#Elephants} * \sum y_{i,j} \end{cases} \tag{15}$$

Also, in this TDP, the lower bound of position is (0, 0) and the upper one depends on the environment side length (namely: *Size*), i.e. the position (*Size-1, Size-1*). Thus, as in Eq. (13), the robot with the worst fitness calculates its next position within Eq. (16).

$$\begin{cases} x_{\text{worst},i} = (\text{Size} - 1) * rand \\ y_{\text{worst},i} = (\text{Size} - 1) * rand \end{cases} \tag{16}$$

Once new positions of robots are calculated, the robot navigation method (path planning method) is invoked in order to build a short and obstacle-free (safe) path from the actual robot position $(x_{i,j}, y_{i,j})$ to the new one $(x_{\text{new},i,j}, y_{\text{new},i,j})$. If the velocity between the two positions is greater than the robots' maximum velocity, the destination position is updated to satisfy this constraint. The evaluation of the new robots' positions take into account all the positions included in their range of perception.

Stop conditions To terminate the targets' search mission, there are three termination conditions:

- Maximum number of iterations reached: robots' battery capacity.
- Maximum number of new targets reached: abort the mission.
- All targets are found: success of the mission.

Algorithm The main steps of REHO are described in Algorithm 1, where $\#Clans * \#Elephants$ is the population size. $Fitness(x_{i,j}, y_{i,j})$ is the fitness function of the jth robot of the ith clan. *MaxTargets* is the estimated limit of targets' number that can be handled by the robots, and *MaxVelocity* is the maximum robots' velocity (the distance that a robot can cross in one iteration).

```
Input: MaxGen, #Clans, #Elephants, $\alpha$, $\beta$, $\#Target_0$,
       MaxTargets, MaxVelocity.
Output: PathList : List of paths from each target to robot's start
        position.
begin
    Initialization
    Initialize the generation counter t $\leftarrow 0$;
    Initialize the target counter n $\leftarrow 0$;
    Initialize the Clans' positions;
    Generate the robot's positions $(x_{i,j}, y_{i,j})_0$
    while t $< MaxGen$ and n $< \#Target_t$ do
        for $i \leftarrow 1$ to $\#Clans$ do
            Sort all the elephants according to their fitness.
            for $j \leftarrow 1$ to $\#Elephants$ do
                Calculate new elephant position
                $(x_{new,i,j}, y_{new,i,j})_t$ using Eq. (14), (15) and (16)
                if velocity > MaxVelocity then
                    adjustVelocity();
                robot Path Planning from $(x_{i,j}, y_{i,j})_t$ to
                $(x_{new,i,j}, y_{new,i,j})_t$ ;
                Evaluate the new position $(x_{new,i,j}, y_{new,i,j})_t$ ;
                if Fitness$(x_{new,i,j}, y_{new,i,j})_t$ == 1 then
                    n $\leftarrow$ n + 1;
                    PathList [n] $\leftarrow$ get Path from $(x_{i,j}, y_{i,j})_0$ to
                    $(x_{new,i,j}, y_{new,i,j})_t$ ;
        Update Containment Rate $CT_t$ of Environment;
        Update $\#Target_{t+1}$ using Eq. (5);
        if $\#Target_{t+1}$ > MaxTargets then
            abortMission();
        $(x_{i,j}, y_{i,j})_{t+1} \leftarrow (x_{new,i,j}, y_{new,i,j})_t$ ;
        t $\leftarrow$ t + 1;
    Return (PathList)
```

Algorithm 1: REHO for the TDP

4.3 Robots collision-free path planning

Basically, collision-free path planning means: (1) avoid collisions between obstacles and the robots, and (2) optimize the path according to predefined constraints like path length or smoothness [24]. Mobile robots gradually build the path between their current positions and their destination positions.

The literature shows a variety of approaches of obstacles avoidance strategies to solve the collision issue. Khatib was the first to propose a potential field method based on attractive force towards the goal and repulsive force of obstacles [25], Lumelsky and Stepanov presented two algorithms namely Bug1 and Bug2 exploiting touch sensors [26]. In [27] authors proposed an obstacle avoidance method called *Dynamic Window* that estimates the adapted linear and angular speed in the local view of the robot. The suggested method in [28] is a simple fuzzy logic controller approach that determines the robot's collision-free path. Input Space Sampling-based (ISS) planning for obstacle avoidance was first presented by the authors in [29], subsequently it was widely used and tested on real mobile robots [30,31].

Due to the proven efficacy, safety, rapidity, and adaptability of the Sampling-based method on autonomous mobile robots, we choose to adapt it for our robot path planning. Since we are dealing with discrete environments, we need to define the following concepts:

Direction The local space of a robot is divided into four zones of 90°, being North–East, North–West, South–East, and South–West. The robot's view field is an angle of 90° chosen according to the destination position P_{dest} relative to its current position P_{cur}.

Accessible positions Robot view is limited by the sensor's range of perception, thus accessible positions are the ones included within this range in the robot view's direction.

Admissible paths A path is admissible when it is obstacles-free, for that we exploited Bresenham's algorithm [32] to extract successive positions of a straight line and check if it contains obstacles. In principle, the path is constructed as a set of k waypoints and can be denoted as Path = $\text{wp}_1, \text{wp}_2, \ldots \text{wp}_k$, where wp_1 is equal to P_{cur} and wp_k to an admissible intermediate position P_t .

Optimal admissible path The criteria to choose the best path from the admissible ones is by calculating the distance remaining between an intermediate position P_t and the destination position P_{dest}, using the Euclidean distance.

Accordingly, the collision-free path planning algorithm can be resumed in Algorithm 2.

```
Input: $P_{cur}$ : robot's initial position, $P_{dest}$ : robot's destination
       position.
Output: GlobalPath : path from initial position to destination.
begin
    Initialization
    Initialize the step counter t ← 0;
    Initialize the intermediate position of the t$^{th}$ step $P_t \leftarrow P_{cur}$;
    Initialize the robot's perception range R;
    while $P_t <> P_{dest}$ and t $< MaxT$ do
        D ← Determine the direction to take;
        Positions ← get accessible positions in direction D
        within range R;
        SubPaths ← get admissible paths from $P_t$ to Positions;
        OptimalSubPath ← get optimal admissible path from
        SubPaths;
        GlobalPath ← GlobalPath ∪ OptimalSubPath;
        $P_{t+1}$ ← Last position of OptimalSubPath;
        t ← t + 1;
    Return GlobalPath
```

Algorithm 2: Input Space Sampling path planning

5 Experimental results and discussion

In order to validate the effectiveness and efficiency of our proposed REHO approach, we conduct a series of experiments. In these experiments we are dealing with complex environments with multiple targets and the maximum velocity of robots is set to 500. The three termination conditions of the search mission are:

1. Maximum number of generations: fail (*MaxGen* set to 1500 in this study).
2. Number of new targets is too large to be handled: abort the mission (*MaxTargets* set to 10,000 in this study).
3. All targets are reached: success.

In this Section, a description of the experiment's organization and the REHO's and comparative approaches parameter settings are presented, followed by the experimental results. These experiments are organized into two main sub-sections, the analysis of REHO performances is first presented. Then, the proposed REHO approach is compared with two other SI methods, namely A-RPSO and MFPSO. Both experiment subsections are composed of four parts. Part 1, studies the impact of the initial reproduction rate R_0 of the virus. The second part studies the influence of the initial number of targets $\#Target_0$ in the environment. Then, part 3 studies the robots' speed influence $IterDay$ on the search mission. Each of these three parts is experienced under three scenarios of the Containment Rate evolution, which are: growing Containment Rate, randomly changing Containment Rate, and diminishing Containment Rate. Furthermore, in REHO analysis robots' performances are compared in three environment sizes. Small environments with 500 × 500 squares, medium environments with 2500 × 2500 squares, and large environments of 5000 × 5000 squares. Finally, part 4 concerns an experiment on real data of COVID-19 provided by the USA. For each run, the number of iterations, the execution time (in seconds), and the success rate, which is the percentage of targets found, are recorded from what are calculated the mean and standard deviation (mean ± std).

5.1 Settings

Following the parameters tuning of the REHO approach that we obtained, its parameter values are set as: the factors α = 0.5, β = 0.6, the number of clans $\#Clans$ = 5, and the number of robots in each clan $\#Elephants$ = 4. Which corresponds to a population size of $5 \times 4 = 20$ robots. For the comparative methods, A-RPSO parameters are adapted from [3] with $c1$ = 2, the initial inertia weight (W_{ini}) = 1, α = 0.5, β = 0.7, and the number of robots (NR) = 20 robots. And MFPSO parameters as given in [17] where $c1$ = 2, $c2$ = 2, number of swarms $\#Swarms$ = 5, number of particles in each swarm $\#Particles$ = 4, which makes a population of 20 robots. Number of fruit flies per particle $\#FruitFlies$ = 10, fruit flies max velocity = 10, and fruit flies max iterations = 10.

These approaches as many other Swarm Intelligence-based methods are dependent on certain stochastic distribution. Thus, different runs will generate different results. In this work, many independent runs of the same dataset are implemented in order to get the most representative statistical results.

The dataset environments have been created according to the environment modeling described in Sect. 3, examples of test environments are presented in Fig. 7. The REHO approach, the ISS path planning method, the comparative approaches (A-RPSO and MFPSO), and the generation of the environments have all been implemented in *Java*.

5.2 Analysis of REHO algorithm

5.2.1 Influence of the initial reproduction rate: R_0

This part presents the experiment relatives to the initial reproduction rate R_0 . According to [33], the initial reproduction

Table 1 Average iterations number, average execution time, and success rate in complex environments with different Containment Rate updates facing the variation of initial reproduction rate (R0) values

$R0$	Small (500)			Medium (2500)			Large (5000)		
	Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
Growing CR									
1.5	12 ± 1.9	0.3 ± 0.1	100 ± 0.0	13 ± 2.0	1.2 ± 0.2	100 ± 0.0	14 ± 3.3	2.4±0.4	100 ±0.0
2	13 ± 2.3	0.3± 0.1	100 ± 0.0	14 ± 2.4	1.3 ± 0.2	100 ± 0.0	17 ± 4.6	2.8 ± 0.8	100 ± 0.0
2.5	13 ± 3.0	0.3 ± 0.1	100 ±0.0	14 ± 2.8	1.3 ± 0.2	100 ± 0.0	18 ± 5.4	2.9 ± 0.8	100 ± 0.0
3	13 ± 3.3	0.3 ± 0.1	100 ± 0.0	14 ± 2.0	1.5 ± 0.3	100 ± 0.0	18 ± 5.1	3.2 ± 1.5	100 ± 0.0
3.5	13 ± 3.3	0.3 ± 0.1	100 ± 0.0	16 ± 3.1	1.6 ± 0.4	100 ± 0.0	21 ± 8.0	4.0 ± 2.1	100 ± 0.0
4	14 ± 3.4	0.3 ± 0.1	100 ± 0.0	17 ± 3.8	1.6 ± 0.6	100 ± 0.0	22 ± 7.0	4.1 ± 2.9	100 ± 0.0
4.5	15 ± 3.7	0.3 ± 0.1	100 ± 0.0	18 ± 4.0	1.6 ± 0.5	100 ± 0.0	25 ± 7.7	4.1 ± 2.8	100 ± 0.0
5	18 ± 7.3	0.5 ± 0.5	100 ± 0.0	21 ± 6.0	1.9 ± 1.2	100 ± 0.0	44 ± 63.0	11.4 ± 31.1	100 ± 0.0
Random CR									
1.5	14 ± 3.1	0.3 ± 0.1	100 ± 0.0	17 ± 4.7	1.3 ± 0.2	100 ± 0.0	16 ± 5.8	2.5 ± 0.7	100 ± 0.0
2	13 ± 3.2	0.3 ± 0.1	100 ± 0.0	16 ± 4.0	1.3 ± 0.3	100 ± 0.0	17 ± 4.7	2.8 ± 0.8	100 ± 0.0
2.5	15 ± 4.5	0.3± 0.1	100 ± 0.0	17 ± 4.2	1.5 ± 0.3	100 ± 0.0	19 ± 4.2	3.1 ± 0.7	100 ± 0.0
3	15 ± 4.5	0.3 ± 0.1	100 ± 0.0	19 ± 5.1	1.7 ± 0.5	100 ± 0.0	19 ± 6.7	3.4 ± 1.7	100 ± 0.0
3.5	16 ± 5.2	0.5 ± 1.1	100 ± 0.0	21 ± 6.8	1.9 ± 0.8	100 ± 0.0	60 ± 230.8	6.0 ± 10.7	97 ± 0.1
4	17 ± 4.5	0.4 ± 0.1	100 ± 0.0	111 ± 351.0	4.7 ± 9.7	94 ± 0.2	287 ± 568.2	10.9 ± 18.5	87 ± 0.3
4.5	77 ± 290.6	0.6 ± 1.1	96 ± 0.2	201 ± 480.0	6.7 ± 11.5	88 ± 0.3	465 ± 677.4	20.3 ± 24.8	70 ± 0.4
5	195 ± 482.0	1.0 ± 1.8	88 ± 0.3	497 ± 688.4	12.1 ± 13.8	69 ± 0.5	659 ± 732.4	23.3 ± 24.1	60 ± 0.5
Diminishing CR									
1.5	17 ± 6.5	0.3 ± 0.1	100 ± 0.0	24 ± 12.7	1.7 ± 0.6	100 ± 0.0	92 ± 290.4	7.3 ± 17.9	96 ± 0.2
2	57 ± 217.0	0.6 ± 1.4	98 ± 0.1	85 ± 289.2	3.3 ± 7.7	96 ± 0.2	175 ± 442.1	9.7 ± 17.4	90 ± 0.3
2.5	52 ± 207.2	0.5 ± 0.9	98 ± 0.1	298 ± 563.7	8.2 ± 12.3	82 ± 0.4	675 ± 731.5	26.1 ± 25.7	57 ± 0.5
3	317 ± 591.4	1.4 ± 2.0	80 ± 0.4	793 ± 736.1	14.8 ± 12.3	49 ± 0.5	939 ± 716.6	29.6 ± 20.9	39 ± 0.5
3.5	407 ± 647.7	1.6 ± 2.0	74 ± 0.4	882 ± 726.7	19.4 ± 14.9	43 ± 0.5	1086 ± 663.2	31.0 ± 17.8	29 ± 0.4
4	612 ± 725.1	2.0 ± 2.0	60 ± 0.5	1088 ± 660.8	17.9 ± 10.0	29 ± 0.4	1235 ± 565.3	31.8 ± 13.4	19 ± 0.4
4.5	851 ± 732.0	2.5 ± 1.9	45 ± 0.5	1265 ± 538.1	18.5 ± 7.2	17 ± 0.4	1442 ± 285.5	34.9 ± 6.9	5 ± 0.2
5	878 ± 730.6	2.5 ± 1.8	42 ± 0.5	1322 ± 482.2	17.8 ± 6.3	13 ± 0.3	1441 ± 291.5	33.0 ± 7.0	5 ± 0.2

