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

Naila Aziza HOUACINE (nhouacine@usthb.dz)
USTHB - LRIA https://orcid.org/0000-0002-2240-7288
Habiba DRIAS
USTHB - LRIA https://orcid.org/0000-0001-7287-5170

Research Article

Keywords: Swarm Robotics, Swarm Intelligence, COVID-19, Target Detection Problem, Containment, Autonomous robots

Posted Date: August 17th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-59074/v1

Version of Record: A version of this preprint was published at Progress in Artificial Intelligence on May 11th, 2021. See the published version at https://doi.org/10.1007/s13748-021-00245-3.

manuscript No.
(will be inserted by the editor)

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

Naila Aziza HOUACINE · Habiba DRIAS

Received: date / Accepted: date

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 (SI) approaches, Swarm Robotics (SRs) 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 EHO (Elephants Herding Optimization) 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 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 Containment Rate, 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

N. A. Houacine
LRIA, USTHB, BP 32 El-Alia Bab-Ezzouar, Algiers, Algeria
E-mail: nhouacine@usthb.dz

H. Drias
LRIA, USTHB, BP 32 El-Alia Bab-Ezzouar, Algiers, Algeria
E-mail: hdrias@usthb.dz

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 likened to a problem of searching for targets in a complex and unknown environment, where the targets will be potentially affected individuals, and the search will 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 SRs, which are basically MRSs (Multi-Robots Systems) 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 and price by the application of SI algorithms [2,3].

This paper proposes an extension of EHO (Elephants Herding Optimization) [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 targets' 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 elephants' group for the position update and providing a collision-free path planning (Sampling-based) for environments of discrete nature, that aim to 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 present the modeling of the dynamic environment that mimics the virus spread and how we convert it to a TDP. In Section 4 we introduce our REHO search approach. Experimental results are presented and analyzed in Section 5. Finally, Section 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] and [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 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]. 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 traditional public health outbreak response tactics: isolation, quarantine, social distancing, and community containment. In the article [9] authors used a mathematical model to assess the virus evolution when applying isolation and contact tracing. Many other published papers insist on the importance of the containment. Besides, authors in [10] developed two modeling approaches to infer the growth rate of the outbreak.

As reported in [11], containment measures taken in China proved to reduce human-to-human transmission successfully. However, most of the previous 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 [9] 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 [12] and [13]. Currently, after determining the potentially infected people, certain governments take care of recovering their phone numbers in order to contact and notify them. However, people's unconsciousness and other parameters can lead to mission failure. People can ignore the call, they cannot or do not want to go to the hospital to take the test, or the risk of being contaminated when going to hospital and so on.

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 [14], 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 [15], and MFSO (Multi-swarm hybrid FOA-PSO) [16]. However, all these approaches have only been applied for unique target search in completely static environments. On the other hand, in [17], Multi-Target based methods such as IGES (Improved Group Explosion Strat-

egy) for searching multiple targets using Swarm Robotics were proposed. In the Multi-target Search Problem with Environmental Restrictions in Swarm Robotics [18], three algorithms are compared (IGES, GES and RPSO). 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. They all consider partially dynamic environments considering both static and dynamic obstacles, but still do not account for the dynamic evolution of targets' number.

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 [19].

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** has a square shape, 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 and the distance between the robot and a target. It is defined by Equation (1).

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

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 square's diagonal, and *Dist(x,y)* is the euclidean distance between a target and the current robot position, it's given by the Equation (2).

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

Since environment shapes are squares, *diagSize* is equal to 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, 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 range, as shown in Equation (3).

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

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 Equation (4).

$$R_{d,z} = R_0 * (1 - CR_{d,z}) \tag{4}$$

where R_0 is the estimated Reproduction rate without containment and $CR_{d,z}$ 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 Equation (5).

$$\#Target_d = R_d * \#Target_{d-1} \tag{5}$$

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 Figure 1 the greater is the Containment Rate, the slower the virus evolves, and the fewer individuals are infected.

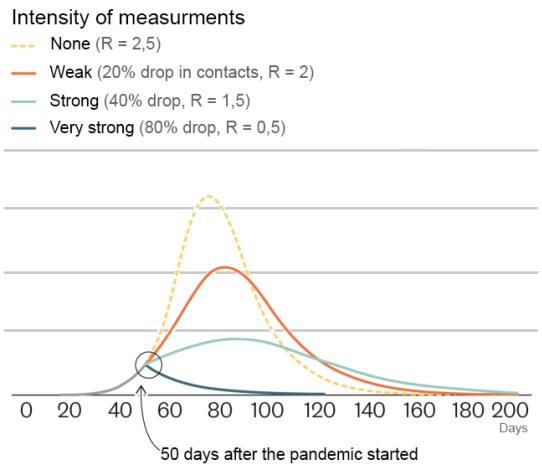

Fig. 1 Containment Rate impact on the number of new cases [20].

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 Equation (5) to calculate the size of the search space at successive times, we find Equation (6).

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

Then, according to Equation (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: $R_0 = R_1 = R_2 = ... = R_{d-1} = R_d$. And when we replace them in Equation (6) it will turn into Equation (7).

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

Here we conclude that the complexity in the worst case for this problem is exponential[1] and is given by Equation (8).

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

3.4 A Real case study: the case of 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 [21]. This Dataset is updated every day with the new 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. Then, we used a second dataset [22] 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.

Normalization of the coordinate positions to fit into a 5000 x 5000 squares representation requires the following steps. First, we get the USA Latitude [25, 50]

[1] when the value of R_0 is > 1

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 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 Equation (9) to define targets and borders positions.

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

Figure 2 depicts the result[2] 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 x 5000 squares size, in which dashed green rectangles are used to represent zones with different Containment Rate. The environment on the right illustrates the real case study, the USA territorial is represented surrounded by obstacles to limit the search area.

