Distributed Embodied Evolution in Networks of Agents

Anil Yaman[1,2][0000−0003−1379−3778] and Giovanni Iacca[3][0000−0001−9723−1830]

[1] Department of Bio and Brain Engineering, Korea Advanced Institute of Science and Technology, Daejeon, 34141, Republic of Korea
[2] Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, 5612 AP, The Netherlands
[3] Department of Information Engineering and Computer Science, University of Trento, Trento, 38122, Italy
a.yaman@tue.nl, giovanni.iacca@unitn.it

Abstract. In most network problems, the optimum behaviors of agents in the network are not known before deployment. In addition to that, agents might be required to adapt, i.e. change their behavior based on the environment conditions. In these scenarios, offline optimization is usually costly and inefficient, while online methods might be more suitable. In this work we propose a distributed embodied evolutionary approach to optimize spatially distributed, locally interacting agents by allowing them to exchange their behavior parameters and learn from each other to adapt to a certain task within a given environment. Our numerical results show that the local exchange of information, performed by means of crossover of behavior parameters with neighbors, allows the network to converge to the global optimum more efficiently than the cases where local interactions are not allowed, even when there are large differences on the optimal behaviors within a neighborhood.

Keywords: Embodied evolution · Distributed evolution · Networks.

1 Introduction

Networks of agents, such as sensor networks, wireless networks, swarm of drones or terrestrial robots, etc. are used nowadays in many tasks in the context of environment exploration, monitoring, and Internet of Things (IoT) applications. Typically, the behavior of the network as a whole derives from the local behavior of each single agent. However, modelling the mutual interactions of such agent-local behaviors (which can be very different across different parts of the network), as well as the interactions of the agents with the environment, can be difficult. In fact, the environment conditions might not be known *a priori*, i.e. before deployment, and may change dynamically and unexpectedly. As such, finding the optimal behavior of network agents offline, analytically, can be challenging: on the one hand, an analytical formulation is rarely available; on the other, collecting all the agents to tune their parameters when needed can can be inefficient -and expensive- especially when the number of agents is large. Moreover, in some cases the agents might not even be accessible for collection or data

extraction, e.g. if their position is not known or they are placed in hard-to-access environments such as pipes, underground tunnels, or collapsed buildings.

For all these reasons, a large body of literature has investigated various concepts such as *autonomic*, or *self-adaptive* networking [1, 2], in the quest for solutions to make networks capable of adapting automatically to the environment and thus optimizing autonomously, at runtime, their behavior. In this area, bio-inspired techniques -especially distributed Evolutionary Algorithms (EAs)- have attracted a great attention [3, 4] due to their intrinsic adaptivity. It is worth mentioning though that the idea of distributed EAs was initially introduced mainly in the context of numerical optimization, in the form of structured populations of solutions evolved in parallel over multiple cores or over computing networks [5–7]. A special flavour of these algorithms is Cellular Evolution (CE) [8–10], where a population of individuals is restricted to interact (i.e. through mating) locally. This can help preserve the population diversity longer. Of note, in CE, all individuals optimize the same objective function.

On the other hand, more recently distributed EAs have been used also in physical or simulated networks to evolve the agent-local parameters, rather than using the network as a collection of computing nodes to be used to solve an offline problem. One field of application of this concept is Wireless Sensor Networks (WSNs), for which previous works have proposed distributed Evolutionary Algorithms [11, 12] and distributed Genetic Programming [13, 14] to evolve the sensor nodes' parameters and functioning logic. However, robotics is by far the area where distributed EAs have shown their greatest potential so far. In this context, Embodied Evolution (EE), i.e. the idea that the evolutionary algorithm runs distributedly on a group of agents, has been successfully applied to optimize the behavior of robotic swarms [15–17]. In particular, *environment-driven* EE, a form of artificial ecology where the environment conditions guide the evolutionary process of a collective of agents, has emerged as a promising mechanism to obtain truly self-adaptive swarms [18]. As such, environment-driven EE is currently a very active area of research, not only on algorithmic aspects [18–23], but also from an application perspective: for instance, an interesting application on indoor surveillance and location has been proposed in [24, 25].

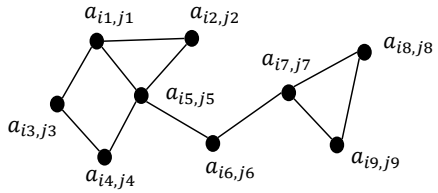


Fig. 1. An illustration of a network of spatially distributed, locally connected agents.