rate R_0 depends on several parameters and varies according to the countries, it can reach up to a value of $5.03 \sim 5$. Also the value of R_0 becomes significant when it is > 1, this is why we are going to carry out our experiments by varying the value of R_0 considering the following set $\{1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$, in order to test our REHO limits. Then, the robots' speed needs to be fixed, we choose a slow speed to effectively study the influence of R_0 on the search mission, so we put *IterDay* equal to 5, and we set the initial number of targets to 50 targets. We experimented this parameter with a 50 times execution for every value of R_0 . Thereby, we performed 400 executions to study the impact of this parameter on the robots' search for each environment size and each Containment Rate evolution. The obtained results for the different behavior types of the Containment Rate are shown in Table 1.

First, we will discuss the numerical results obtained when the environment's Containment Rate is growing. It can be observed in Table 1 that 100% of the targets are reached no matter the reproduction rate R_0 and the environment size. These outcomes can be directly linked to the environment growing CR that slows the virus spread and therefore facilitate the robots' search mission. The average number of iteration smoothly increases with the increasing value of R_0 , it reaches 18 iterations for the maximum R_0 value in small environments, whereas in medium and large environments it performs 20 and 43 iterations, respectively. The average execution time follows the same soft increase with a maximum of 0.45 s in small environments, 1.93 and 11.39 s in medium and large ones, respectively.

We now pass to the random update Containment Rate environments, the visual representation of the average success rate and iterations evolution are represented in Figs. 8 and 9. It can be observed that for all environment sizes, the success rate starts decreasing and the iteration number visibly rising at a certain R_0 value, at 3.5 for the large environments, and at 4 and 4.5 for medium and small ones, respectively. The average iterations number attains a maximum of 194 iterations in small environments, where it reaches 496 and 659 iterations in medium and large environments, respectively.

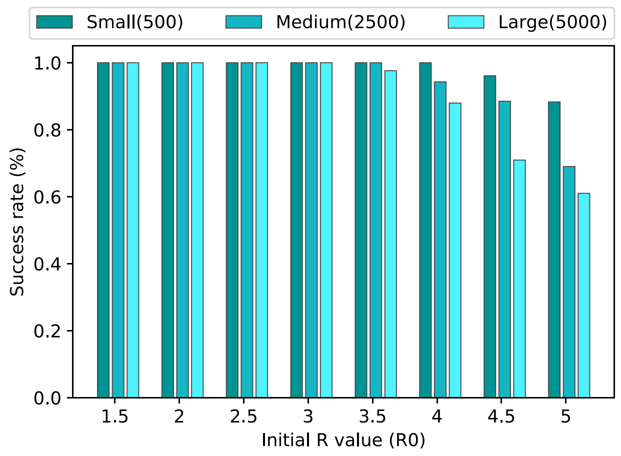


Fig. 8 Comparing the success rate in different environment sizes with increasing R_0 in a random CR update

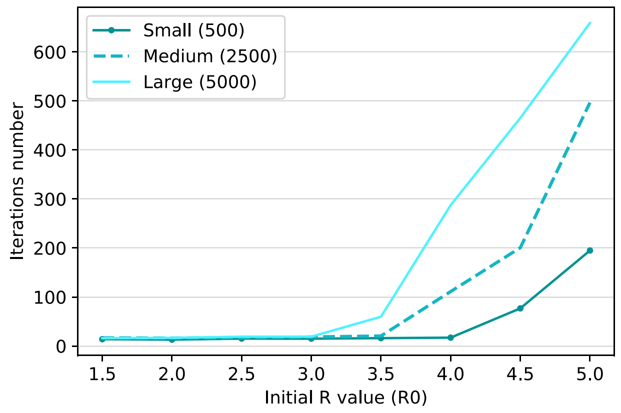


Fig. 9 Comparing the iterations number in different environment sizes with increasing R_0 in a random CR update

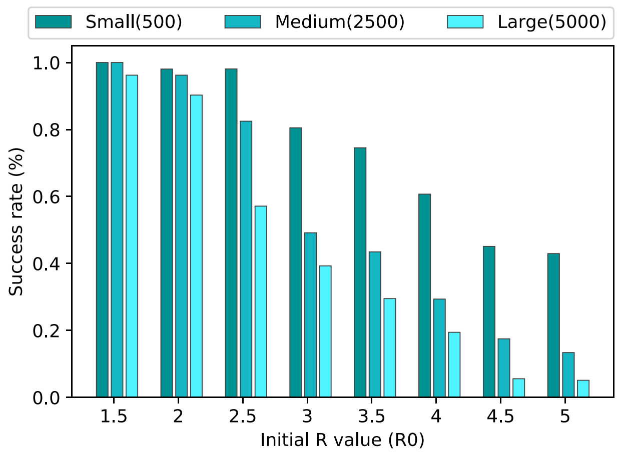


Fig. 10 Comparing the success rate in different environment sizes with increasing R_0 in a diminishing CR

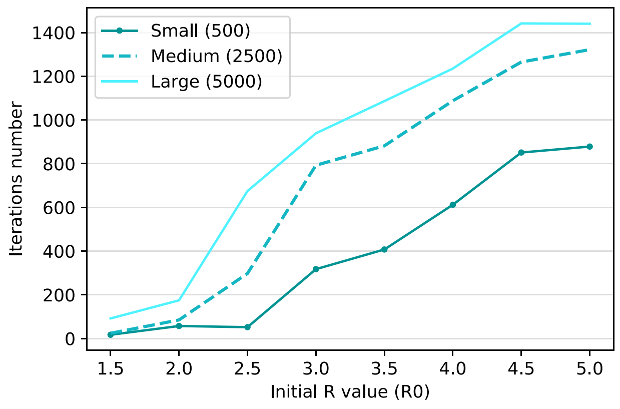


Fig. 11 Comparing the iterations number in different environment sizes with increasing R_0 in a diminishing CR

Table 2 Average iterations number, average execution time, and success rate in complex environments with different Containment Rate updates facing the variation of initial targets' number ($\#Target_0$)

$\#Target_0$	Small (500)			Medium (2500)			Large (5000)		
	Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
Growing CR									
5	3 ± 1.1	0.0 ± 0.0	100 ± 0.0	4 ± 1.8	0.1 ± 0.0	100 ± 0.0	3 ± 1.7	0.1 ± 0.1	100 ± 0.0
50	14 ± 5.7	0.2 ± 0.1	100 ± 0.0	19 ± 8.1	1.3 ± 0.3	100 ± 0.0	14 ± 3.9	2.3 ± 0.4	100 ± 0.0
100	20 ± 4.7	0.7 ± 0.1	100 ± 0.0	34 ± 10.8	4.6 ± 0.9	100 ± 0.0	34 ± 12.5	8.8 ± 1.3	100 ± 0.0
150	27 ± 6.0	1.3 ± 0.1	100 ± 0.0	38 ± 9.2	8.5 ± 1.3	100 ± 0.0	40 ± 19.9	16.0 ± 2.9	100 ± 0.0
200	30 ± 3.9	2.2 ± 0.2	100 ± 0.0	44 ± 7.7	13.0 ± 1.1	100 ± 0.0	48 ± 14.0	28.6 ± 5.3	100 ± 0.0
250	40 ± 5.9	3.5 ± 0.4	100 ± 0.0	51 ± 8.8	19.3 ± 1.8	100 ± 0.0	61 ± 33.9	40.6 ± 8.0	100 ± 0.0
300	47 ± 7.3	4.8 ± 0.5	100 ± 0.0	64 ± 8.1	29.5 ± 2.8	100 ± 0.0	75 ± 16.8	65.2 ± 19.5	100 ± 0.0
350	54 ± 6.1	6.6 ± 0.8	100 ± 0.0	72 ± 11.6	40.0 ± 6.6	100 ± 0.0	81 ± 17.7	85.9 ± 20.0	100 ± 0.0
400	62 ± 7.1	8.5 ± 1.0	100 ± 0.0	87 ± 11.8	61.5 ± 11.5	100 ± 0.0	104 ± 22.4	135.1 ± 44.2	100 ± 0.0
500	91 ± 13.7	15.6 ± 3.0	100 ± 0.0	125 ± 22.3	108.7 ± 28.0	100 ± 0.0	179 ± 61.6	318.1 ± 136.4	100 ± 0.0
Random CR									
5	3 ± 0.9	0.0 ± 0.0	100 ± 0.0	4 ± 2.6	0.1 ± 0.0	100 ± 0.0	3 ± 1.4	0.1 ± 0.0	100 ± 0.0
50	13 ± 3.9	0.2 ± 0.1	100 ± 0.0	21 ± 8.0	1.5 ± 0.3	100 ± 0.0	13 ± 3.4	2.1 ± 0.3	100 ± 0.0
100	19 ± 3.6	0.7 ± 0.1	100 ± 0.0	33 ± 10.5	4.5 ± 0.9	100 ± 0.0	32 ± 9.2	8.9 ± 1.4	100 ± 0.0
150	29 ± 6.9	1.4 ± 0.2	100 ± 0.0	36 ± 7.0	8.5 ± 1.1	100 ± 0.0	39 ± 13.6	16.1 ± 2.8	100 ± 0.0
200	30 ± 5.0	2.2 ± 0.2	100 ± 0.0	45 ± 8.5	13.1 ± 1.3	100 ± 0.0	50 ± 17.5	28.9 ± 6.6	100 ± 0.0
250	41 ± 6.3	3.6 ± 0.4	100 ± 0.0	52 ± 9.0	19.6 ± 2.0	100 ± 0.0	57 ± 12.1	39.5 ± 6.3	100 ± 0.0
300	46 ± 7.6	4.7 ± 0.5	100 ± 0.0	66 ± 9.5	31.9 ± 4.2	100 ± 0.0	85 ± 24.5	68.7 ± 23.5	100 ± 0.0
350	56 ± 6.0	6.9 ± 0.7	100 ± 0.0	78 ± 14.0	43.4 ± 9.6	100 ± 0.0	96 ± 28.2	102.8 ± 37.5	100 ± 0.0
400	67 ± 7.9	9.0 ± 1.0	100 ± 0.0	105 ± 21.9	72.3 ± 19.2	100 ± 0.0	297 ± 457.5	199.6 ± 132.2	89 ± 0.3
500	170 ± 272.8	22.0 ± 8.8	96 ± 0.2	1158 ± 577.8	230.2 ± 66.1	37 ± 0.4	1467 ± 207.0	458.0 ± 78.9	17 ± 0.1
Diminishing CR									
5	3 ± 1.9	0.0 ± 0.0	100 ± 0.0	3 ± 1.7	0.1 ± 0.0	100 ± 0.0	4 ± 2.3	0.1 ± 0.1	100 ± 0.0
50	13 ± 5.0	0.2 ± 0.1	100 ± 0.0	22 ± 11.5	1.5 ± 0.4	100 ± 0.0	14 ± 4.8	2.3 ± 0.3	100 ± 0.0
100	19 ± 4.2	0.7 ± 0.1	100 ± 0.0	44 ± 23.3	4.9 ± 1.2	100 ± 0.0	69 ± 95.3	9.8 ± 4.4	100 ± 0.0
150	30 ± 7.4	1.4 ± 0.1	100 ± 0.0	44 ± 13.0	9.3 ± 1.6	100 ± 0.0	54 ± 44.6	16.7 ± 4.1	100 ± 0.0
200	32 ± 7.3	2.2 ± 0.2	100 ± 0.0	57 ± 15.0	14.1 ± 1.5	100 ± 0.0	64 ± 28.0	30.6 ± 7.4	100 ± 0.0
250	43 ± 8.4	3.5 ± 0.4	100 ± 0.0	72 ± 26.3	58.6 ± 23.2	100 ± 0.0	155 ± 193.8	48.7 ± 14.3	100 ± 0.0
300	55 ± 14.3	5.1 ± 0.7	100 ± 0.0	142 ± 53.9	23.2 ± 5.2	100 ± 0.0	451 ± 590.8	156.6 ± 140.5	77 ± 0.4
350	69 ± 10.7	7.7 ± 1.0	100 ± 0.0	191 ± 269.4	49.1 ± 33.9	96 ± 0.2	762 ± 659.3	263.5 ± 162.7	60 ± 0.4
400	97 ± 19.7	11.7 ± 2.2	100 ± 0.0	879 ± 674.2	141.5 ± 78.0	51 ± 0.4	1422 ± 309.4	347.5 ± 101.6	14 ± 0.2
500	631 ± 652.5	32.6 ± 16.2	68 ± 0.4	1313 ± 463.7	165.5 ± 57.5	21 ± 0.3	1500 ± 0.0	247.4 ± 5.7	7 ± 0.0

Average execution time also knows a small increase when dealing with greater R_0 values. So, robots start having difficulties when random CR update is combined with big R_0 values. The virus spread becomes somehow faster but not totally uncontrollable.