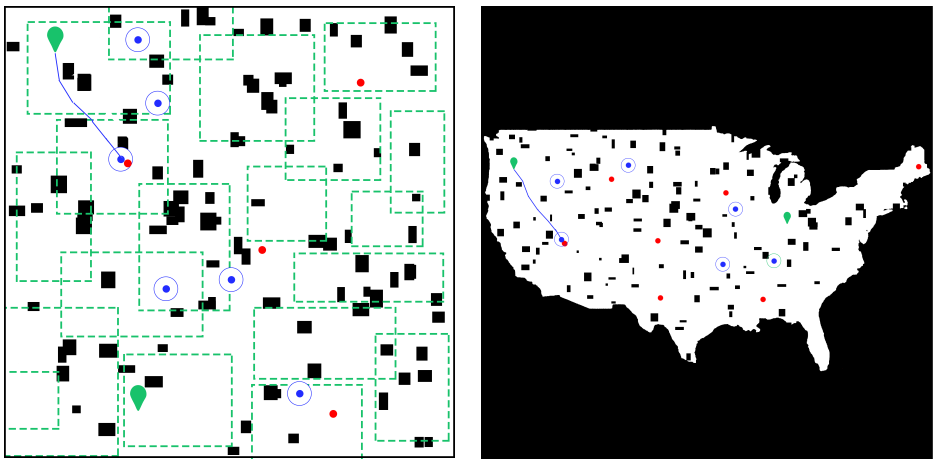

Fig. 2 Left: Random dataset. Right: USA data representation for the TDP.

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

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 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 Equation (10) for all elephants in each clan.

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

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

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

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

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

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

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 i^{th} clan and $X_{center,i}$: the gravity center of clan C_i.
$X_{i,j}$,$X_{new,i,j}$: respectively the actual and newly updated positions for elephant j in clan C_i.
$X_{best,i}$,$X_{worst,i}$: 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 Equation (14), which is originally based on Equation (10):

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

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

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

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 Equation (13), the robot with the worst fitness calculates its next position within Equation (16).

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

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 $(x_{new,i,j}, y_{new,i,j})$. 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 j^{th} robot of the i^{th} 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, α, β, $\#Target_0$, MaxTargets, MaxVelocity.
Output: PathList : List of paths from each target to robots 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 [23]. 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. *O. Khatib* was the first to propose a potential field method based on attractive force towards the goal and repulsive force of obstacles [24], *V. Lumelsky and A. Stepanov* presented two algorithms namely Bug1 and Bug2 exploiting touch sensors [25]. In [26] 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 [27] 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 [28], subsequently it was widely used and tested on real mobile robots [29,30].

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^{o}, being North-East, North-West, South-East, and South-West. The robot's view field is an angle of 90^{o} 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 [31] 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 = wp_1, wp_2, ... 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 experiments organization and the REHO's optimal parameter settings are presented, followed by the experimental results. These experiments are presented in four parts. Part 1, studies the impact of the initial reproduction rate R_0 of the virus. 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 are experimented under three scenarios of the Containment Rate evolution, which are : growing Containment Rate, randomly changing Containment Rate, and diminishing Containment Rate. Furthermore, robots' performances are compared in three environment sizes. Small environments with 500 x 500 squares, medium environments with 2 500 x 2 500 squares, and large environments of 5 000 x 5 000 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.

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 clan $\#Clans$ = 5, and the number of robots in each clan $\#Elephants$ = 4. Which corresponds to a population size of 5 x 4 = 20 robots. This method as many other Swarm Intelligence-based methods is depended 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 Section 3, examples of test environments are presented in Figure 2. The REHO approach, the ISS path planning method, and the generation of the environments have all been implemented in *Java*.

5.2 Influence of the initial reproduction rate : R_0

This part presents the experiment relatives to the initial reproduction rate R_0 . According to [32], the initial reproduction 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 seconds in small environments, 1.93 and 11.39 seconds 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 Figures 3 and 4. 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

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 (R_0).

	R_0	Small (500)			Medium (2 500)			Large (5 000)		
		Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
Growing CR	**1.5**	12.12	0.26	100	12.50	1.20	100	14.28	2.40	100
	2	12.70	0.26	100	13.58	1.26	100	16.52	2.76	100
	2.5	13.10	0.26	100	14.12	1.34	100	17.58	2.92	100
	3	13.40	0.27	100	14.10	1.53	100	18.42	3.24	100
	3.5	13.42	0.27	100	15.50	1.58	100	21.00	3.99	100
	4	14.06	0.29	100	16.86	1.62	100	21.88	4.14	100
	4.5	15.34	0.32	100	18.28	1.64	100	24.72	4.07	100
	5	18.00	0.45	100	20.64	1.93	100	43.68	11.39	100
Random CR	**1.5**	13.90	0.26	100	16.70	1.33	100	16.00	2.52	100
	2	12.98	0.26	100	16.24	1.35	100	16.63	2.77	100
	2.5	15.08	0.29	100	17.02	1.51	100	19.05	3.12	100
	3	14.96	0.32	100	18.56	1.69	100	19.08	3.42	100
	3.5	16.06	0.50	100	20.50	1.91	100	59.73	5.98	97
	4	17.08	0.37	100	111.1	4.75	94	287.3	10.87	87
	4.5	76.82	0.63	96	200.5	6.74	88	465.2	20.28	70
	5	194.8	1.03	88	496.6	12.08	69	659.2	23.26	60
Diminishing CR	**1.5**	17.22	0.30	100	24.00	1.69	100	91.92	7.30	96
	2	57.24	0.61	98	85.06	3.31	96	174.8	9.70	90
	2.5	51.98	0.53	98	297.6	8.24	82	674.8	26.14	57
	3	317.3	1.38	80	792.8	14.79	49	939.0	29.62	39
	3.5	407.4	1.64	74	881.7	19.40	43	1086.4	30.99	29
	4	611.9	2.01	60	1087.9	17.94	29	1235.1	31.8	19
	4.5	851.1	2.54	45	1265.1	18.53	17	1441.7	34.92	5
	5	878.3	2.48	42	1321.9	17.82	13	1440.5	32.97	5

small environments, where it reaches 496 and 659 iterations in medium and large environments, respectively. 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.