In this work, we continue this research stream by investigating the effect of local *information exchange* on distributed embodied evolution. More specifically, we consider a network of agents distributed at fixed locations within a certain environment, see Figure 1 for an example. We assume that each agent is required to perform a certain local behavior, that is described by some agent-local parameters, and that in general the optimal local behaviors are different across agents as they depend on the local environmental conditions. On the other hand, in some

cases the behaviors of the agents in close proximity can be similar, since they may share similar environmental conditions. We also assume that the parameters for the optimal local behaviors is not known at the time of deployment. In our setup, the agents are required to optimize their local parameters (i.e., learn their optimal behavior) autonomously while they operate within the environment. To achieve that, we propose a distributed embodied EE where each agent runs a population-less EA *in situ*, to optimize its own behavior. In contrast to CE, we assume that the fitness function of each agent can be different.

We introduce local information exchange in terms of exchange of behavior parameters with local neighbors: we consider various kinds of information exchange such as copying the parameters from the best (or a random) agent in the neighborhood with some mutation and/or exchange the parameters partially, i.e. by means of crossover of parameters with neighbors. We hypothesize that using information exchange as crossover and local perturbations for mutation leads to a social learning process through which it is possible to adapt a network of agents to a task at runtime, and in a distributed fashion. While it may be intuitive to exchange information with neighbors when their local conditions are similar, we are also interested in investigating the effect of information exchange in the cases where the optimal behaviors of the neighboring agents are drastically different. To assess that, we devise two experimental scenarios -that we refer to as imitation and illumination problems- with different network densities and different levels of similarities in terms of optimal behavior within a neighborhood.

Overall, our numerical results show that local information exchange is beneficial in any case -even with large differences of the optimal parameters within a neighborhood- while the optimization performance can be largely affected by the frequency of the parameter exchange, and the amount of exchanged parameters.

The remaining of this paper is structured as follows. In the next section, we describe the proposed distributed embodied evolution algorithm. Then, we present the experimental setup and the numerical results. Finally, we give the conclusions and highlight possible future research directions.

2 Methods

Without loss of generality, we consider a system composed of a collection of k agents, spatially distributed on a 2-dimensional plane at fixed locations i, j, with i and $j \in \mathbb{N}_0$. Each agent $a_{i,j}$ must show a behavior such that it optimizes a certain local function. The performance of each agent is measured by its local fitness value $f_{i,j}$, that measures how good is the agent's behavior w.r.t. the desired local function. In this paper, we formulate all problems as *minimization problems* (the lower fitness, the better). Furthermore, each agent can communicate (bidirectionally) with its local neighbors defined by a neighborhood function N. An example of such a system is shown in Figure 1: for instance, the neighborhood function of $a_{i5,j5}$ is defined by $N = \{(i1, j1), (i2, j2), (i3, j3), (i4, j4), (i6, j6)\}$, i.e. the set of indices of the agents that are connected to $a_{i5,j5}$.

We assume that the behavior of an agent $a_{i,j}$ can be encoded using a certain representation $\boldsymbol{X}^{i,j}$, that we refer to as the *genotype* of the agent. Here, we

adopt the terminology from biological evolution. Otherwise, $\boldsymbol{X}^{i,j}$ could represent a cultural phenomenon, an idea or a meme from the cultural evolution point of view. For the sake of simplicity, we assume that $\boldsymbol{X}^{i,j}$ is represented as a D-dimensional vector where each element is real-valued, defined in a given range $[lb, ub]$, and is referred to as a *gene*. The fitness value $f_{i,j}$ of $a_{i,j}$ is measured as $f_{i,j} \leftarrow eval(\boldsymbol{X}^{i,j})$ where $eval()$ is a function that evaluates the genotype of the agent for a certain period, based on certain performance metric. Here, it is assumed that each agent can compute its own (local) fitness function.

We employ an embodied evolutionary approach [17] where each agent runs its own (population-less) evolutionary algorithm to optimize its own behavior. This can be achieved with or without information sharing (i.e. sharing all or some of the elements of their genotypes) with the neighboring agents. In the case where there is no information sharing, each agent can iteratively test a randomly perturbed version of its genotype, and store the genotype that performs the best (this approach can be referred to as Hill Climbing). In the case where information sharing is allowed, the agents can copy and/or perform crossover with the genotypes of the neighboring agents. As we said, each agent aims to optimize its own behavior, which may be different from the neighboring agents. Therefore, the possible advantages/disadvantages of sharing information in this condition is one of the main focus of this work.

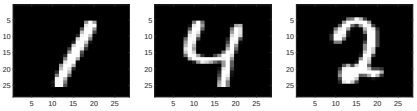
We define two helper functions, $best(\boldsymbol{X}^{i,j})$ and $rand(\boldsymbol{X}^{i,j})$, that return respectively the genotype of the best and the genotype of a (uniformly selected) random agent in the neighborhood of a given agent $\boldsymbol{X}^{i,j}$. In real-world scenarios, this may be implemented by allowing each agent to communicate its fitness and genotype with its neighbors. In this case, and assuming that the local fitness values are comparable, it would be possible for each agent to pick the best genotype, or a random genotype, to perform evolutionary operators.