For the worst possible Containment Rate evolution "diminishing CR", we got the obtained results presented in Figs. 10 and 11. It appears that the success chances of robots are widely reduced by the increasing Reproduction rate in a diminishing Containment Rate because the number of new targets expands at a high speed. Especially, for medium and large environments. The robots' efforts and search time increase in consequence and tend to 878 iterations for small environments, and 1321 and 1440 iterations for medium and large environments, respectively.

Considering the presented results, we remark that the initial reproduction rate R_0 essentially affects the robot search in lack of containment respect (Random and diminishing CR). Also, we deduce that the greater the R_0, the quicker the virus spread and increase the number of infected individuals, thus the smallest is the warranty to eradicate the virus spread.

5.2.2 Influence of the Initial targets' number: $\#Target_0$

In the report [34], the Initial value of R_0 is estimated to $[2 - 2.5]$ in China. According to [21], in the case of several European countries, the value of R_0 varies between 2.39 and 2.58. Considering these statistics, we set the value of R_0 to 2.5 for the experiments relating to the initial number of affected individuals and that relative to the speed of the robots presented in Sect. 5.2.3. We also, fix the Robots' speed *IterDay* to 25 iterations per day (25 iters/day), and we study REHO's performance in the face of the growing number of initial number of targets "$\#Target_0$". For that, we choose the following numbers: {5, 50, 100, 150, 200, 250, 300, 350, 400, 500}. We performed a total of 500 executions to draw the graphics relative to these experiments, 50 executions for each value of $\#Target_0$, and for each Containment Rate types update, distributed on 5 different datasets. Table 2 below presents the numerical results of this part.

In Table 2 are shown that for the growing Containment Rate evolution, REHO provides 100% of success rate, finding all the targets, even with 500 initial targets' number. This can be explained by the slowed targets' number growth due to the high Containment Rate. However, the average number of iterations and execution time are growing to reach a maximum of 90 iterations in 15 s within small environment, whereas it performs 125 iterations in 108 s and 178 iterations in 318 s in medium and large environments, respectively, and this is only related to the search space size.

Now, if we turn to the results obtained in the randomly updated Containment Rate. We found that as illustrated in Figs. 12 and 13, the success rate is only affected when start-

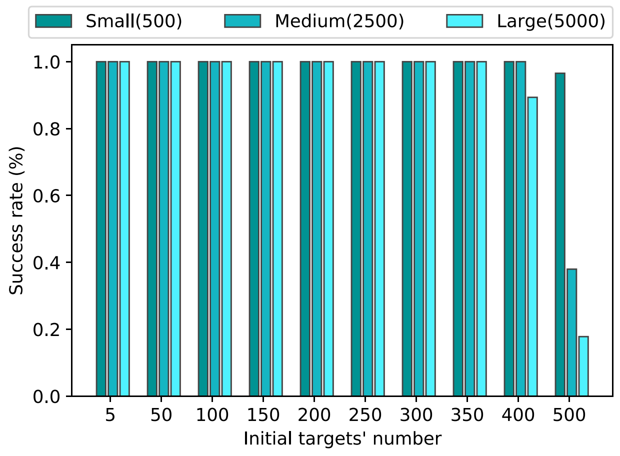


Fig. 12 Comparing the success rate in different environment sizes with increasing $\#Target_0$ in a random CR update

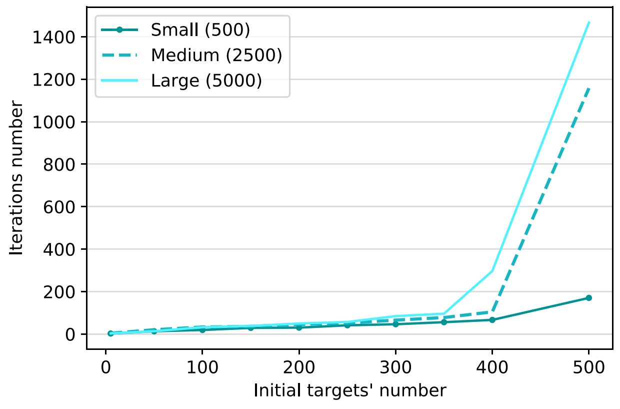


Fig. 13 Comparing the iterations number in different environment sizes with increasing $\#Target_0$ in a random CR update

ing with a large amount of initial targets. i.e., from 400 targets in large environment and 500 targets in small and medium ones. Whereas, the average number of iterations and execution time growth more significantly than in the previous part (growing CR). It executes a maximum of 170 iterations in 22 s within small environments, and around 1157 iterations in 230 s and 1466 iterations in 458 s when dealing with 500 initial targets' number, in medium and large environments, respectively. These outcomes are justified by the additional efforts that robots have to accomplish to stop the spread of the virus in a bigger search space with average containment respect.

Lastly, Figs. 14 and 15 show the success rate and average iterations number achieved by the Swarm of Robots, during the search mission in a diminishing Containment Rate. We notice that small environments encounter some difficulties when the initial number of targets is set to 500, while in medium and large ones it starts when dealing with 350 and 300 initial number of targets, respectively. These difficulties are expressed by the decreasing success rate at finding all

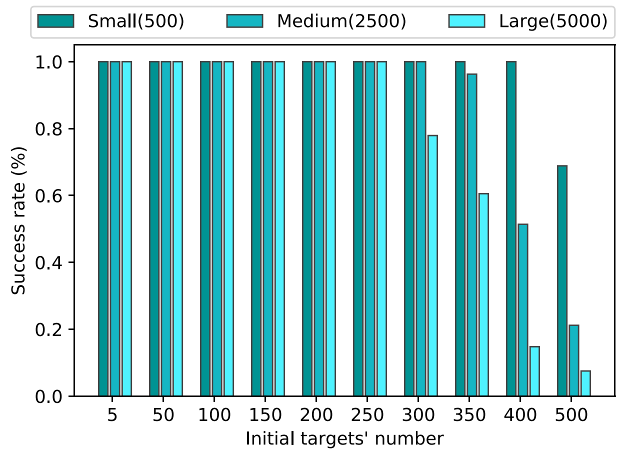


Fig. 14 Comparing the success rate in different environment sizes with increasing $\#Target_0$ in a diminishing CR

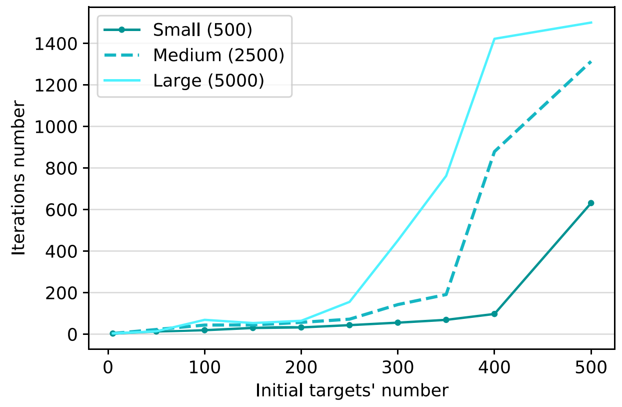


Fig. 15 Comparing the iterations number in different environment sizes with increasing $\#Target_0$ in a diminishing CR

the targets to 68%, 21% and 7% in small, medium and large environments, respectively, and the increasing efforts made by executing up to 631 iterations in 32 s, 1313 iterations in 165 s and 1500 iterations in 247 s within small, medium and large environments, respectively.

Another important observation can be made by comparing the average execution time between Random CR and diminishing CR. Such as in this diminishing CR search seems to take less time, the reason is that the decrease of the CR with big initial targets' number generates such a large growth in the number of targets that robots are forced to abort the search mission.

To summarize, the initial number of targets $\#Target_0$ affect the Target search mission of the swarm of robots, but only reduce its efficacy in the worst scenario. In other words, if this robotic approach is used too late, the virus already infected too many individuals, and the containment still not well applied (random and diminishing CR), the proposed REHO approach has more difficulties in achieving its mission.

5.2.3 Influence of the robots' speed: IterDay

Robot's speed represents the speed of robots in terms of iterations per day. Every single robot accomplishes a number of *IterDay* iterations in 1 day, and every *IterDay* (i.e. every day) the effective reproduction rate R_{day} is updated according to the Containment Rate of its zone. To study the impact of this parameter on the REHO approach, we have to fix the other parameters such as the initial targets' number ($\#Target_0$) to 50 targets, and the initial reproduction rate (R_0) to 2.5. The values of *IterDay* that we selected are the following: 1, 2, 5, 10, 15, 20, 25, 30, 35, 40. As for the study of the impact of $\#Target_0$ parameter, for the *IterDay* parameter, we achieved 500 executions to obtain the results presented in Table 3.

As seen in the first third of Table 3, our REHO approach manages to reach all of the targets when the Containment Rate is increasing (success rate equal to 100%). However, we notice that with very slow robots' speed, it takes more effort and more time to finish the search mission. Such as when it is at 1 iteration/day REHO executes 49 iterations in 17 s in small environments. While, it takes 53 iterations in 35 s and 107 iterations in 177 s within medium and large environments, respectively.

Considering the second third of Table 3, we are able to draw Figs. 16 and 17 that are concerning the random Containment Rate update. As we can see, when robots accomplish less than 5 iterations per day, they are too slow and almost automatically lead to failing at the search mission no matter the environment size. Contrariwise, from a speed of 5 iterations per day and more, robots become sufficiently rapid to find out all the targets with very few efforts, and in a very reasonable time.

Finally, we are going to discuss the achieved results when the Containment Rate diminishes every day until it attains 0%, where the effective reproduction rate R_{day} becomes equal to the initial one $R_0 = 2.5$. These results are graphically exposed in Figs. 18 and 19 below. In a similar way to the previously examined results, robots are not able to finish the search mission with a speed of fewer than 5 iterations per day. However, when the Containment Rate is going down, robots still cannot warranty 100% of success rate until their speed attains 10 iterations per day and more. Also, the iterations number and execution time decrease with the raising robots' speed. It registers 12 iterations in less than 1 s in small environments, and an average of 21 iterations in 1 s and 14 iterations in 2 s within medium and large environments, respectively.

The discussed experiments about the robots' speed *IterDay* show that the robots' speed has to be adapted to the virus spread speed. As we saw, when containment measures are not respected the virus spread more quickly and leads to more and more infected individuals that robots are not able to handle with a slow action speed.