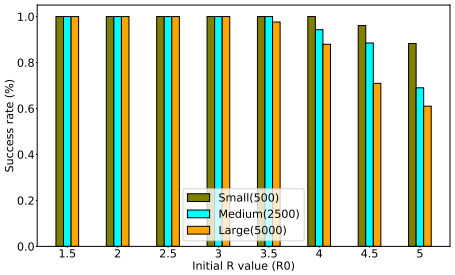

Fig. 3 Comparing the success rate in different environment sizes with increasing R_0 in a random CR update.

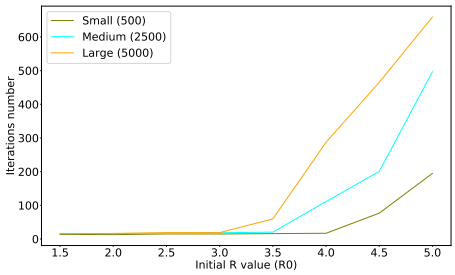

Fig. 4 Comparing the iterations number in different environment sizes with increasing R_0 in a random CR update.

For the worst possible Containment Rate evolution diminishing CR, we got the obtained results presented in Figures 5 and 6. 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.

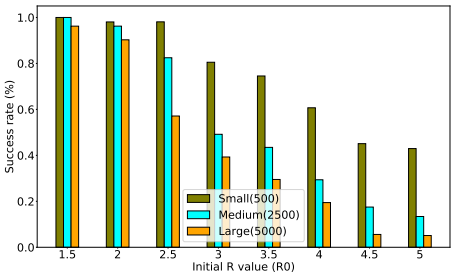

Fig. 5 Comparing the success rate in different environment sizes with increasing R_0 in a diminishing CR.

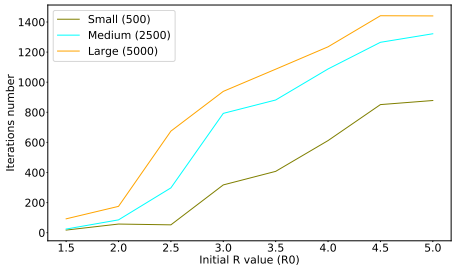

Fig. 6 Comparing the iterations number in different environment sizes with increasing R_0 in a diminishing CR.

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.

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

In the report [33], the Initial value of R_0 is estimated to [2 - 2.5] in China. According to [20], 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 Section 5.4. 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 seconds within small environment, whereas it performs 125 iterations in 108 seconds and 178 iterations in 318 seconds in medium and large environments, respectively, and this is only

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

	$\#Target_0$	**Small (500)**			**Medium (2 500)**			**Large (5 000)**		
		Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
Growing CR	**5**	2.92	0.01	100	3.58	0.06	100	3.45	0.13	100
	50	13.90	0.24	100	18.90	1.34	100	14.05	2.32	100
	100	20.46	0.65	100	34.08	4.58	100	33.55	8.81	100
	150	27.22	1.34	100	38.18	8.52	100	40.18	16.02	100
	200	30.20	2.17	100	43.56	12.95	100	47.65	28.60	100
	250	40.16	3.54	100	51.08	19.28	100	61.05	40.60	100
	300	47.20	4.79	100	64.48	29.48	100	74.55	65.24	100
	350	54.46	6.59	100	72.22	40.01	100	81.20	85.89	100
	400	61.9	8.53	100	86.84	61.46	100	103.7	135.1	100
	500	90.50	15.61	100	125.4	108.7	100	178.6	318.1	100
Random CR	**5**	2.60	0.01	100	4.20	0.07	100	3.30	0.11	100
	50	13.22	0.25	100	20.50	1.49	100	13.45	2.14	100
	100	19.28	0.67	100	33.48	4.52	100	32.43	8.92	100
	150	29.10	1.42	100	36.12	8.49	100	38.63	16.11	100
	200	30.40	2.25	100	45.48	13.09	100	49.88	28.89	100
	250	41.34	3.58	100	51.58	19.62	100	56.93	39.53	100
	300	46.36	4.70	100	65.78	31.90	100	84.55	68.69	100
	350	56.04	6.88	100	77.86	43.38	100	96.05	102.8	100
	400	66.70	9.03	100	104.5	72.27	100	296.9	199.6	89
	500	170.3	22.05	96	1157.9	230.2	37	1466.8	458.0	17
Diminishing CR	**5**	3.02	0.01	100	3.38	0.07	100	3.68	0.13	100
	50	12.96	0.24	100	22.40	1.51	100	13.70	2.26	100
	100	19.12	0.65	100	43.78	4.93	100	68.90	9.82	100
	150	29.78	1.36	100	44.24	9.27	100	53.56	16.67	100
	200	32.48	2.24	100	56.76	14.08	100	64.32	30.59	100
	250	43.38	3.50	100	72.12	58.62	100	155.3	48.75	100
	300	55.34	5.10	100	142.4	23.21	100	451.0	156.6	77
	350	68.82	7.72	100	190.7	49.11	96	762.0	263.4	60
	400	97.20	11.75	100	878.8	141.4	51	1421.8	347.5	14
	500	631.1	32.60	68	1313.0	165.4	21	1500	247.4	7

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 Figures 7 and 8, the success rate is only affected when starting 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 seconds within small environments, and around 1157 iterations in 230 seconds and 1466 iterations in 458 seconds 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, Figures 9 and 10 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

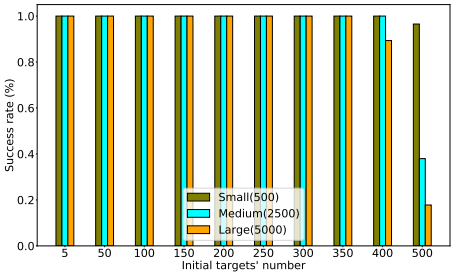

Fig. 7 Comparing the success rate in different environment sizes with increasing $\#Target_0$ in a random CR update.