Finally, we use two evolutionary operators, namely *uniform crossover* and *Gaussian mutation*, applied in this order. The crossover operator generates a new genotype $\boldsymbol{X}'$ from two given genotypes (parents), $\boldsymbol{X}^{i,j}$ and $\boldsymbol{X}^{k,l}$, as follows: first, $\boldsymbol{X}^{k,l}$ is copied into $\boldsymbol{X}'$; then, each element of $\boldsymbol{X}^{i,j}$ is copied into the corresponding element of $\boldsymbol{X}'$ with a probability of $cr \in [0, 1]$. The crossover probability $cp \in [0, 1]$ specifies the probability of performing the crossover operator. If the genotype of the agents consist of a single parameter, then we use *arithmetic crossover*, that computes the mean of the parents' parameters, i.e. $x' = (x^{i,j} + x^{k,l})/2$ (note that in this case cr is not needed). After crossover, the mutation operator perturbs each element of $\boldsymbol{X}'$ using a Gaussian mutation (sampled independently for each element) $\mathcal{N}(0, \sigma)$ with zero mean and standard deviation σ (in the following, we will refer to σ as mutation rate mr).

For illustration purposes, we provide in Algorithm 1 a pseudo-code of a version of the algorithm in which we apply crossover with a random neighbor. We dub this algorithm as XoverRand. The procedure $randomInitialize()$ indicates a random initialization of the initial behavior parameters $\boldsymbol{X}^{i,j}$, where each element is uniformly sampled within its range $[lb, ub]$, while g and $maxGenerations$ indicate, respectively, the current and the maximum number of generations.

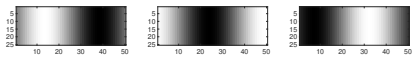
Algorithm 1 Embodied evolution of an agent $a_{i,j}$ (and its corresponding behavior $\boldsymbol{X}^{i,j}$) with crossover applied with a random neighbor (XoverRand).

```
procedure Evolve
    X^{i,j} ← randomInitialize()
    f_{i,j} = eval(X^{i,j})
    g = 0
    while g < maxGenerations do
        X' ← xover(X^{i,j}, rand(X^{i,j}), cp, cr)
        X' ← X' + N(0, σ)                    ▷ Mutation
        f' ← eval(X')
        if f' < f_{i,j} then                 ▷ Minimization
            f_{i,j} ← f'
            X^{i,j} ← X'
        end if
        g = g + 1
    end while
end procedure
```



(a) $t = 1$ (b) $t = 50$ (c) $t = 100$

Fig. 2. Imitation problem: Visualization of three images (ground truth) indexed at $t = 1$, $t = 50$ and $t = 100$.



(a) $t = 0$ (b) $t = 7$ (c) $t = 15$

Fig. 3. Illumination problem: Visualization of the optimal illumination (ground truth) indexed at $t = 0$, $t = 7$ and $t = 15$.

3 Experimental setup

In this section, we provide the details of our experimental setup. We are mainly interested in investigating the behavior of the embodied evolutionary algorithm with and without exchanging information, i.e. exchanging the agents' genotypes (behavior parameters) with their neighbors, as well the effect of the amount of shared information and the frequency of information sharing. Therefore, we test different versions of the embodied evolutionary algorithm introduced in the previous section, where in some of them it is allowed to exchange information with different proportions and frequencies, while in others it is not.

In all the experiments, we use a 2-dimensional environment represented as an $m \times n$ grid. Each cell i, j on this grid is occupied by an agent $a_{i,j}$. We use the following neighborhood function: $N_{Moore} = \{(i-1, j-1), (i, j-1), (i+1, j-1), (i-1, j), (i+1, j), (i-1, j+1), (i, j+1), (i+1, j+1)\}$. This function, known as the Moore Neighborhood [26], allows each agent to communicate with its nearest horizontal, vertical and diagonal neighbors. The agents that are located at the border of the environment can communicate only with their existing neighbors.

In the following, we first define the two test problems we used in our experimentation (and, for both problems, two alternative scenarios). Then, we describe the five versions of the algorithm we tested on the two problems.

3.1 Imitation problem

We define as *"imitation problem"* a generic problem in which each agent in a network is required to "imitate" a desired pattern (the ground truth) over time. We demonstrate this by optimizing a network of agents to imitate a sequence of 100 images taken from the well-known MNIST handwritten digits dataset [27].

28×28 scenario. In the first scenario, the network consists of 28×28 agents, each one corresponding to one pixel of the MNIST images. The grayscale inten-

sity in each cell is controlled by agent $a_{i,j}$ at time $t \in \{1, 2, \ldots, 100\}$, where t represents the t-th image. Thus, the genotype of $a_{i,j}$ is $\boldsymbol{X}^{i,j} = (x_1^{i,j}, x_2^{i,j}, \ldots, x_{100}^{i,j}) \in [0,1]^{100}$, where each t-th element determines the grayscale intensity (the phenotype) corresponding to the agent's cell at time t. A visualization of some selected images (ground truth) at indices $t = 1$, $t = 50$ and $t = 100$ is shown in Figure 2.