Table 3 Average iterations number, average execution time, and success rate in complex environments with different Containment Rate updates facing the variation of robots' speed (iter/day)

IterDay	Small (500)			Medium (2500)			Large (5000)		
	Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
Growing CR									
1	49 ± 20.7	7.1 ± 5.0	100 ± 0.0	54 ± 18.6	35.3 ± 20.7	100 ± 0.0	108 ± 48.5	177.1 ± 103.4	100 ± 0.0
5	24 ± 10.0	1.2 ± 1.5	100 ± 0.0	33 ± 12.7	11.1 ± 11.5	100 ± 0.0	62 ± 28.0	72.0 ± 57.5	100 ± 0.0
10	13 ± 2.9	0.3 ± 0.1	100 ± 0.0	20 ± 6.4	1.8 ± 0.7	100 ± 0.0	21 ± 5.3	3.0 ± 1.1	100 ± 0.0
20	12 ± 2.6	0.2 ± 0.0	100 ± 0.0	16 ± 5.3	1.3 ± 0.2	100 ± 0.0	17 ± 6.6	2.3 ± 0.4	100 ± 0.0
35	13 ± 3.8	0.2 ± 0.0	100 ± 0.0	14 ± 4.3	1.2 ± 0.2	100 ± 0.0	16 ± 4.2	2.2 ± 0.3	100 ± 0.0
50	13 ± 3.5	0.2 ± 0.0	100 ± 0.0	15 ± 3.7	1.2 ± 0.2	100 ± 0.0	15 ± 4.9	2.1 ± 0.3	100 ± 0.0
100	13 ± 3.8	0.2 ± 0.0	100 ± 0.0	14 ± 2.9	1.3 ± 0.2	100 ± 0.0	17 ± 4.1	2.1 ± 0.3	100 ± 0.0
150	13 ± 4.5	0.3 ± 0.1	100 ± 0.0	15 ± 3.6	1.3 ± 0.2	100 ± 0.0	17 ± 4.9	2.2 ±0.3	100 ± 0.0
200	12 ± 3.0	0.3 ± 0.1	100 ± 0.0	14 ± 3.5	1.3 ± 0.2	100 ± 0.0	15 ± 3.7	2.2 ± 0.3	100 ± 0.0
300	14 ± 4.8	0.2 ± 0.0	100 ± 0.0	15 ± 5.7	1.2 ± 0.2	100 ± 0.0	17 ± 6.4	2.1 ± 0.3	100 ± 0.0
Random CR									
1	1500 ± 0.0	1.4 ± 0.2	0 ± 0.0	1500 ± 0.0	6.6 ± 0.6	0 ± 0.0	1500 ± 0.0	12.4 ± 1.1	0 ± 0.0
5	1500 ± 0.0	3.1 ± 0.3	1 ± 0.0	1500 ± 0.0	14.1 ± 1.2	1 ± 0.0	1500 ± 0.0	27.3 ± 2.0	1 ± 0.0
10	15 ± 4.1	0.3 ± 0.1	100 ± 0.0	21 ± 6.3	2.0 ± 0.9	100 ± 0.0	18 ± 4.3	2.9 ± 0.8	100 ± 0.0
20	12 ± 3.4	0.2 ± 0.1	100 ± 0.0	18 ± 4.7	1.4 ± 0.3	100 ± 0.0	15 ± 5.4	2.2 ± 0.4	100 ± 0.0
35	13 ± 2.6	0.2 ± 0.0	100 ± 0.0	18 ± 6.3	1.4 ± 0.3	100 ± 0.0	13 ± 3.8	2.2 ± 0.3	100 ± 0.0
50	13 ± 3.3	0.2 ± 0.0	100 ± 0.0	19 ± 7.4	1.4 ± 0.3	100 ± 0.0	14 ± 4.6	2.1 ± 0.3	100 ± 0.0
100	13 ± 3.2	0.2 ± 0.0	100 ± 0.0	20 ± 7.6	1.4 ± 0.3	100 ± 0.0	13 ± 3.9	2.1 ± 0.3	100 ± 0.0
150	14 ± 4.5	0.3 ± 0.0	100 ± 0.0	22 ± 9.0	1.4 ± 0.3	100 ± 0.0	15 ± 4.2	2.2 ± 0.2	100 ± 0.0
200	12 ± 2.9	0.2 ± 0.0	100 ± 0.0	21 ± 9.6	1.4 ± 0.3	100 ± 0.0	14 ± 5.5	2.2 ± 0.3	100 ± 0.0
300	13 ± 4.0	0.2 ± 0.0	100 ± 0.0	22 ± 11.0	1.4 ± 0.3	100 ± 0.0	15 ± 6.1	2.2 ±0.3	100 ± 0.0
Diminishing CR									
1	1500 ± 0.0	1.0 ± 0.1	0 ± 0.0	1500 ± 0.0	4.7 ± 0.4	0 ± 0.0	1500 ± 0.0	8.8 ± 0.7	0 ± 0.0
5	1500 ± 0.0	2.0 ± 0.2	1 ± 0.0	1500 ± 0.0	9.4 ± 0.9	0 ± 0.0	1500 ± 0.0	18.3 ± 1.6	0 ± 0.0
10	79 ± 290.2	0.6 ± 1.2	96 ± 0.2	265 ± 539.2	7.3 ± 11.9	84 ± 0.4	586 ± 715.5	24.1 ± 26.9	62 ± 0.5
20	14 ± 6.0	0.3 ± 0.1	100 ± 0.0	28 ± 21.7	1.8 ± 0.9	100 ± 0.0	17 ± 10.5	2.5 ± 0.6	100 ± 0.0
35	12 ± 2.7	0.2 ± 0.0	100 ± 0.0	24 ± 13.7	1.6 ± 0.4	100 ± 0.0	13 ± 4.9	2.3 ± 0.4	100 ± 0.0
50	13 ± 5.4	0.2 ± 0.0	100 ± 0.0	23 ± 11.6	1.5 ± 0.3	100 ± 0.0	15 ± 6.2	2.5 ± 0.5	100 ± 0.0
100	13 ± 4.1	0.2 ± 0.0	100 ± 0.0	25 ± 16.4	1.6 ± 0.4	100 ± 0.0	13 ± 4.1	2.3 ± 0.3	100 ± 0.0
150	13 ± 3.8	0.2 ± 0.0	100 ± 0.0	21 ± 12.2	1.5 ± 0.4	100 ± 0.0	13 ± 4.0	2.4 ± 0.4	100 ± 0.0
200	14 ± 6.0	0.2 ± 0.1	100 ± 0.0	24 ± 14.3	1.6 ± 0.4	100 ± 0.0	14 ± 3.8	2.4 ± 0.3	100 ± 0.0
300	13 ± 4.2	0.2 ± 0.0	100 ± 0.0	21 ± 9.5	1.5 ± 0.3	100 ± 0.0	14 ± 5.1	2.2 ±.3	100 ± 0.0

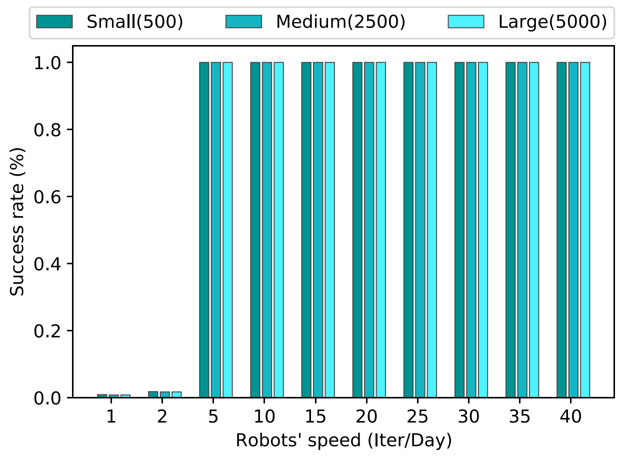


Fig. 16 Comparing the success rate in different environment sizes with increasing robots' speed (IterDay) in a random CR updating

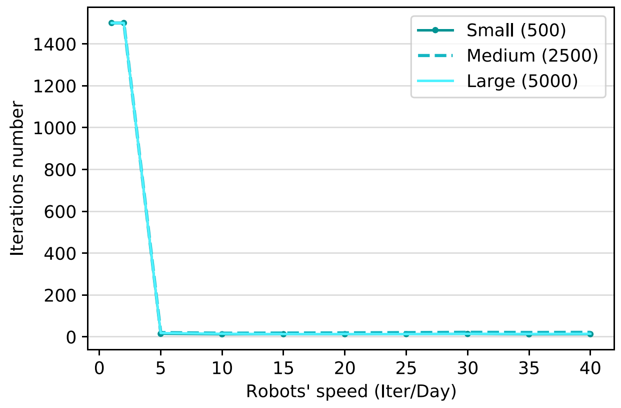


Fig. 17 Comparing the iterations number in different environment sizes with increasing robots' speed (IterDay) in a random CR updating

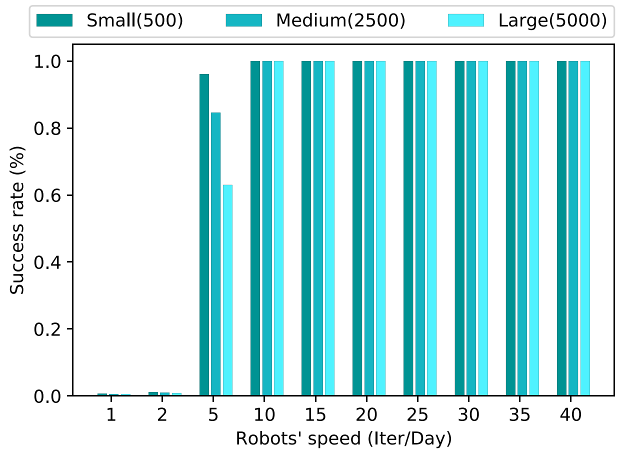


Fig. 18 Comparing the success rate in different environment sizes with increasing robots' speed (IterDay) in a diminishing CR

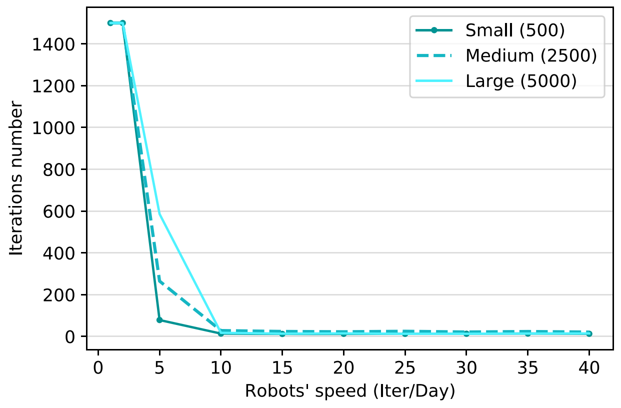


Fig. 19 Comparing the iterations number in different environment sizes with increasing robots' speed (IterDay) in a diminishing CR

5.2.4 Results of the real case study

In this part, we aim to test our REHO search approach on real data. The idea is to investigate the relation between the search start day at time (day) $d + k$ and the REHO performances, corresponding to a time lag of k days. We choose a $k = 5$ days. First COVID-19 case in the USA have been registered on 21/01/2020, in Table 4 are presented the results of the REHO targets' search with the different search start days. For each search mission, we update the number of targets and assign them to their real position on the USA Map in each *IterDay* (i.e., day), for that we also test three robots' speed with *IterDay* equal to 10, 50 and 100 iterations per day.

From Fig. 20, It can be observed that the REHO search approach makes a 100% success rate when intervening before 2020-03-01 no matter its robots' speed. However, from 2020-03-01, low-speed robots face difficulties to reach all the targets and do not exceed 9% , after what it abort the search mission. Medium and High-speed robots are not able to reach 100% of the targets when starting the search mission from 2020-03-05, they get 21% and 37%, respectively.

Figure 21 shows that the REHO approach provides very few efforts (less than 12 iterations in less than 4 s) when early starting the search mission. Unlike the late search start where robots become unable to succeed in the mission. In other words, the later the research is launched, the less the mission is likely to succeed.

We notice that the USA data used here did not apply the containment at all, but even with that fact, the REHO approach was able to accomplish some successful simulated missions.

Table 4 Average iterations number, average execution time, and success rate of REHO search approach in USA simulation with different search start days

Day	10 iterDay			50 iterDay			100 iterDay		
	Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
2020-01-21	1 ± 0.0	0.0 ± 0.0	100 ± 0.0	1 ± 0.0	0.0 ± 0.0	100 ± 0.0	1 ± 0.0	0.0 ± 0.0	100 ± 0.0
2020-01-26	3 ± 0.8	0.1 ± 0.0	100 ± 0.0	3 ± 0.8	0.1 ± 0.0	100 ± 0.0	3 ± 0.9	0.1 ± 0.0	100 ± 0.0
2020-01-31	3 ± 0.5	0.1 ± 0.0	100 ± 0.0	3 ± 0.5	0.1 ± 0.0	100 ± 0.0	3 ± 0.7	0.1 ± 0.0	100 ± 0.0
2020-02-05	5 ± 0.9	0.1 ± 0.0	100 ± 0.0	5 ± 1.1	0.2 ± 0.0	100 ± 0.0	5 ± 1.1	0.2 ± 0.0	100 ± 0.0
2020-02-10	5 ± 1.0	0.2 ± 0.1	100 ± 0.0	5 ± 2.6	0.2 ± 0.1	100 ± 0.0	5 ± 1.0	0.2 ± 0.0	100 ± 0.0
2020-02-15	5 ± 0.9	0.2 ± 0.0	100 ± 0.0	5 ± 0.8	0.2 ± 0.0	100 ± 0.0	5 ± 2.5	0.2 ± 0.0	100 ± 0.0
2020-02-20	7 ± 2.0	0.4 ± 0.0	100 ± 0.0	8 ± 2.4	0.4 ± 0.1	100 ± 0.0	7 ± 1.3	0.4 ± 0.0	100 ± 0.0
2020-02-25	11 ± 4.0	1.1 ± 0.2	100 ± 0.0	9 ± 1.9	1.0 ± 0.1	100 ± 0.0	10 ± 1.7	1.0 ± 0.1	100 ± 0.0
2020-03-01	1500 ± 0.0	158.5 ± 7.3	9 ± 0.0	18 ± 2.0	3.6 ± 0.3	100 ± 0.0	18 ± 1.2	3.4 ± 0.3	100 ± 0.0
2020-03-05	1500 ± 0.0	175.2 ± 7.0	9 ± 0.0	1500 ± 0.0	554.5 ± 34.5	21 ± 0.0	1500 ± 0.0	913.3 ± 47.3	37 ± 0.0

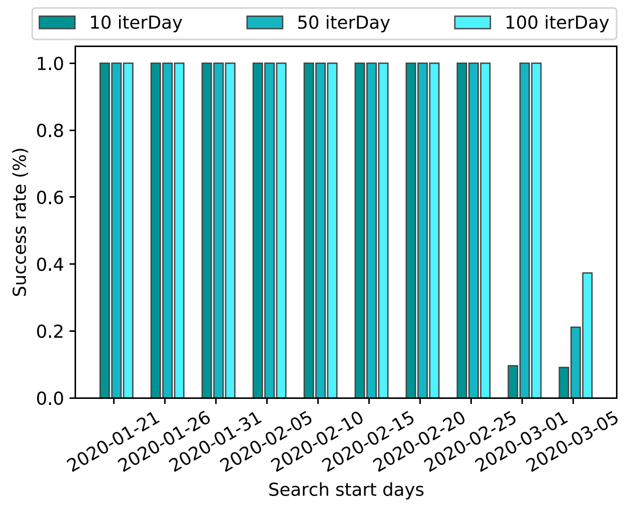


Fig. 20 Comparing the success rate with different search start days and different robots' speed

5.3 Comparison with other swarm intelligence algorithms

In order to situate our proposed REHO approach relative to the existing methods, we compared it with two recent SI based algorithms from the state-of-the-art that are A-RPSO [3] and MFPSO [17]. These two approaches have initially been developed for the static TDP, here we adapted them for the proposed dynamic target search problem modeling. In what follow are presented the comparative results of these approaches in environments' size of 2500 × 2500 squares.