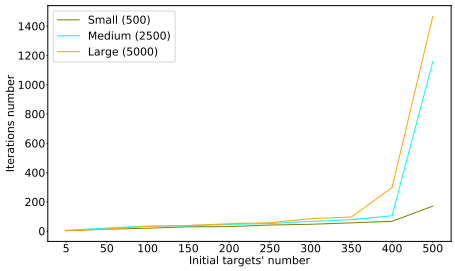

Fig. 8 Comparing the iterations number in different environment sizes with increasing $\#Target_0$ in a random CR update.

starts when dealing with 350 and 300 initial number of targets, respectively. These difficulties are expressed by the decreasing success rate at finding all 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 seconds, 1313 iterations in 165 seconds and 1500 iterations in 247 seconds within small, medium and large environments, respectively.

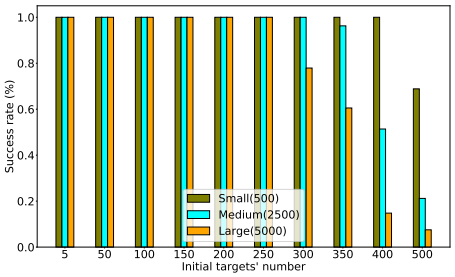

Fig. 9 Comparing the success rate in different environment sizes with increasing $\#Target_0$ in a diminishing CR.

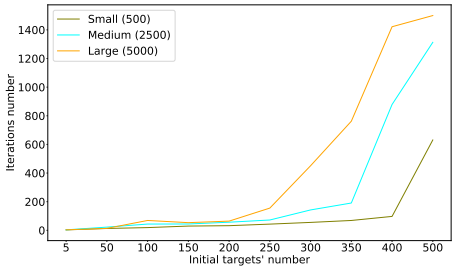

Fig. 10 Comparing the iterations number in different environment sizes with increasing $\#Target_0$ in a diminishing CR.

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.

5.4 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.

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 (2 500)**			**Large (5 000)**		
		Iter	Time	Succ	Iter	Time	Succ	Iter	Time	Succ
Growing CR	**1**	49.18	7.07	100	53.56	35.28	100	107.9	177.1	100
	2	23.60	1.17	100	33.34	11.14	100	62.10	72.02	100
	5	13.20	0.27	100	19.56	1.83	100	20.54	2.99	100
	10	12.16	0.23	100	15.66	1.28	100	16.62	2.28	100
	15	13.20	0.23	100	14.20	1.25	100	16.16	2.16	100
	20	13.06	0.23	100	14.68	1.21	100	15.46	2.15	100
	25	13.34	0.23	100	13.94	1.32	100	17.06	2.11	100
	30	12.92	0.26	100	14.50	1.26	100	16.96	2.18	100
	35	12.48	0.25	100	13.50	1.27	100	15.34	2.16	100
	40	13.88	0.24	100	14.58	1.20	100	17.30	2.15	100
Random CR	**1**	1500	1.42	0	1500	6.58	0	1500	12.44	0
	2	1500	3.14	1	1500	14.11	1	1500	27.30	1
	5	14.58	0.31	100	20.54	2.02	100	18.13	2.89	100
	10	12.42	0.25	100	17.66	1.41	100	14.90	2.19	100
	15	12.90	0.24	100	18.34	1.39	100	13.43	2.17	100
	20	12.92	0.24	100	19.24	1.39	100	13.63	2.13	100
	25	13.38	0.24	100	20.42	1.41	100	13.10	2.13	100
	30	14.00	0.25	100	21.88	1.42	100	14.63	2.16	100
	35	12.18	0.24	100	21.14	1.38	100	14.05	2.19	100
	40	12.90	0.24	100	21.70	1.41	100	14.65	2.20	100
Diminishing CR	**1**	1500	0.98	0	1500	4.65	0	1500	8.83	0
	2	1500	2.03	1	1500	9.35	0	1500	18.27	0
	5	78.68	0.62	96	264.8	7.33	84	586.1	24.06	62
	10	13.76	0.25	100	28.28	1.79	100	17.42	2.51	100
	15	12.04	0.23	100	24.08	1.58	100	13.48	2.28	100
	20	12.66	0.24	100	22.54	1.55	100	14.56	2.48	100
	25	12.58	0.24	100	24.92	1.56	100	12.90	2.35	100
	30	13.02	0.24	100	21.48	1.48	100	13.04	2.44	100
	35	14.04	0.24	100	23.78	1.61	100	14.00	2.36	100
	40	12.82	0.24	100	21.18	1.52	100	14.42	2.16	100

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 seconds in small environments. While, it takes 53 iterations in 35 seconds and 107 iterations in 177 seconds within medium and large environments, respectively.

Considering the second third of Table 3, we are able to draw Figures 11 and 12 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.

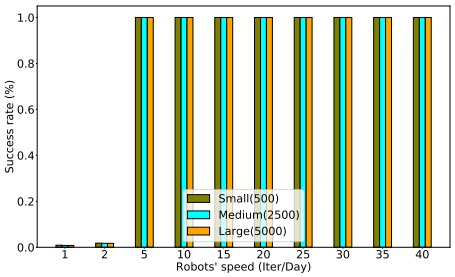

Fig. 11 Comparing the success rate in different environment sizes with increasing robots' speed (IterDay) in a random CR updating.

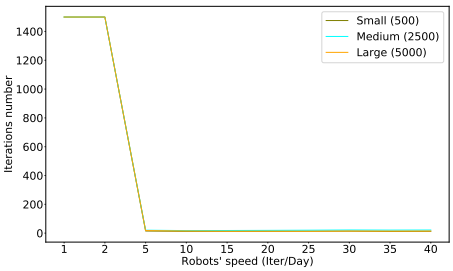

Fig. 12 Comparing the iterations number in different environment sizes with increasing robots' speed (IterDay) in a random CR updating.

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 Figures 13 and 14 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 second in small environments, and an average of 21 iterations in 1 second and 14 iterations in 2 seconds 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.