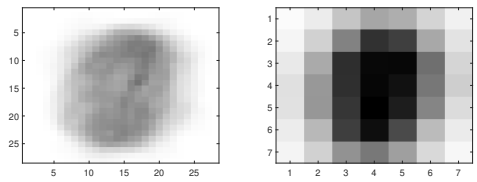
In the initialization phase, the genotype of all agents are generated by randomly sampling their genes in $[0, 1]$ with uniform probability. We use the embodied evolutionary algorithm to find the optimal parameters of each agent that can collectively imitate the sequence of 100 images as close as possible. Thus, in total, there are 784×10^2 parameters to be optimized collectively (28×28 agents $\times 100$ parameters). The fitness value of each agent is calculated as $f_{i,j} = \frac{1}{100} \sum_{t=1}^{100} \mid I_{i,j}(t) - x_t^{i,j} \mid$, where $I_{i,j}(t)$ denotes the desired i, j pixel value of the t-th image selected from the MNIST dataset (pixel values are scaled in the range $[0, 1]$), and $x_t^{i,j}$ represents the corresponding t-th element of the agent's genotype $\boldsymbol{X}^{i,j}$. The collective fitness of the network at generation g, F_g, is then computed as the average fitness across the agents ($F_g = \sum_i \sum_j f_{i,j}$). This value is used for comparing various versions of the embodied evolutionary algorithm with various parameter settings. We run the various versions of the algorithm for 10 independent runs, each one consisting of 20000 generations.

7×7 scenario. One of the key aspects of the approach proposed in this work is to make use of the neighboring agents during the embodied evolutionary process. For instance, an agent can copy the behavior parameters of the best performing agent in its neighborhood (with some mutation). However, not always these parameters can benefit the agent, since the expected behavior of each agent can drastically be different from that of its neighbors. The imitation problem demonstrates this point in that there can be large differences between adjacent cells, e.g. at the edge between the background and the digit. Moreover, these differences change from one digit to another. To further stress these differences, we have defined an additional scenario where we have reduced the number of agents to 7×7, by assigning to each agent the control of a tile made of of 4×4 pixels. As such, in this case each agent controls the intensity level of 16 pixels (rather than just one as in the 28×28 scenario), for a sequence 100 images. Thus, the number of parameters per agent increases from 100 to $4 \times 4 \times 100 = 1600$ (also in this case these parameters are encoded in the genotype of each agent in a real-valued vector). Since the spatial differences of these tiles are larger, the difference between the optimal parameters of two neighboring agents are larger.

We show the average (across 100 MNIST images) differences of the optimal parameters of the agents with their neighbors for the two scenarios (28×28 vs 7×7 agents) in Figure 4. For each cell, we first find the average (across genes) differences with each of its neighbors, and then we take the average (across neighbors) of these differences. The two resulting matrices are finally normalized by scaling each cell value w.r.t. the maximum value among all cells in the two scenarios: in fact, the maximum value in the 7×7 scenario (Figure 4b) is about twice as big as the maximum value in the 28×28 scenario (Figure 4a), thus both matrices are scaled w.r.t. the maximum value in the 7×7 scenario. A higher

grayscale intensity in a cell indicates a higher difference with its neighbors. It can be observed that the differences are much higher in the middle of the images (this is a consequence of the different shapes of the digits, while the cells at the border are more similar since they encode the background), and in general the differences in the 7×7 scenario are much higher (this is a consequence of the lower agent density, which leads to each agent controlling more pixels, such that the desired behavior can be quite different between adjacent tiles).

In this second scenario, we run the various versions of the embodied evolutionary algorithm for 10 independent runs, each one consisting of 1000000 generations: compared to the 28×28 scenario, we extend the number of generations since the number of parameters per agent is 16 times higher.



(a) 28×28 scenario (b) 7×7 scenario

Fig. 4. Imitation problem: Average differences of the optimal parameters of each agent with the optimal parameters of the neighboring agents in 28×28 and 7×7 scenarios. The grayscale intensity indicates the average difference of an agent w.r.t. its neighbors.