5.3.1 Influence of the initial reproduction rate: R_0

This part presents comparative experiments relative to the initial reproduction rate R_0 within three kinds of Containment Rate's evolution types: growing CR, random CR, and diminishing CR. Each experiment is repeated 50 times. The robots' speed ($IterDay$) is fixed to 5 iterations/day, and we set the initial number of targets ($\#Target_0$) to 50 targets. Also, the size of the search environment in all experiments of this section is set to 2500 × 2500 squares. The obtained results are shown in Table 5.

As seen in Figs. 22 and 23, when increasing the initial reproduction rate R_0 in a growing CR, the proposed approach has significant performance compared to A-RPSO and MFPSO. More precisely, Fig. 22 shows the higher success rates of REHO compared with other approaches, such as REHO achieve 100% success rate even with small R_0 values, while the success rate of A-RPSO and MFPSO decease from 17% to less than 1% and from 98 to 15%, respectively. Furthermore, from Fig. 23 we can observe that the proposed approach performs far fewer iterations compared with A-RPSO and MFPSO.

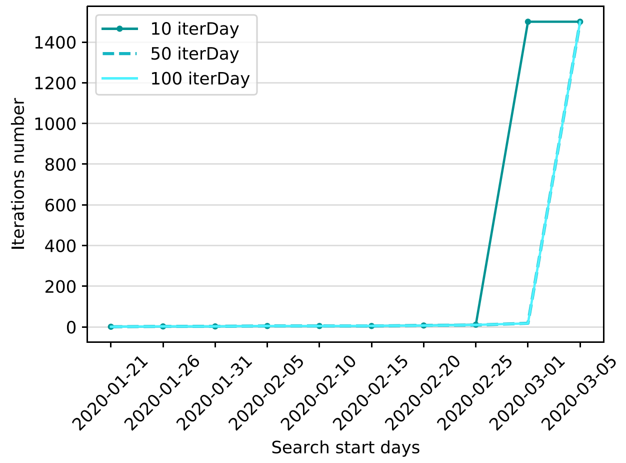


Fig. 21 Comparing the iterations number with different start search days and different robots' speed

From Figs. 24 and 25 we can see the important advantage of REHO over A-RPSO and MFPSO in random Containment Rate. Figure 24 points out that with the increase of R_0, the success rate of the proposed approach decline from 100 to 69%. Compared with A-RPSO and MFPSO which both have less than 1% of success rates for all the tested R_0 values. Additionally, Fig. 25 shows the disparity between the number of iterations of REHO (less than 500 iterations) compared with the other methods that reach the maximum iterations number.

The last 8 rows of Table 5 show that in a diminishing CR the results of the experiments are close to the one obtained in a Random CR, the proposed search algorithm is the only one to get successful search missions even with the drop of its success rate from 100 to 13% and the raise of its iterations number from 24 to 1322 iterations when increasing R_0.

Table 5 Average iterations number, average execution time, and success rate of the compared algorithms in complex environments with different Containment Rate updates facing the variation of initial reproduction rate (R0) values

*R*0	A-RPSO [3]			MFPSO [17]			REHO		
	Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
Growing CR									
1.5	1500 ± 0.0	10.5 ± 2.1	17 ± 0.1	1428 ± 126.2	6.1 ± 1.8	98 ± 0.0	13 ± 2.0	1.2 ± 0.2	100 ± 0.0
2	1500 ± 0.0	16.5 ± 3.0	11 ± 0.0	1498 ± 13.7	11.6 ± 4.1	92 ± 0.1	14 ± 2.4	1.3 ± 0.2	100 ± 0.0
2.5	1500 ± 0.0	20.8 ± 2.0	8 ± 0.0	1500 ± 0.0	18.8 ± 5.5	82 ± 0.1	14 ± 2.8	1.3 ± 0.2	100 ± 0.0
3	1500 ± 0.0	22.1 ± 1.0	4 ± 0.0	1500 ± 0.0	26.0 ± 4.4	63 ± 0.1	14 ± 2.0	1.5 ± 0.3	100 ± 0.0
3.5	1500 ± 0.0	22.0 ± 0.5	2 ± 0.0	1500 ± 0.0	29.1 ± 1.0	43 ± 0.1	16 ± 3.1	1.6 ± 0.4	100 ± 0.0
4	1500 ± 0.0	22.2 ± 2.5	1 ± 0.0	1500 ± 0.0	29.4 ± 0.5	28 ± 0.1	17 ± 3.8	1.6 ± 0.6	100 ± 0.0
4.5	1500 ± 0.0	21.3 ± 0.1	1 ± 0.0	1500 ± 0.0	29.8 ± 0.6	20 ± 0.1	18 ± 4.0	1.6 ± 0.5	100 ± 0.0
5	1500 ± 0.0	21.3 ± 0.0	0 ± 0.0	1500 ± 0.0	29.7 ± 0.4	15 ± 0.1	21 ± 6.0	1.9 ± 1.2	100 ± 0.0
Random CR									
1.5	1500 ± 0.0	0.7 ± 0.0	0 ± 0.0	1500 ± 0.0	1.0 ± 0.1	0 ± 0.0	17 ± 4.7	1.3 ± 0.2	100 ± 0.0
2	1500 ± 0.0	0.5 ± 0.0	0 ± 0.0	1500 ± 0.0	0.8 ± 0.0	0 ± 0.0	16 ± 4.0	1.3 ± 0.3	100 ± 0.0
2.5	1500 ± 0.0	0.5 ± 0.0	0 ± 0.0	1500 ± 0.0	0.7 ± 0.0	0 ± 0.0	17 ± 4.2	1.5 ± 0.3	100 ± 0.0
3	1500 ± 0.0	0.4 ± 0.0	0 ± 0.0	1500 ± 0.0	0.6 ± 0.0	0 ± 0.0	19 ± 5.1	1.7 ± 0.5	100 ± 0.0
3.5	1500 ± 0.0	0.4 ± 0.0	0 ± 0.0	1500 ± 0.0	0.5 ± 0.0	0 ± 0.0	21 ± 6.8	1.9 ± 0.8	100 ± 0.0
4	1500 ± 0.0	0.4 ± 0.0	0 ± 0.0	1500 ± 0.0	0.5 ± 0.0	0 ± 0.0	111 ± 351.0	4.7 ± 9.7	94 ± 0.2
4.5	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1500 ± 0.0	0.4 ± 0.0	0 ± 0.0	201 ± 480.0	6.7 ± 11.5	88 ± 0.3
5	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1500 ± 0.0	0.4 ± 0.0	0 ± 0.0	497 ± 688.4	12.1 ± 13.8	69 ± 0.5
Diminishing CR									
1.5	1500 ± 0.0	0.5 ± 0.0	0 ± 0.0	1500 ± 0.0	0.6 ± 0.0	0 ± 0.0	24 ± 12.7	1.7 ± 0.6	100 ± 0.0
2	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1500 ± 0.0	0.5 ± 0.0	0 ± 0.0	85 ± 289.2	3.3 ± 7.7	96 ± 0.2
2.5	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1500 ± 0.0	0.4 ± 0.0	0 ± 0.0	298 ± 563.7	8.2 ± 12.3	82 ± 0.4
3	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1500 ± 0.0	0.4 ± 0.0	0 ± 0.0	793 ± 736.1	14.8 ± 12.3	49 ± 0.5
3.5	1500 ± 0.0	0.2 ± 0.0	0 ± 0.0	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	882 ± 726.7	19.4 ± 14.9	43 ± 0.5
4	1500 ± 0.0	0.2 ± 0.0	0 ± 0.0	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1088 ± 660.8	17.9 ± 10.0	29 ± 0.4
4.5	1500 ± 0.0	0.2 ± 0.0	0 ± 0.0	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1265 ± 538.1	18.5 ± 7.2	17 ± 0.4
5	1500 ± 0.0	0.2 ± 0.0	0 ± 0.0	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1322 ± 482.2	17.8 ± 6.3	13 ± 0.3

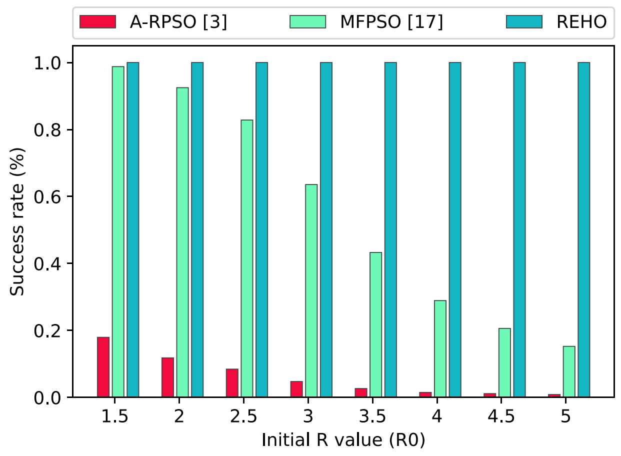


Fig. 22 Comparing the success rate of different approaches with an increasing R_0 in a Growing CR

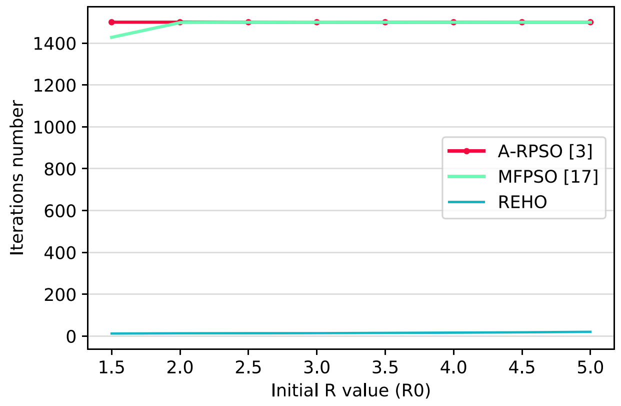


Fig. 23 Comparing the iterations number of different approaches with an increasing R_0 in a Growing CR

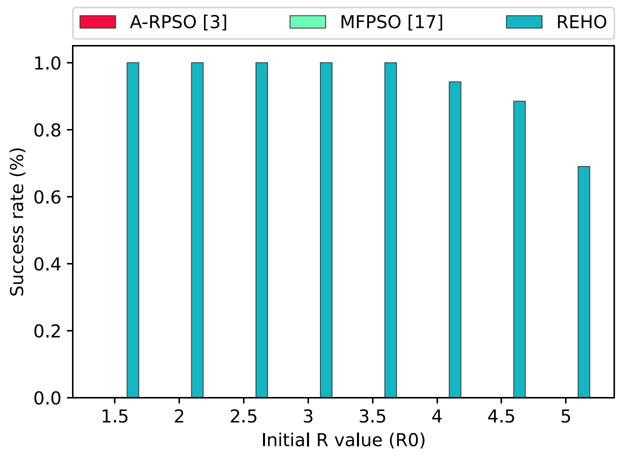


Fig. 24 Comparing the success rate of different approaches with an increasing R_0 in a Random CR

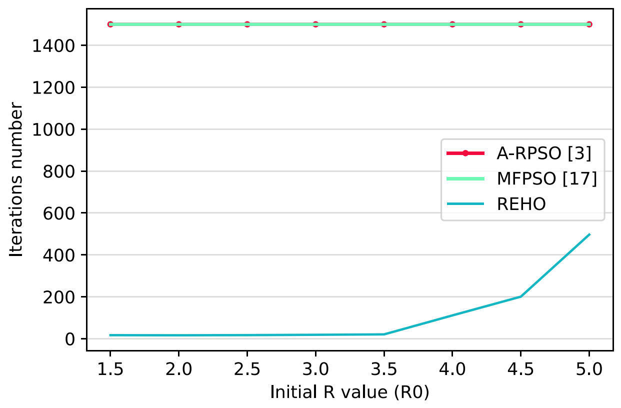


Fig. 25 Comparing the iterations number of different approaches with an increasing R_0 in a Random CR

From the comparative results on the initial reproduction rate R_0 just presented, we clearly remark that the proposed REHO outperforms the other methods in either growing, random or diminishing CR. This can be explained by the multi-clans structure of REHO and the separating operator that enhance the exploration phase and prevent from falling in local optima. For these reasons, REHO has a better ability to find much more targets before their number grows too rapidly and becomes out of control.