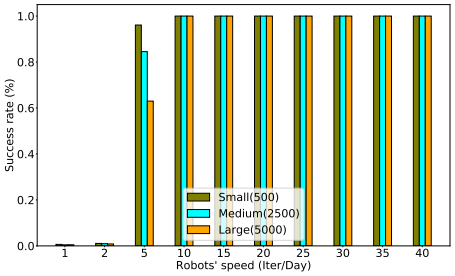

Fig. 13 Comparing the success rate in different environment sizes with increasing robots' speed (IterDay) in a diminishing CR.

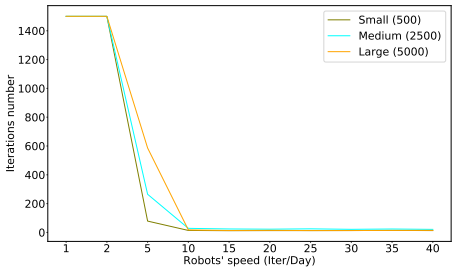

Fig. 14 Comparing the iterations number in different environment sizes with increasing robots' speed (IterDay) in a diminishing CR.

5.5 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.

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
21/01/2020	1.00	0.02	100	1.00	0.02	100	1.00	0.02	100
26/01/2020	2.60	0.05	100	2.70	0.06	100	2.80	0.06	100
31/01/2020	2.67	0.07	100	3.03	0.07	100	2.77	0.07	100
05/02/2020	4.67	0.15	100	4.93	0.16	100	4.87	0.16	100
10/02/2020	4.67	0.19	100	4.87	0.19	100	4.73	0.18	100
15/02/2020	4.67	0.20	100	4.73	0.19	100	5.37	0.21	100
20/02/2020	7.17	0.43	100	7.57	0.43	100	6.87	0.40	100
25/02/2020	11.03	1.05	100	9.40	0.95	100	10.17	1.02	100
01/03/2020	1500	158.4	9	18.03	3.59	100	17.50	3.44	100
05/03/2020	1500	175.2	9	1500	554.4	21	1500	913.2	37

From Figures 15, It can be observed that the REHO search approach makes a 100% success rate when intervening before 01/03/2020 no matter its robots' speed. However, from 01/03/2020, 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 05/03/2020, they get 21% and 37%, respectively.

Figure 16, shows that the REHO approach provides very few efforts (less than 12 iterations in less than 4 seconds) 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.

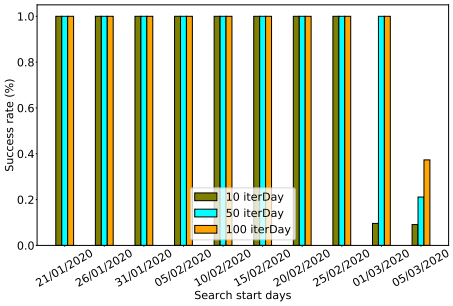

Fig. 15 Comparing the success rate with different search start days and different robots' speed.

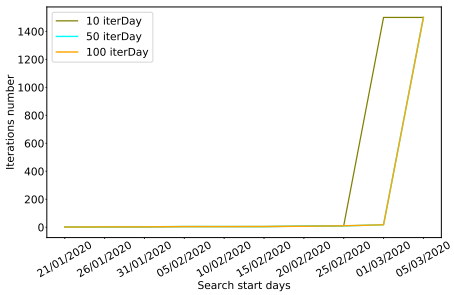

Fig. 16 Comparing the iterations number with different start search days and different robots' speed.

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.

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, the REHO approach offers better performances when the containment is abided 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 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.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. C. Robin, S. Lacroix, Autonomous Robots **40**(4), pp.729 (2016). URL https://hal.archives-ouvertes.fr/hal-01183372
2. S. Lu, Y. Hao, in *3rd International Conference on Electromechanical Control Technology and Transportation* (SCITEPRESS - Science and Technology Publications, 2018). DOI 10.5220/0006972904760486. URL https://doi.org/10.5220/0006972904760486
3. M. Dadgar, S. Jafari, A. Hamzeh, Neurocomputing **177**, 62 (2016). DOI 10.1016/j.neucom.2015.11.007. URL https://doi.org/10.1016/j.neucom.2015.11.007
4. G.G. Wang, S. Deb, L. dos S. Coelho, in *2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)* (IEEE, 2015). DOI 10.1109/iscbi.2015.8. URL https://doi.org/10.1109/iscbi.2015.8
5. G.G. Wang, L.D.S. Coelho, X.Z. Gao, S. Deb, International Journal of Bio-Inspired Computation **8**(6), 394 (2016). DOI 10.1504/ijbic.2016.10002274. URL https://doi.org/10.1504/ijbic.2016.10002274
6. S. Flaxman, S. Mishra, A. Gandy, H. Unwin, H. Coupland, T. Mellan, H. Zhu, T. Berah, J. Eaton, P. Perez Guzman, N. Schmit, L. Cilloni, K. Ainslie, M. Baguelin, I. Blake, A. Boonyasiri, O. Boyd, L. Cattarino, C. Ciavarella, L. Cooper, Z. Cucunuba Perez, G. Cuomo-Dannenburg, A. Dighe, A. Djaafara, I. Dorigatti, S. Van Elsland, R. Fitzjohn, H. Fu, K. Gaythorpe, L. Geidelberg, N. Grassly, W. Green, T. Hallett, A. Hamlet, W. Hinsley, B. Jeffrey, D. Jorgensen, E. Knock, D. Laydon, G. Nedjati Gilani, P. Nouvellet, K. Parag, I. Siveroni, H. Thompson, R. Verity, E. Volz, C. Walters, H. Wang, Y. Wang, O. Watson, P. Winskill, X. Xi, C. Whittaker, P. Walker, A. Ghani, C. Donnelly, S. Riley, L. Okell, M. Vollmer, N. Ferguson, S. Bhatt, (2020). DOI 10.25561/77731. URL http://spiral.imperial.ac.uk/handle/10044/1/77731
7. Q. Lin, S. Zhao, D. Gao, Y. Lou, S. Yang, S.S. Musa, M.H. Wang, Y. Cai, W. Wang, L. Yang, D. He, International Journal of Infectious Diseases **93**, 211 (2020). DOI 10.1016/j.ijid.2020.02.058. URL https://doi.org/10.1016/j.ijid.2020.02.058
8. Z. Wu, J.M. McGoogan, JAMA **323**(13), 1239 (2020). DOI 10.1001/jama.2020.2648. URL https://doi.org/10.1001/jama.2020.2648
9. J. Hellewell, S. Abbott, A. Gimma, N.I. Bosse, C.I. Jarvis, T.W. Russell, J.D. Munday, A.J. Kucharski, W.J. Edmunds, S. Funk, R.M. Eggo, F. Sun, S. Flasche, B.J. Quilty, N. Davies, Y. Liu, S. Clifford, P. Klepac, M. Jit, C. Diamond, H. Gibbs, K. van Zandvoort, The Lancet Global Health **8**(4), e488 (2020). DOI 10.1016/s2214-109x(20)30074-7. URL https://doi.org/10.1016/s2214-109x(20)30074-7
10. S. Steven, T.L. Yen, X. Chonggang, R.S. Ethan, H. Nick, K. Ruian, Emerging Infectious Diseases **26**(7) (2020). DOI 10.3201/eid2607.200282. URL https://doi.org/10.3201/eid2607.200282
11. S. Zhang, Z. Wang, R. Chang, H. Wang, C. Xu, X. Yu, L. Tsamlag, Y. Dong, H. Wang, Y. Cai, Frontiers of Medicine **14**(2), 215 (2020). DOI 10.1007/s11684-020-0766-9. URL https://doi.org/10.1007/s11684-020-0766-9
12. M.J. Keeling, T.D. Hollingsworth, J.M. Read, (2020). DOI 10.1101/2020.02.14.20023036. URL https://doi.org/10.1101/2020.02.14.20023036