3.2 Illumination problem

We further specify the imitation problem to illustrate the simulation of a hypothetical real-world application that we refer to as the *"illumination problem"*. In this case, a network of agents is required to learn to optimally illuminate an environment. We consider a 2-dimensional grid of $25 \times 50 = 1250$ agents, where each agent "illuminates" its own cell, see Figure 3. We assume that different locations in the environment receive different levels of natural light during the day. Each agent $a_{i,j}$ is then required to learn the optimal local illumination, based on the hour of the day, $t \in \{0, 1, \ldots, 23\}$, and its location, i, j. We define the optimal illumination (ground truth) for each cell location as $l_{i,j}(t) = sin\left(\frac{2 \cdot \pi \cdot j}{n} + \frac{2 \cdot \pi \cdot t}{24}\right)$, where n is the maximum value of j. To convert $l_{i,j}(t)$ to pixel values, we scale its values to $[0, 1]$ (i.e. $l_{i,j}(t) = (l_{i,j}(t) + 1)/2$).

Single parameter scenario. In the first scenario, each agent has only one real-valued parameter, $x^{i,j} \in [0, 50]$. The agent's behavior is then obtained as: $a_{i,j}(t) = sin\left(\frac{2 \cdot \pi \cdot x^{i,j}}{50} + \frac{2 \cdot \pi \cdot t}{24}\right)$. At the beginning of the algorithm, $x^{i,j}$ is randomly initialized in $[0, 50]$. The fitness value of each agent $f_{i,j}$ is calculated as the average the difference from the optimal illumination during the day, namely: $f_{i,j} = \frac{1}{24} \sum_{t=0}^{23} \mid l_{i,j}(t) - a_{i,j}(t) \mid$. We use the embodied evolutionary approach to find the optimal parameters of each agent that can collectively imitate the desired illumination pattern (in total, 25×50 parameters for the whole network). Also in this case we use the collective fitness F_g (defined as for the imitation problem) to perform comparisons among the various versions of the embodied evolutionary algorithm. In this case, we perform 10 independent runs for each version of the embodied evolutionary algorithm for 5000 generations.

Vector representation scenario. In the second scenario, we encode the agents' behavior using a 24-dimensional real-valued vector (similarly to the imitation problem), $\boldsymbol{X}^{i,j} = (x_1^{i,j}, x_2^{i,j}, \ldots, x_{24}^{i,j}) \in [0,1]^{24}$, to control the agent-local level of illumination at each time step $t = \{0, 1, \ldots, 23\}$. Here, the values in $[0,1]$ encode the amount of light, from no light (0) to the maximum amount of light (1). The main difference w.r.t. the single parameter scenario is then that in this case we do not enforce a sinusoidal pattern in each agent, but this should be discovered automatically by the embodied evolutionary process. At the beginning of the algorithm, the vector $\boldsymbol{X}^{i,j}$ of each agent is randomly initialized in $\in [0,1]^{24}$. Similarly to the first scenario, the fitness value of each agent $f_{i,j}$ is the average difference from the optimal illumination during the day, namely: $f_{i,j} = \frac{1}{24}\sum_{t=0}^{23} \mid l_{i,j}(t) - a_{i,j}(t) \mid$. In this scenario, there are in total 30×10^4 parameters (25×50 agents $\times 24$ parameters). For this problem, we perform 10 independent runs for each version of the algorithm for 20000 generations (the number of generations is increased due to the larger number of parameters).

3.3 Versions of the embodied evolutionary algorithm

We tested in total five versions of the embodied evolutionary algorithm presented in the previous section, configured as follows:

- **XoverRand:** this version of the algorithm is given in Algorithm 1.
- **XoverBest:** this version is similar to the previous one, the only difference being that the function $rand(\boldsymbol{X}^{i,j})$ in Line 6 of Algorithm 1 is replaced with $best(\boldsymbol{X}^{i,j})$. In this case, the algorithm performs the crossover with the best agent in its neighborhood.
- **HillClimbing:** in this version, we do not use any neighborhood function, therefore the agents do not share any information with their neighbors. We modify Algorithm 1 by deleting Line 6 (where crossover operator is used), and changing Line 7 with: $\boldsymbol{X}' \leftarrow \boldsymbol{X}^{i,j} + \mathcal{N}(0, \sigma)$.
- **CopyBest:** in this version, each agent copies the genotype of the best agent in its neighborhood and then applies the mutation operator. Similar to HillClimbing, crossover operator is not used. Thus, we modify Algorithm 1 by deleting Line 6 and changing Line 7 with: $\boldsymbol{X}' \leftarrow best(\boldsymbol{X}^{i,j}) + \mathcal{N}(0, \sigma)$.
- **CopyRand:** in this version, each agent copies the genotype of a randomly selected agent in its neighborhood and then applies the mutation operator. Also in this case, crossover operator is not used. Thus, we modify Algorithm 1 by deleting Line 6 and changing Line 7 with: $\boldsymbol{X}' \leftarrow rand(\boldsymbol{X}^{i,j}) + \mathcal{N}(0, \sigma)$.