5.3.2 Influence of the initial targets' number: #*Target*$_0$

In this Section, the influence of the initial target's number $\#Target_0$ on the three compared methods is addressed. The experiments are carried for three Containment Rate types: growing, random and diminishing CR. Each experiment is repeated 50 times in environments of size 2500×2500 squares. The other parameters are fixed with an initial reproduction rate (R_0) set to 2.5 and a robots' speed ($IterDay$) of 25 iterations per day. The comparative results of REHO, A-RPSO, and MFPSO are exhibited in Table 6.

Figures 26 and 27 are the graphical representation of the comparative results of REHO, A-RPSO, and MFPSO algorithms in a growing CR. We can notice REHO's superiority in success rate and iterations number. The proposed approach reaches a 100% of success rate in less than 109 iterations. Compared to A-RPSO success rate that decreases from 64% to less than 1% with from 1388 to 1500 iterations, and the success rate of MFPSO decrease from 99 to 10% with between 659 and 1500 iterations.

Figures 28 and 29 compare the success rate and average iterations number in random Containment Rate for REHO, A-RPSO, and MFPSO algorithms. We notice that A-RPSO and MFPSO fail at the search mission even with few initial numbers of targets, while REHO attains 100% of success rate in less than 106 iterations. Except when $\#Target_0 =$

Table 6 Average iterations number, average execution time, and success rate of the compared algorithms in complex environments with different Containment Rate updates facing the variation of initial targets' number

$\#Target_0$	A-RPSO [3]			MFPSO [17]			REHO		
	Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
Growing CR									
5	1388 ± 239.9	1.9 ± 0.9	64 ± 0.3	659 ± 284.4	1.1 ± 0.7	99 ± 0.0	4 ± 1.8	0.1 ± 0.0	100 ± 0.0
50	1500 ± 0.0	19.0 ± 2.7	12 ± 0.1	1500 ± 0.0	17.7 ± 4.8	81 ± 0.1	19 ± 8.1	1.3 ± 0.3	100 ± 0.0
100	1500 ± 0.0	20.9 ± 0.0	2 ± 0.0	1500 ± 0.0	27.9 ± 1.3	51 ± 0.1	34 ± 10.8	4.6 ± 0.9	100 ± 0.0
150	1500 ± 0.0	21.0 ± 0.0	2 ± 0.0	1500 ± 0.0	28.7 ± 0.0	37 ± 0.1	38 ± 9.2	8.5 ± 1.3	100 ± 0.0
200	1500 ± 0.0	21.1 ± 0.4	1 ± 0.0	1500 ± 0.0	28.8 ± 0.0	30 ± 0.1	44 ± 7.7	13.0 ± 1.1	100 ± 0.0
250	1500 ± ± 0.0	21.1 ± 0.2	1 ± 0.0	1500 ± 0.0	28.8 ± 0.0	22 ± 0.1	51 ± 8.8	19.3 ± 1.8	100 ± 0.0
300	1500 ± 0.0	21.8 ± 0.6	0 ± 0.0	1500 ± ± 0.0	28.8 ± 0.0	17 ± 0.0	64 ± 8.1	29.5 ± 2.8	100 ± 0.0
350	1500 ± 0.0	21.1 ± 0.6	0 ± 0.0	1500 ± 0.0	28.8 ± 0.0	15 ± 0.0	72 ± 11.6	40.0 ± 6.6	100 ± 0.0
400	1500 ± 0.0	21.6 ± 0.6	0 ± 0.0	1500 ± 0.0	28.8 ± 0.0	12 ± 0.0	87 ± 11.8	61.5 ± 11.5	100 ± 0.0
500	1500 ± 0.0	22.0 ± 0.5	0 ± 0.0	1500 ± 0.0	28.8 ± 0.0	10 ± 0.0	125 ± 22.3	108.7 ± 28.0	100 ± 0.0
Random CR									
5	1500 ± 0.0	2.4 ± 0.1	0 ± 0.0	1500 ± 0.0	3.4 ± 0.2	0 ± 0.0	4 ± 2.6	0.1 ± 0.0	100 ± 0.0
50	1500 ± 0.0	2.4 ± 0.1	0 ± 0.0	1500 ± 0.0	3.2 ± 0.1	0 ± 0.0	21 ± 8.0	1.5 ± 0.3	100 ± 0.0
100	1500 ± 0.0	2.3 ± 0.0	0 ± 0.0	1500 ± 0.0	3.1 ± 0.1	0 ± 0.0	33 ± 10.5	4.5 ± 0.9	100 ± 0.0
150	1500 ± 0.0	2.1 ± 0.0	0 ± 0.0	1500 ± 0.0	2.8 ± 0.1	0 ± 0.0	36 ± 7.0	8.5 ± 1.1	100 ± 0.0
200	1500 ± 0.0	2.1 ± 0.0	0 ± 0.0	1500 ± 0.0	2.9 ± 0.0	0 ± 0.0	45 ± 8.5	13.1 ± 1.3	100 ± 0.0
250	1500 ± 0.0	2.0 ± 0.2	0 ± 0.0	1500 ± 0.0	2.7 ± 0.2	0 ± 0.0	52 ± 9.0	19.6 ± 2.0	100 ± 0.0
300	1500 ± 0.0	1.8 ± 0.1	0 ± 0.0	1500 ± 0.0	2.5 ± 0.1	0 ± 0.0	66 ± 9.5	31.9 ± 4.2	100 ± 0.0
350	1500 ± 0.0	1.8 ± 0.0	0 ± 0.0	1500 ± 0.0	2.4 ± 0.0	0 ± 0.0	78 ± 14.0	43.4 ± 9.6	100 ± 0.0
400	1500 ± 0.0	1.8 ± 0.0	0 ± 0.0	1500 ± 0.0	2.4 ± 0.0	0 ± 0.0	105 ± 21.9	72.3 ± 19.2	100 ± 0.0
500	1500 ± 0.0	1.7 ± 0.2	0 ± 0.0	1500 ± 0.0	2.3 ± 0.2	0 ± 0.0	1158 ± 577.8	230.2 ± 66.1	37 ± 0.4
Diminishing CR									
5	1500 ± 0.0	1.5 ± 0.1	0 ± 0.0	1500 ± 0.0	2.0 ± 0.1	0 ± 0.0	3 ± 1.7	0.1 ± 0.0	100 ± 0.0
50	1500 ± 0.0	1.4 ± 0.1	0 ± 0.0	1500 ± 0.0	1.9 ± 0.1	0 ± 0.0	22 ± 11.5	1.5 ± 0.4	100 ± 0.0
100	1500 ± 0.0	1.5 ± 0.2	0 ± 0.0	1500 ± 0.0	2.1 ± 0.2	0 ± 0.0	44 ± 23.3	4.9 ± 1.2	100 ± 0.0
150	1500 ± 0.0	1.4 ± 0.0	0 ± 0.0	1500 ± 0.0	1.8 ± 0.1	0 ± 0.0	44 ± 13.0	9.3 ± 1.6	100 ± 0.0
200	1500 ± 0.0	1.5 ± 0.1	0 ± 0.0	1500 ± 0.0	1.9 ± 0.0	0 ± 0.0	57 ± 15.0	14.1 ± 1.5	100 ± 0.0
250	1500 ± 0.0	1.4 ± 0.0	0 ± 0.0	1500 ± 0.0	1.9 ± 0.0	0 ± 0.0	72 ± 26.3	58.6 ± 23.2	100 ± 0.0
300	1500 ± 0.0	1.4 ± 0.1	0 ± 0.0	1500 ± 0.0	1.9 ± 0.0	0 ± 0.0	142 ± 53.9	23.2 ± 5.2	100 ± 0.0
350	1500 ± 0.0	1.3 ± 0.2	0 ± 0.0	1500 ± 0.0	1.8 ± 0.2	0 ± 0.0	191 ± 269.4	49.1 ± 33.9	96 ± 0.2
400	1500 ± 0.0	1.1 ± 0.1	0 ± 0.0	1500 ± 0.0	1.6 ± 0.2	0 ± 0.0	879 ± 674.2	141.5 ± 78.0	51 ± 0.4
500	1500 ± 0.0	1.1 ± 0.0	0 ± 0.0	1500 ± 0.0	1.5 ± 0.0	0 ± 0.0	1313 ± 463.7	165.5 ± 57.5	21 ± 0.3

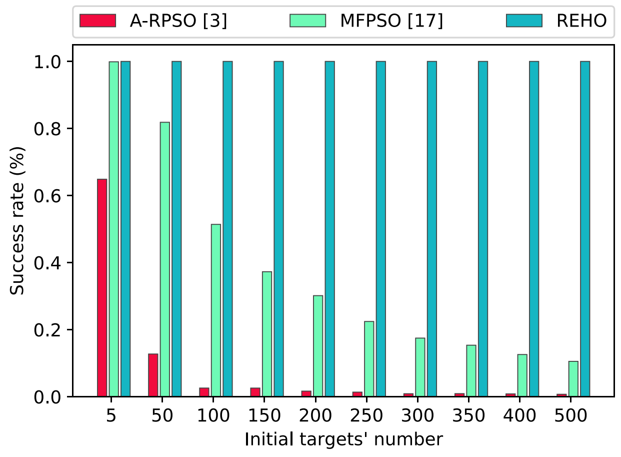


Fig. 26 Comparing the success rate of different approaches with an increasing $\#Target_0$ in a Growing CR

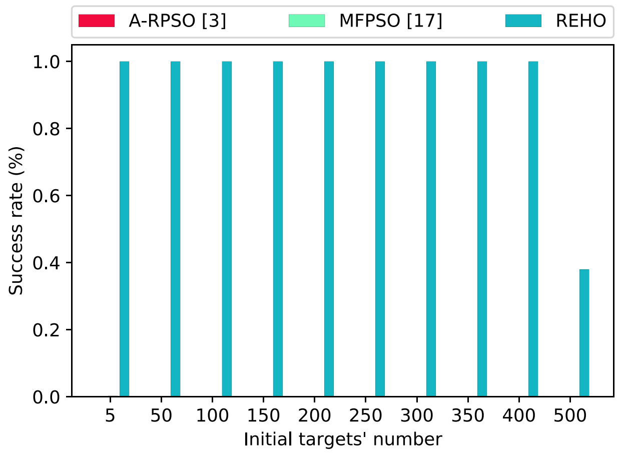


Fig. 28 Comparing the success rate of different approaches with an increasing $\#Target_0$ in a Random CR

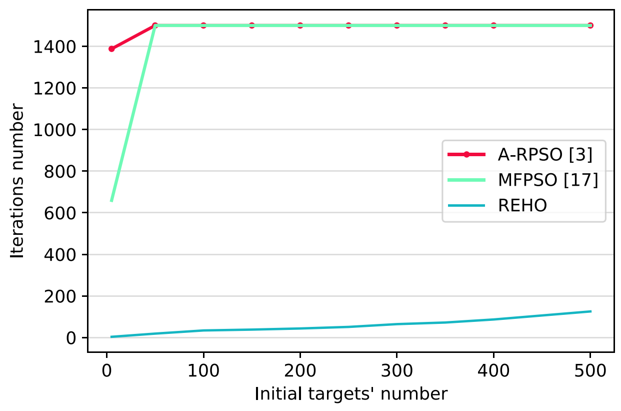


Fig. 27 Comparing the iterations number of different approaches with an increasing $\#Target_0$ in a Growing CR

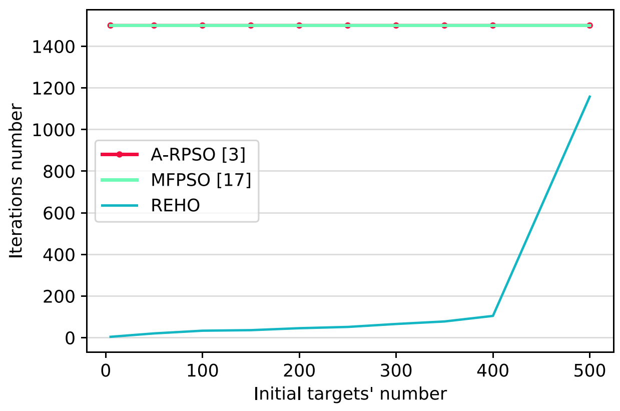


Fig. 29 Comparing the iterations number of different approaches with an increasing $\#Target_0$ in a Random CR

500, it achieves 37% of success rate with an average of 1158 iterations.

In the part that concern diminishing CR of Table 6 are presented the experimental results of the proposed method and the two compared algorithms. As for random CR, A-RPSO and MFPSO abort the mission after less than 2s because of the height and rapid inflation of targets' number. Whereas, REHO success rate is somehow inversely proportional to the $\#Target_0$ values such as with the increase of initial targets' number, its success rate decrease from 100 to 21% with from 3 to 1313 iterations.

To sum up these experiments, we remark that contrary to the proposed REHO, the tested A-RPSO and MFPSO are totally inefficient in the worst scenario (random and diminishing CR). But, even when the containment is well applied (growing CR), REHO has significantly better performance than others. This is due to the multi-swarms strategy of REHO and its separating operator that increase considerably the covered search space and the number of targets that can be handled in the less possible time.