13. Osong Public Health and Research Perspectives **11**(1), 60 (2020). DOI 10.24171/j.phrp.2020.11.1.09. URL https://doi.org/10.24171/j.phrp.2020.11.1.09
14. M.N. Rastgoo, B. Nakisa, M.Z.A. Nazri, International Journal of Advanced Robotic Systems **12**(7), 86 (2015). DOI 10.5772/60624. URL https://doi.org/10.5772/60624
15. M.S. Couceiro, R.P. Rocha, N.M.F. Ferreira, in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics* (IEEE, 2011). DOI 10.1109/ssrr.2011.6106751. URL https://doi.org/10.1109/ssrr.2011.6106751
16. H. Tang, W. Sun, H. Yu, A. Lin, M. Xue, Y. Song, Applied Intelligence **49**(7), 2603 (2019). DOI 10.1007/s10489-018-1390-0. URL https://doi.org/10.1007/s10489-018-1390-0
17. Z. Zheng, J. Li, J. Li, Y. Tan, in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (IEEE, 2014). DOI 10.1109/smc.2014.6973915. URL https://doi.org/10.1109/smc.2014.6973915
18. J. Li, Y. Tan, in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)* (IEEE, 2014). DOI 10.1109/robio.2014.7090748. URL https://doi.org/10.1109/robio.2014.7090748
19. V. Trianni, A. Campo, in *Springer Handbook of Computational Intelligence* (Springer Berlin Heidelberg, 2015), pp. 1377–1394. DOI 10.1007/978-3-662-43505-2_71. URL https://doi.org/10.1007/978-3-662-43505-2_71
20. Le Monde (2020). URL https://www.lemonde.fr/sciences/article/2020/03/30/comment-l-epidemiologie-tente-de-cerner-l-epidemie-due-au-nouveau-coronavirus_6034947_1650684.html. Accessed: April 3, 2020
21. M. Smith, K. Yourish, S. Almukhtar, K. Collins, D. Ivory, A. Harmon. Coronavirus (covid-19) data in the united states. https://github.com/nytimes/covid-19-data. Accessed: April 20, 2020
22. S. Company. United states cities database. https://simplemaps.com/data/us-cities. Accessed: 2020-04-22
23. L. Larsen, J. Kim, M. Kupke, A. Schuster, Procedia Manufacturing **11**, 241 (2017). DOI 10.1016/j.promfg.2017.07.237. URL https://doi.org/10.1016/j.promfg.2017.07.237
24. O. Khatib, in *Proceedings. 1985 IEEE International Conference on Robotics and Automation* (Institute of Electrical and Electronics Engineers). DOI 10.1109/robot.1985.1087247. URL https://doi.org/10.1109/robot.1985.1087247
25. V. Lumelsky, A. Stepanov, IEEE Transactions on Automatic Control **31**(11), 1058 (1986). DOI 10.1109/tac.1986.1104175. URL https://doi.org/10.1109/tac.1986.1104175
26. D. Fox, W. Burgard, S. Thrun, IEEE Robotics & Automation Magazine **4**(1), 23 (1997). DOI 10.1109/100.580977. URL https://doi.org/10.1109/100.580977
27. S. Jin, B.J. Choi, in *Communications in Computer and Information Science* (Springer Berlin Heidelberg, 2011), pp. 1–6. DOI 10.1007/978-3-642-26010-0_1. URL https://doi.org/10.1007/978-3-642-26010-0_1
28. A. Kelly, A. Stentz, Autonomous Robots **5**(2), 129 (1998). DOI 10.1023/a:1008801421636. URL https://doi.org/10.1023/a:1008801421636
29. J. Park, K. Iagnemma, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2015). DOI 10.1109/iros.2015.7353651. URL https://doi.org/10.1109/iros.2015.7353651
30. W. Khaksar, K.S.M. Sahari, T.S. Hong, in *Autonomous Vehicle* (InTech, 2016). DOI 10.5772/64730. URL https://doi.org/10.5772/64730
31. J.E. Bresenham, IBM Systems Journal **4**(1), 25 (1965). DOI 10.1147/sj.41.0025. URL https://doi.org/10.1147/sj.41.0025
32. Z. Liu, P. Magal, O. Seydi, G. Webb, (2020). DOI 10.20944/preprints202002.0365.v1. URL https://doi.org/10.20944/preprints202002.0365.v1
33. A. Bruce, L. Wannian, D. Xiaoping, E. Tim, F. Dale, I. Chikwe, L. Clifford, L. Jong-Koo, L. Gabriel, L. Jiangtao, L. Haiying, P. Natalia, S. Aleksandr, T. Hitoshi, V.K. Maria, W. Bin, W. Guangfa, W. Fan, W. Zhongze, W. Zunyou, X. Jun, Y. Kwok-Yung, Z. Weigong, Z. Yong, Z. Lei, Report of the who-china joint mission on coronavirus disease 2019 (covid-19). Tech. rep., World Health Organization, China (2020). URL https://www.who.int/docs/ default-source/coronaviruse/who-china-joint-mission-on-covid-19-final-report.pdf. Accessed: April 8, 2020