4 Experimental Results

On the two problems and four scenarios, we tested different mutation rates, namely 1%, 0.1% and 0.01% of the search domains. Thus, in the case of the imitation and illumination problems with vector representation (both with domain $[0,1]$), we use $mr \in \{0.0001, 0.001, 0.01\}$, while for the illumination problem with single parameter (with domain $[0,50]$) we use $mr \in \{0.005, 0.05, 0.5\}$. Due to space limitations, we report some of the experimental results, as well as the statistical analysis, in the Appendix.

4.1 Imitation problem

28 × 28 scenario. Figure 5 shows the average (across 10 runs per algorithm) collective fitness (F_g) trends of the various versions of the embodied evolutionary algorithm obtained with $mr \in \{0.0001, 0.001, 0.01\}$, $cp \in \{0.2, 0.5, 1.0\}$ and $cr = 0.5$. Note that in the following, in all fitness trend figures we show on the x-axis the number of generations, while on the y-axis the fitness F_g.

We observe that performing crossover with a random neighbor performs better than performing crossover with the best neighbor, which in turn performs better than not doing any crossover. This indicates that exchanging partial components of the genotype helps the optimization process, even though the optimal parameters of the neighbors are different. The fact that crossing over with a random neighbor is more efficient than doing that with the best neighbor is likely due to a higher chance of selecting a neighbor whose optimal behavior is closer to that of the focal agent, which might not be the case of the best neighbor (e.g. if the focal agent is on the digit while its best neighbor is on the background).

We further observe that the versions that copy the neighbors perform better than HillClimbing. Also, when crossover is not used, lower mutation rates appear to slow the convergence: being the algorithm population-less, the only way to converge faster is to perform larger mutations. However, we observe the opposite effect when crossover is used: since mutation is performed *after* crossover, in this case too high mutation rates might obliterate any advantage obtained from the neighbor's exchanged genes. Overall, XoverRandCP1CR05 (this notation, used also in the following, indicates $cp = 1.0, cr = 0.5$) appears to perform the best for all mutation rates. However, as illustrated in Figure 5d, different mutation rates perform differently ($mr = 0.01$ performs the worst).

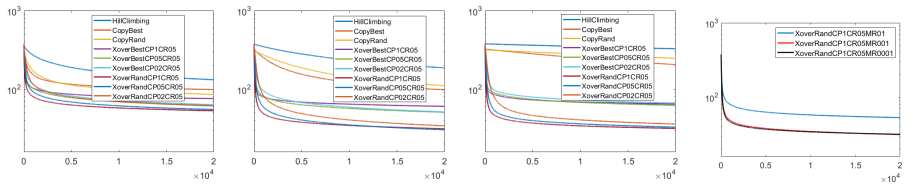
Figure 6 shows a visualization of the results obtained by HillClimbing, CopyBest and XoverRandCP1CR05 (with $mr = 0.001$) during the evolutionary process[1]. We observe that different versions of the algorithm show different behaviors. For instance, with HillClimbing the reconstructed image appears very noisy: whereas some agents appear to imitate well the ground truth, there are many isolated agents that are not well adapted. This is expected since each agent is trying to optimize its own local fitness function without any genotypical exchange. On the other hand, in the case of CopyBest and XoverRand we observe that the images are much smoother and less noisy. Also, CopyBest seems unable to optimize the agents in the middle of the image: this is due to the fact that those agents have much larger differences (in terms of their optimal parameters) w.r.t. their neighbors. Therefore, copying the neighbors provides little or no benefit.

We have also investigated the effect of cp and cr, which affect how frequently crossover occurs, and how much information is exchanged (see Appendix A). Again, we observe that in general XoverRand performs better than XoverBest. Also, the results indicate a better performance when crossover is performed frequently (higher cp) but exchanging a small number of components (lower cr). In particular, the best result is achieved by XoverRandCP1CR005.

[1] Video of the evolutionary process available at: `https://youtu.be/DfjSvKA6KNI`.

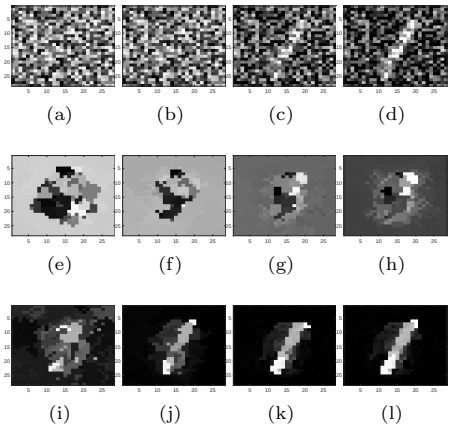
7 × 7 scenario. Similarly to the 28 × 28 scenario, the versions of the algorithm that do not employ crossover perform worse than those that use it (see Appendix A). This shows that even if the optimal parameters of the neighbor agents are quite different (as we have shown in Figure 4), sharing behavior parameters with neighbors helps the optimization process. Also in this scenario, better performance are obtained with high values of cp and lower values of cr.