5.3.3 Influence of the robots' speed: IterDay

This part is devoted to the study of the impact of the robots' speed on the different compared approaches. Table 7 reports the experimental results obtained in environments with a growing, random, and diminishing CR. The represented average success rate, average iterations number, and average execution time are the results of 50 executions per environment, within 2500×2500 squares environments' sizes. Also, the initial reproduction rate R_0 is set to 2.5, and the initial number of targets $\#Target_0$ is fixed to 50 targets.

Figure 30 shows that increasing the robots' speed in a growing CR doesn't especially affect the search performances of the tested approaches. For example, REHO records the best success rate (100%) independently of $IterDay$ values. While A-RPSO and MFPSO achieve around 12% and 83% success rates, respectively. However, from Fig. 31 we can observe that REHO has superiority in iterations number. With the increase of the robots' speed, robots search and find the targets faster.

Table 7 Average iterations number, average execution time, and success rate of the compared algorithms in complex environments with different Containment Rate updates facing the variation of robots' speed (iter/day)

IterDay	A-RPSO [3]			MFPSO [17]			REHO		
	Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
Growing CR									
1	1500 ± 0.0	20.8 ± 2.1	12 ± 0.1	1500 ± 3.5	17.8 ± 5.2	83 ± 0.1	54 ± 18.6	35.3 ± 20.7	100 ± 0.0
5	1500 ± 0.0	20.8 ± 1.8	13 ± 0.1	1498 ± 8.9	16.8 ± 5.4	84 ± 0.1	33 ± 12.7	11.1 ± 11.5	100 ± 0.0
10	1500 ± 0.0	20.4 ± 2.8	13 ± 0.1	1500 ± 0.0	18.9 ± 5.3	80 ± 0.1	20 ± 6.4	1.8 ± 0.7	100 ± 0.0
20	1500 ± 0.0	19.6 ± 2.8	12 ± 0.1	1500 ± 0.0	17.7 ± 5.0	82 ± 0.1	16 ± 5.3	1.3 ± 0.2	100 ± 0.0
35	1500 ± 0.0	19.6 ± 2.5	11 ± 0.1	1492 ± 50.3	16.7 ± 5.5	83 ± 0.1	14 ± 4.3	1.2 ± 0.2	100 ± 0.0
50	1500 ± 0.0	19.5 ± 2.3	10 ± 0.1	1500 ± 0.0	16.7 ± 4.4	83 ± 0.1	15 ± 3.7	1.2 ± 0.2	100 ± 0.0
100	1500 ± 0.0	18.9 ± 2.7	13 ± 0.1	1499 ± 3.9	16.9 ± 4.8	83 ± 0.1	14 ± 2.9	1.3 ± 0.2	100 ± 0.0
150	1500 ± 0.0	19.2 ± 2.8	13 ± 0.1	1495 ± 36.0	16.4 ± 4.9	83 ± 0.1	15 ± 3.6	1.3 ± 0.2	100 ± 0.0
200	1500 ± 0.0	18.6 ± 2.6	13 ± 0.1	1500 ± 0.0	15.3 ± 5.2	85 ± 0.1	14 ± 3.5	1.3 ± 0.2	100 ± 0.0
300	1500 ± 0.0	18.3 ± 2.9	13 ± 0.1	1498 ± 14.6	15.2 ± 5.3	83 ± 0.1	15 ± 5.7	1.2 ± 0.2	100 ± 0.0
Random CR									
1	1500 ± 0.0	0.1 ± 0.0	0 ± 0.0	1500 ± 0.0	0.1 ± 0.0	0 ± 0.0	1500 ± 0.0	6.6 ± 0.6	0 ± 0.0
5	1500 ± 0.0	0.2 ± 0.0	0 ± 0.0	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1500 ± 0.0	14.1 ± 1.2	1 ± 0.0
10	1500 ± 0.0	0.5 ± 0.0	0 ± 0.0	1500 ± 0.0	0.6 ± 0.0	0 ± 0.0	21 ± 6.3	2.0 ± 0.9	100 ± 0.0
20	1500 ± 0.0	1.0 ± 0.0	0 ± 0.0	1500 ± 0.0	1.3 ± 0.1	0 ± 0.0	18 ± 4.7	1.4 ± 0.3	100 ± 0.0
35	1500 ± 0.0	1.4 ± 0.1	0 ± 0.0	1500 ± 0.0	1.9 ± 0.1	0 ± 0.0	18 ± 6.3	1.4 ± 0.3	100 ± 0.0
50	1500 ± 0.0	1.9 ± 0.1	0 ± 0.0	1500 ± 0.0	2.6 ± 0.0	0 ± 0.0	19 ± 7.4	1.4 ± 0.3	100 ± 0.0
100	1500 ± 0.0	2.4 ± 0.1	0 ± 0.0	1500 ± 0.0	3.2 ± 0.1	0 ± 0.0	20 ± 7.6	1.4 ± 0.3	100 ± 0.0
150	1500 ± 0.0	2.8 ± 0.2	0 ± 0.0	1500 ± 0.0	3.8 ± 0.1	0 ± 0.0	22 ± 9.0	1.4 ± 0.3	100 ± 0.0
200	1500 ± 0.0	3.3 ± 0.2	0± 0.0	1500 ± 0.0	4.5 ± 0.1	0 ± 0.0	21 ± 9.6	1.4 ± 0.3	100 ± 0.0
300	1500 ±0.0	3.8±0.2	0±0.0	1500 ± 0.0	5.1± 0.2	0 ± 0.0	22 ± 11.0	1.4± 0.3	100 ± 0.0
Diminishing CR									
1	1500 ± 0.0	0.1 ± 0.0	0 ± 0.0	1500 ± 0.0	0.1 ± 0.0	0 ± 0.0	1500 ± 0.0	4.7 ± 0.4	0 ± 0.0
5	1500 ± 0.0	0.1 ± 0.0	0 ± 0.0	1500 ± 0.0	0.2 ± 0.0	0 ± 0.0	1500 ± 0.0	9.4 ± 0.9	0 ± 0.0
10	1500 ± 0.0	0.3 ± 0.0	0 ± 0.0	1500 ± 0.0	0.4 ± 0.0	0 ± 0.0	265 ± 539.2	7.3 ± 11.9	84 ± 0.4
20	1500 ± 0.0	0.6 ± 0.0	0 ± 0.0	1500 ± 0.0	0.8 ± 0.0	0 ± 0.0	28 ± 21.7	1.8 ± 0.9	100 ± 0.0
35	1500 ± 0.0	0.9 ± 0.0	0 ± 0.0	1500 ± 0.0	1.1 ± 0.0	0 ± 0.0	24 ± 13.7	1.6 ± 0.4	100 ± 0.0
50	1500 ± 0.0	1.1 ± 0.0	0 ± 0.0	1500 ± 0.0	1.5 ± 0.0	0 ± 0.0	23 ± 11.6	1.5 ± 0.3	100 ± 0.0
100	1500 ± 0.0	1.4 ± 0.0	0 ± 0.0	1500 ± 0.0	1.9 ± 0.1	0 ± 0.0	25 ± 16.4	1.6 ± 0.4	100 ± 0.0
150	1500 ± 0.0	1.7 ± 0.0	0 ± 0.0	1500 ±0.0	2.4 ± 0.4	0 ± 0.0	21 ± 12.2	1.5± 0.4	100 ± 0.0
200	1500 ± 0.0	2.0 ± 0.0	0 ± 0.0	1500 ± 0.0	2.9 ± 0.2	0 ± 0.0	24 ± 14.3	1.6 ± 0.4	100 ± 0.0
300	1500 ± 0.0	2.3 ± 0.1	0 ± 0.0	1500 ± 0.0	3.4 ± 0.3	0 ± 0.0	21 ± 9.5	1.5 ± 0.3	100 ± 0.0

Figures 32 and 33 illustrate the average success rates and average number of iterations of the compared methods in a random CR. We can see A-RPSO and MFPSO failure at their search mission no matter how fast their robots are, targets proliferate faster. Whereas, The proposed method provides much better performance and higher efficiency with a robots' speed of 5 iterations/day and more, reaching a 100% success rate in less than 22 iterations.

In the last third of Table 7 are summarized the experiments of A-RPSO, MFPSO, and REHO in a diminishing CR. The results are not different from those presented in a random CR. REHO starts getting successful search missions when its robots' speed reaches 10 iterations/day. Compared to other methods are forced to abort the mission due to the number of targets which has grown uncontrollably.

According to the presented results, we remark that REHO provides the best results in the three Containment Rate's evolution types. These results are mostly due to the sub-swarms structure of the robots combined with the efficient path planning strategy that minimize and optimize the robots' efforts event in the worst scenarios.

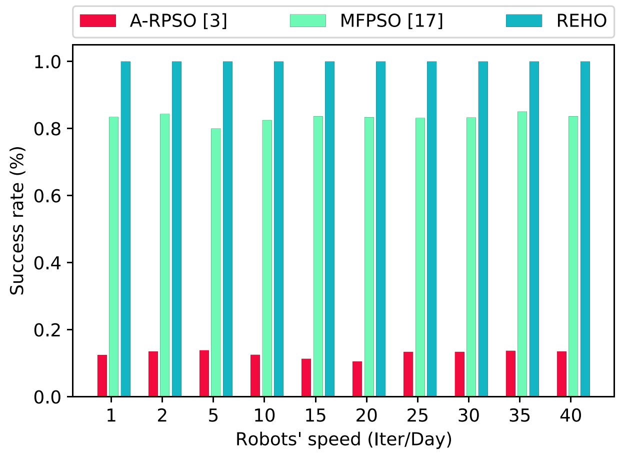


Fig. 30 Comparing the success rate of different approaches with an increasing robots' speed ($IterDay$) in a Growing CR

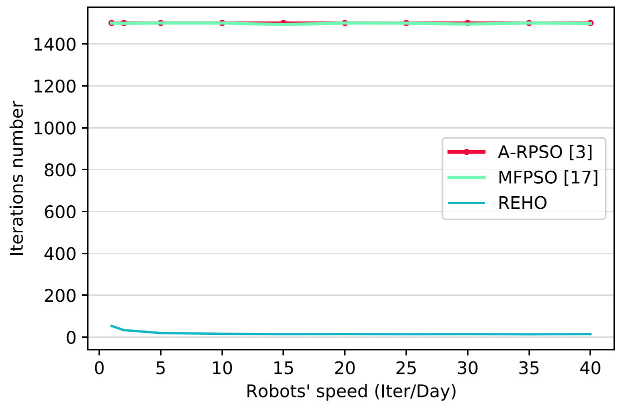


Fig. 31 Comparing the iterations number of different approaches with an increasing robots' speed ($IterDay$) in a Growing CR

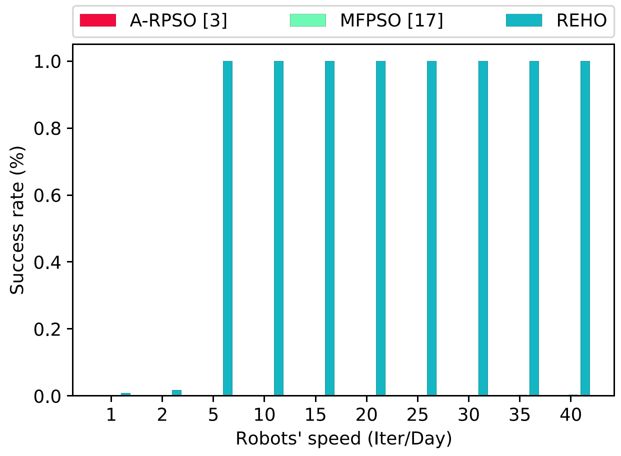


Fig. 32 Comparing the success rate of different approaches with an increasing robots' speed ($IterDay$) in a Random CR

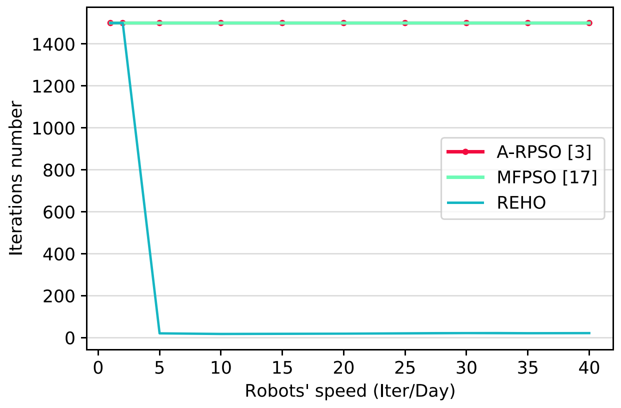


Fig. 33 Comparing the iterations number of different approaches with an increasing robots' speed ($IterDay$) in a Random CR

5.3.4 Results of the real case study

This section concerns the comparison of A-RPSO and MFPSO algorithms with the proposed REHO search approach on real data. Table 8 presents the results of the three compared approaches with different search start days. For each search mission, we update the number of targets and assign them to their real position on the USA Map in each day ($IterDay$), for that we fixed the robots' speed $IterDay$ to 50 iterations per day.