Figures

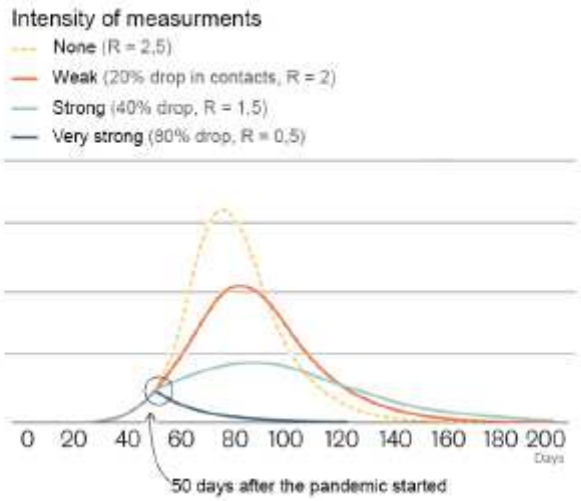

Figure 1

Containment Rate impact on the number of new cases [20].

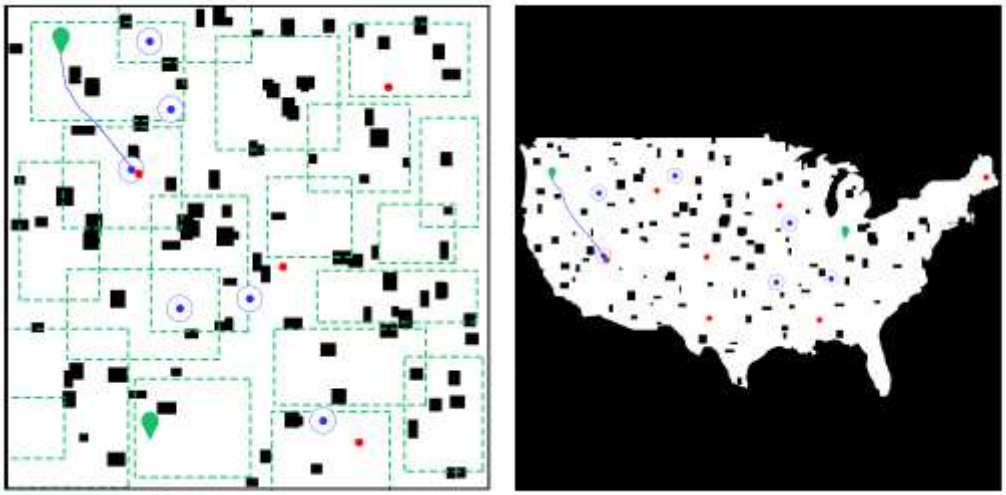

Figure 2

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

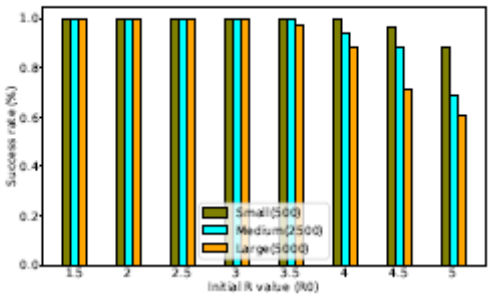

Figure 3

Comparing the success rate in di er- ent environment sizes with increasing R0 in a random CR update.

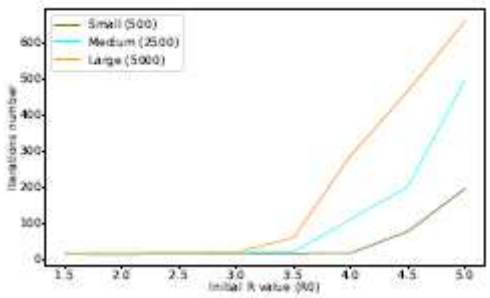

Figure 4

Comparing the iterations number in di erent environment sizes with increasing R0 in a random CR update.

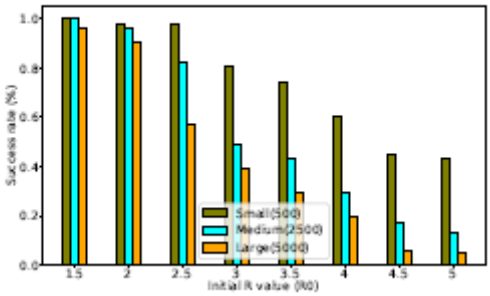

Figure 5

Comparing the success rate in different environment sizes with increasing R0 in a diminishing CR.

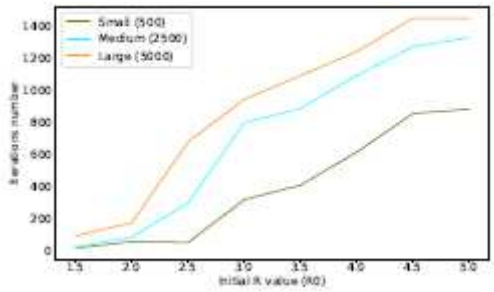

Figure 6

Comparing the iterations number in different environment sizes with increasing R0 in a diminishing CR.