In Figure 7, we show a visualization of the results obtained by various versions of the algorithm (with $mr = 0.001$) at the last generation $g = 1000000$: also here we observe that HillClimbing produces a much noisier image, while XoverRandCP1CR001 performs the best. In general it seems that in most versions based on crossover there are some isolated agents that are non-optimal (placed in an almost-regular spatial pattern).



(a) $mr = 0.01$ (b) $mr = 0.001$ (c) $mr = 0.0001$ (d) XoverRandCP1CR05

Fig. 5. Imitation problem (28 × 28 scenario): Results of different versions of the algorithm with different values of mr and cp, and $cr = 0.5$.

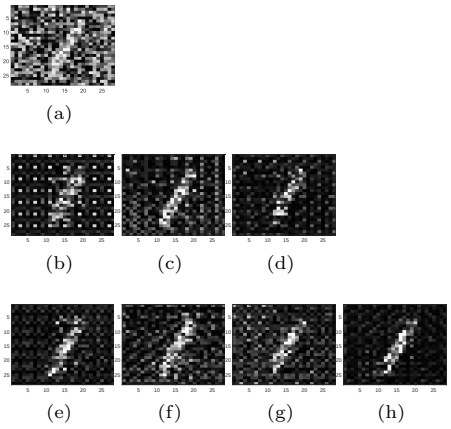


(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Fig. 6. Imitation problem (28 × 28 scenario): Visualization of the evolutionary process at generations $g \in \{100, 1000, 10000, 20000\}$ for each row from left to right (ground truth shown in Figure 2a). The rows present the results of HillClimbing, CopyBest and XoverRandCP1CR05 ($mr = 0.001$) respectively.



(a)

(b) (c) (d)

(e) (f) (g) (h)

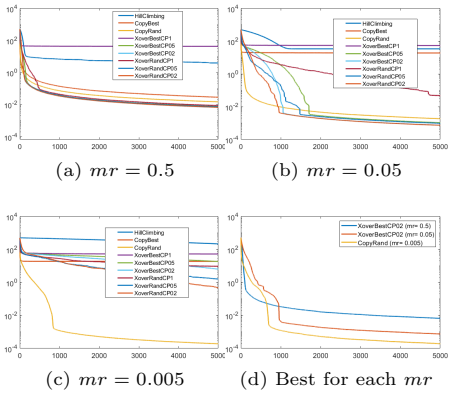
Fig. 7. Imitation problem (7×7 scenario): Visualization of the evolutionary process at the last generation (ground truth shown in Figure 2a) wtih $mr = 0.001$ and different values of cp and cr: HillClimbing (a); XoverBest with CP02CR01 (b), CP02CR001 (c), CP05CR001 (d); XoverRand with CP02CR01 (e), CP02CR001 (f), CP05CR001 (g), CP1CR001 (h).

4.2 Illumination problem

Single parameter scenario. As seen earlier, in this scenario each agent aims to optimize a parameter corresponding to the phase shift in the sine function used for the ground truth. Figure 8 provides the average (across 10 runs per algorithm) F_g trends of the various versions of the embodied evolutionary algorithm obtained with $mr \in \{0.005, 0.05, 0.5\}$ and $cp \in \{0.2, 0.5, 1.0\}$ (in this case we use arithmetic crossover, so cr is not needed). The main observation is that XoverBestCP02 performs best with $mr = 0.05$ and $mr = 0.5$, while for the lowest value ($mr = 0.005$) CopyRand obtains the best results. This might be due to the fact that when the mutation rate is very small its effect cancels out the effect of the arithmetic crossover (which is essentially a kind of mutation), thus producing a slower convergence. On the contrary, with larger mutation rates the two effects are combined and speed up convergence. So, with low mr values, it is just more efficient to copy a random neighbor and apply a small mutation.

In Appendix B, we show a visualization of the results obtained by three selected versions of the algorithm, and we observe again that with no information sharing the results are very noisy, while with crossover results are clearly better.

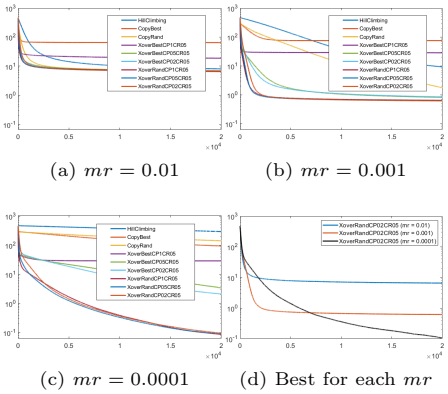
Vector representation scenario. Figure 9 shows the average (across 10 runs per algorithm) F_g trends of the various versions of the algorithm with $mr \in \{0.0001, 0.001, 0.01\}$, and (in the case of XoverRand and XoverBest) $cp \in \{0.2, 0.5, 1.0\}$ and $cr = 0.5$. Similarly to the previous results, we observe that the versions that use crossover operator tend to perform better than those that do not use crossover. Moreover, XoverRandCP02CR05 performs the best for different mutation rates, as it does in the imitation problem. Also, we observe that smaller mutation rates lead to slower convergence in all cases. In Appendix B, we report a visualization of the results obtained by three selected versions of the algorithm, with observations similar to the previous cases.