Figures 34 and 35 show that REHO provides better results than A-RPSO and MFPSO. The later the search mission starts, the less efficient and effective the search approaches are. For example, the proposed REHO has a 100% success rate in less than 3 iterations until 2020-03-01 where it falls to 21% in 1500 iterations. Compared to MFPSO's success rate that decreases from 100% in 214 iterations on 2020-01-21 to less than 1% in 1500 iterations on 2020-03-05. Whereas, A-RPSO algorithm is not able to lead to any successful search mission.

6 Conclusion and future work

In this research, we proposed a Robotic Elephant Herding Optimization algorithm for the target searching problem in complex and unknown environments. The original EHO is modified by adapting it to the robotic field, introducing a collision-free path planning strategy, and the robots' limitations and constraints on a searching algorithm. Then, a dynamic environment with exponential targets emergence is presented to emulate the COVID-19 spread or any other close-contact pathogens.

To verify the performance of the proposed algorithm, several experiments were performed in simulated environments. Considering our results, REHO significantly outperforms other approaches like A-RPSO and MFPSO. The proposed

Table 8 Average iterations number, average execution time, and success rate of the compared algorithms in USA simulation with different search start days

Day	A-RPSO [3]			MFPSO [17]			REHO		
	Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
2020-01-21	1500 ± 0.0	1.5 ± 0.1	0 ± 0.0	214 ± 276.1	0.2 ± 0.3	100 ± 0.0	1 ± 0.0	0.0 ± 0.0	100 ± 0.0
2020-01-26	1500 ± 0.0	2.0 ± 0.0	0 ± 0.0	561 ± 339.0	0.7 ± 0.4	100 ± 0.0	3 ± 0.8	0.1 ± 0.0	100 ± 0.0
2020-01-31	1500 ± 0.0	2.9 ± 0.0	0 ± 0.0	643 ± 359.1	0.8 ± 0.6	96 ± 0.1	3 ± 0.5	0.1 ± 0.0	100 ± 0.0
2020-02-05	1500 ± 0.0	5.1 ± 0.0	0 ± 0.0	948 ± 491.9	2.2 ± 1.7	69 ± 0.3	5 ± 1.1	0.2 ± 0.0	100 ± 0.0
2020-02-10	1500 ± 0.0	8.4 ± 0.0	0 ± 0.0	1261 ± 442.8	7.0 ± 3.5	31 ± 0.4	5 ± 2.6	0.2 ± 0.1	100 ± 0.0
2020-02-15	1500 ± 0.0	11.7 ± 0.1	0 ± 0.0	1458 ± 225.3	13.3 ± 2.4	7 ± 0.2	5 ± 0.8	0.2 ± 0.0	100 ± 0.0
2020-02-20	1500 ± 0.0	14.1 ± 0.1	0 ± 0.0	1500 ± 0.0	17.5 ± 0.2	1 ± 0.0	8 ± 2.4	0.4 ± 0.1	100 ± 0.0
2020-02-25	1500 ± 0.0	13.5 ± 0.0	0 ± 0.0	1500 ± 0.0	18.2 ± 0.4	1 ± 0.0	9 ± 1.9	1.0 ± 0.1	100 ± 0.0
2020-03-01	1500 ± 0.0	12.4 ± 0.0	0 ± 0.0	1500 ± 0.0	17.2 ± 0.5	1 ± 0.0	18 ± 2.0	3.6 ± 0.3	100 ± 0.0
2020-03-05	1500 ± 0.0	10.7 ± 0.1	0 ± 0.0	1500 ± 0.0	15.2 ± 0.4	0 ± 0.0	1500 ± 0.0	554.5 ± 34.5	21 ± 0.0

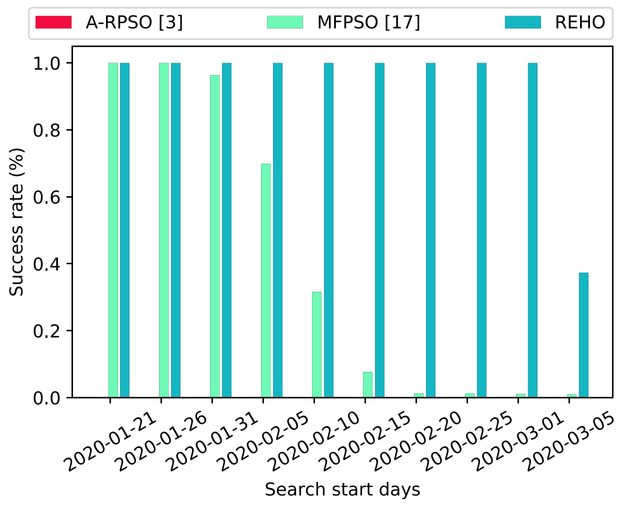


Fig. 34 Comparing the success rate of different approaches with different search start days

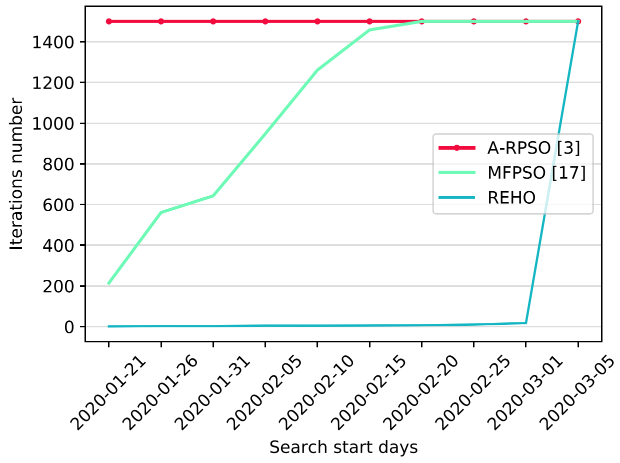


Fig. 35 Comparing the iterations number of different approaches with different start search days

approach offers better performances when the containment is respected and even more when robots have a medium to high speed by performing sufficient iterations per day. Also, we observed that the approach provides better outcomes when starting the search mission early before the number of initial targets increases dramatically. REHO draws its strengths from three main characteristics. First, its multiple clans' composition, that allows an advantageous diversity and coverage of the search space. Second, it separate operator which avoid falling into a local optimum. Finally, the optimized and rapid collision-free path planning that build short and safe paths without slowing down the robots' search. However, in this study, a simplified representation is considered for testing the proposed approach. Therefore, some real-world problems have not been addressed, such as dynamic obstacles and a limited range of communication among robots. For possible future work, we consider improving these topics. Furthermore, we plan to enhance the proposed COVID-19 spread's modeling by considering the new epidemiological models that continue to be daily enriched because of the very recent nature of this problem. On another side, we also envisage addressing other applications destined for autonomous multi-robots systems.

Acknowledgements We would like to express our special thanks of gratitude to the Directorate General for Scientific Research and Technological Development (DGRSDT), for the support of this work under the Grant Number C0662300.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Robin, C., Lacroix, S.: Multi-robot target detection and tracking: taxonomy and survey. Auton. Robots **40**, 729–760 (2016)
2. Lu, S., Hao, Y.: An evaluation of swarm robotic cooperative target search. In: 3rd International Conference on Electromechanical Control Technology and Transportation, SCITEPRESS - Science and Technology Publications (2018)
3. Dadgar, M., Jafari, S., Hamzeh, A.: A PSO-based multi-robot cooperation method for target searching in unknown environments. Neurocomputing **177**, 62–74 (2016)
4. Wang, G.-G., Deb, S., dos S. Coelho, L.: Elephant herding optimization. In: 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI). IEEE (2015)
5. Wang, G.G., Coelho, L.D.S., Gao, X.Z., Deb, S.: A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. Int. J. Bio-Inspired Comput. **8**(6), 394 (2016)
6. Flaxman, S. et al.: Report 13: Estimating the Number of Infections and the Impact of Non-pharmaceutical Interventions on COVID-19 in 11 European Countries (2020)
7. Lin, Q., Zhao, S., Gao, D., Lou, Y., Yang, S., Musa, S.S., Wang, M.H., Cai, Y., Wang, W., Yang, L., He, D.: A conceptual model for the coronavirus disease 2019 (COVID-19) outbreak in Wuhan, China with individual reaction and governmental action. Int. J. Infect. Dis. **93**, 211–216 (2020)
8. Wu, Z., McGoogan, J.M.: Characteristics of and important lessons from the coronavirus disease 2019 (COVID-19) outbreak in China. JAMA **323**, 1239 (2020)
9. Steven, S., Yen, T.L., Chonggang, X., Ethan, R.-S., Nick, H., Ruian, K.: High contagiousness and rapid spread of severe acute respiratory syndrome coronavirus 2. Emerg. Infect. Dis. **26**, 1470–1477 (2020)
10. Hellewell, J., Abbott, S., Gimma, A., Bosse, N.I., Jarvis, C.I., Russell, T.W., Munday, J.D., Kucharski, A.J., Edmunds, W.J., Funk, S., Eggo, R.M., Sun, F., Flasche, S., Quilty, B.J., Davies, N., Liu, Y., Clifford, S., Klepac, P., Jit, M., Diamond, C., Gibbs, H., van Zandvoort, K.: Feasibility of controlling COVID-19 outbreaks by isolation of cases and contacts. Lancet Glob. Health **8**, e488–e496 (2020)
11. Nadler, P., Wang, S., Arcucci, R., Yang, X., Guo, Y.: An epidemiological modelling approach for COVID-19 via data assimilation. Eur. J. Epidemiol. **35**, 749–761 (2020)
12. Zhang, S., Wang, Z., Chang, R., Wang, H., Xu, C., Yu, X., Tsamlag, L., Dong, Y., Wang, H., Cai, Y.: COVID-19 containment: China provides important lessons for global response. Front. Med. **14**, 215–219 (2020)
13. Keeling, M.J., Hollingsworth, T.D. and Read, J.M.: The efficacy of contact tracing for the containment of the 2019 novel coronavirus (COVID-19) (2020)
14. novel investigation techniques for tracing contacts: Contact transmission of COVID-19 in South Korea. Osong Public Health and Research Perspectives **11**, 60–63 (2020)
15. Rastgoo, M.N., Nakisa, B., Nazri, M.Z.A.: A hybrid of modified PSO and local search on a multi-robot search system. Int. J. Adv. Robot. Syst. **12**, 86 (2015)
16. Couceiro, M.S., Rocha, R.P. and Ferreira, N.M.F.: A novel multi-robot exploration approach based on particle swarm optimization algorithms. In: 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics. IEEE (2011)
17. Tang, H., Sun, W., Yu, H., Lin, A., Xue, M., Song, Y.: A novel hybrid algorithm based on PSO and FOA for target searching in unknown environments. Appl. Intell. **49**, 2603–2622 (2019)
18. Zheng, Z., Li, J., Li, J. and Tan, Y.: Improved group explosion strategy for searching multiple targets using swarm robotics. In: 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE (2014)
19. Li, J. and Tan, Y.: The multi-target search problem with environmental restrictions in swarm robotics. In 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014). IEEE (2014)
20. Trianni, V. and Campo, A.: Fundamental collective behaviors in swarm robotics. In: Springer Handbook of Computational Intelligence, pp. 1377–1394, Springer, Berlin (2015)
21. Comment l'épidémiologie tente de cerner l'épidémie due au nouveau coronavirus. Le Monde (2020). Accessed 3 Apr 2020
22. Smith, M., Yourish, K., Almukhtar, S., Collins, K., Ivory, D. and Harmon, A.: Coronavirus (COVID-19) data in the United States. https://github.com/nytimes/covid-19-data. Accessed 20 Apr 2020
23. S. Company: United States Cities Database. https://simplemaps.com/data/us-cities. Accessed 22 Apr 2020
24. Larsen, L., Kim, J., Kupke, M., Schuster, A.: Automatic path planning of industrial robots comparing sampling-based and computational intelligence methods. Procedia Manuf. **11**, 241–248 (2017)
25. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings. 1985 IEEE International Conference on Robotics and Automation, Institute of Electrical and Electronics Engineers
26. Lumelsky, V., Stepanov, A.: Dynamic path planning for a mobile automaton with limited information on the environment. IEEE Trans. Autom. Control **31**, 1058–1063 (1986)
27. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics Autom. Mag. **4**, 23–33 (1997)
28. Jin, S. and Choi, B.-J.: Fuzzy logic system based obstacle avoidance for a mobile robot. In: Communications in Computer and Information Science, pp. 1–6. Springer, Berlin (2011)
29. Kelly, A., Stent, A.: Autonomous Robots **5**(2), 129–161 (1998)
30. Park, J., Iagnemma, K.: Sampling-based planning for maximum margin input space obstacle avoidance. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE (2015)
31. Khaksar, W., Sahari, K.S.M., Hong, T.S.: Application of sampling-based motion planning algorithms in autonomous vehicle navigation. In: Autonomous Vehicle. InTech (2016)
32. Bresenham, J.E.: Algorithm for computer control of a digital plotter. IBM Syst. J. **4**(1), 25–30 (1965)
33. Liu, Z., Magal, P., Seydi, O. and Webb, G.: Predicting the cumulative number of cases for the COVID-19 epidemic in china from early data (2020)
34. Bruce, A. et al.: Report of the who-china joint mission on coronavirus disease 2019 (COVID-19). Technical Report. World Health Organization, China, February 2020. Accessed 8 Apr 2020