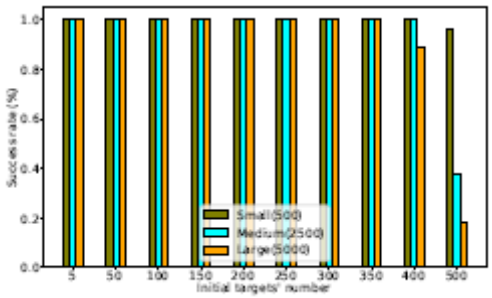

Figure 7

Comparing the success rate in different environment sizes with increasing #Target0 in a random CR update.

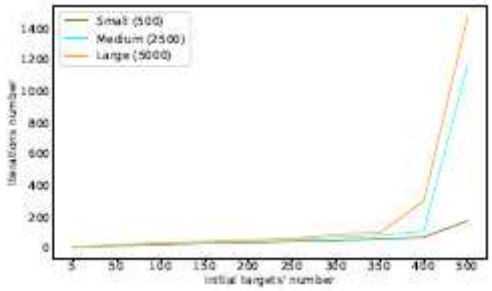

Figure 8

Comparing the iterations number in different environment sizes with increasing #Target0 in a random CR update.

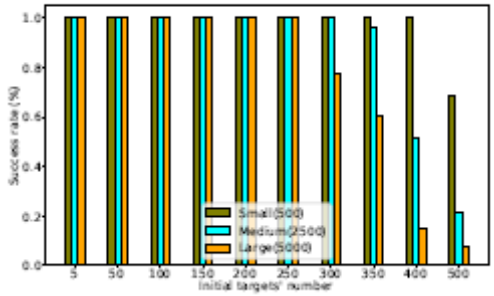

Figure 9

Comparing the success rate in different environment sizes with increasing #Target0 in a diminishing CR.

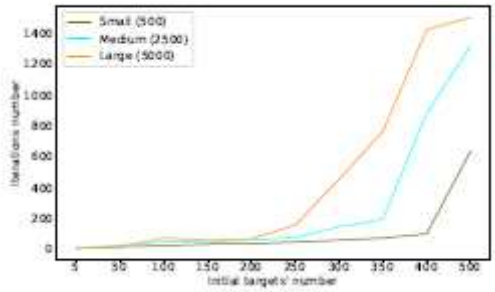

Figure 10

Comparing the iterations number in different environment sizes with increasing #Target0 in a diminishing CR.

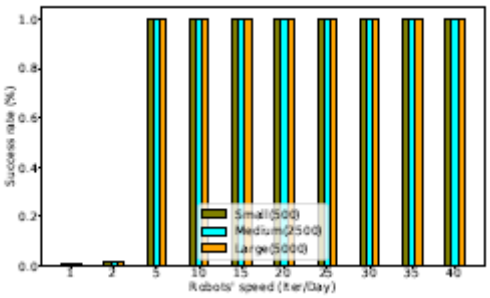

Figure 11

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

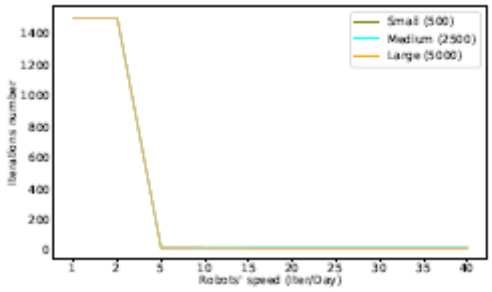

Figure 12

Comparing the iterations number in different environment sizes with increasing robots' speed (IterDay) in a random CR up- dating.

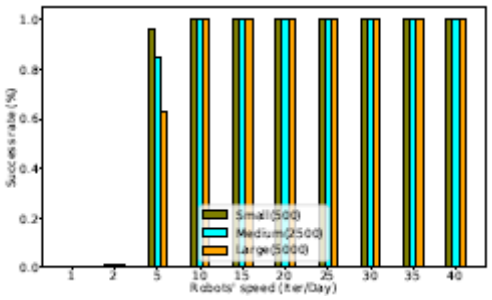

Figure 13

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

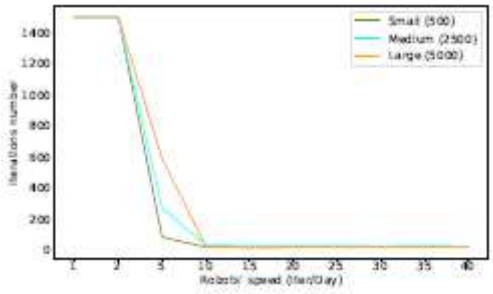

Figure 14

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

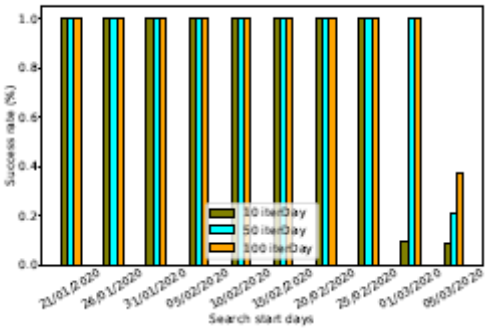

Figure 15

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

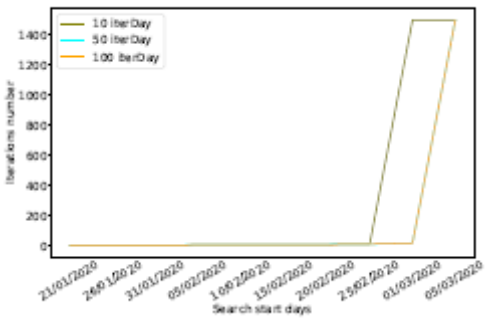

Figure 16

Comparing the iterations number with different start search days and di erent robots' speed.