(a) $mr = 0.5$ (b) $mr = 0.05$

(c) $mr = 0.005$ (d) Best for each mr

Fig. 8. Illumination problem (single parameter scenario): Results of different versions of the algorithm with different values of mr and cp.



(a) $mr = 0.01$ (b) $mr = 0.001$

(c) $mr = 0.0001$ (d) Best for each mr

Fig. 9. Illumination problem (vector representation scenario): Results of different versions of the algorithm with different values of mr, cp and $cr = 0.5$.

5 Conclusions

In many network applications, the optimal behavior of the agents is not known before deployment, and the agents often need to adapt to environment changes. Performing the optimization process offline, by collecting each agent to change its behavior parameters whenever it is needed, would be costly and ineffective. In this work, we proposed a distributed embodied evolutionary approach to optimize the behavior of networks of agents at runtime.

We assumed a collection of spatially distributed agents that can communicate locally. We used the local communication capability to exchange the agents' behavior parameters, so to copy the neighbors' parameters and/or use them for local perturbations and crossover. Intuitively, the agents with close proximity may be required to perform similarly (thus may have similar optimal parameters) since they are likely to share similar environmental conditions. In this case, it would make sense to share their parameters to learn from one another. Nevertheless, even when the optimal parameters are drastically different within a neighborhood, parameter exchange may still be helpful in the optimization process. To test these two cases, we devised a number of test cases with different levels of differences between the optimal parameters of any given agent and those of its neighbors. We further compared these results with the case where each agent optimizes its behavior without any parameter exchange.

We found that information exchange (through crossover) works the best in all cases -except with arithmetic crossover and small mutation rates- including when the differences of the optimal parameters of the agents with their neighbors are large. We also observed a certain variation of the performance depending on the parameter settings of the algorithm (i.e., mr, cp and cr), therefore in the future it might be interesting to evaluate (self)adaptive evolutionary approaches. Overall, we noted that performing frequent crossover (high cp) by exchanging a small fraction of the genotype (low cr) provided the best results. However, applying crossover less frequently can still lead to an improvement. Among the various versions of the algorithm we tested, XoverRand demonstrated the best performance. Since this version does not require the fitness of the neighbors (due to random selection), with small values of cr it might possible to broadcast only some randomly selected components of the agents' genotypes.

This work was mainly aimed at demonstrating the effectiveness of the proposed approach. Therefore, we considered two "ideal" test cases, with a fixed grid topology and perfect communication. In future works, we plan to apply the proposed algorithm to more realistic scenarios, with different network topologies, dynamic connections and possibly noisy communication. Another assumption of this work was that the local fitness functions are in general different (in terms of optimum) across agents, but still they are comparable. However, this might not be the case of all applications: therefore, we would like to see if information exchange is beneficial also in these cases, and how it could be applied (for instance, function similarity could be used to define the neighborhood function, to create niches/communities). Finally, it will be interesting to perform real-world validation with physical networks at various scales.

References

1. Bouabene, G., Jelger, C., Tschudin, C., Schmid, S., Keller, A., May, M.: The autonomic network architecture (ANA). Journal on Selected Areas in Communications **28**(1) (2009) 4–14
2. Xiao, Y.: Bio-inspired computing and networking. CRC Press (2016)
3. Nakano, T.: Biologically inspired network systems: A review and future prospects. Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **41**(5) (2010) 630–643
4. Dressler, F., Akan, O.B.: A survey on bio-inspired networking. Computer Networks **54**(6) (2010) 881–900
5. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. Transactions on Evolutionary Computation **6**(5) (2002) 443–462
6. Arenas, M.G., Collet, P., Eiben, A.E., Jelasity, M., Merelo, J.J., Paechter, B., Preuß, M., Schoenauer, M.: A framework for distributed evolutionary algorithms. In: International Conference on Parallel Problem Solving from Nature, Springer (2002) 665–675
7. Biazzini, M., Montresor, A.: P2POEM: Function optimization in P2P networks. Peer-to-Peer Networking and Applications **6**(2) (2013) 213–232
8. Sipper, M.: Studying artificial life using a simple, general cellular model. Artificial Life **2**(1) (1994) 1–35
9. Sipper, M.: Evolution of parallel cellular machines. Volume 4. Springer (1997)
10. Alba, E., Troya, J.M.: Cellular evolutionary algorithms: Evaluating the influence of ratio. In: International Conference on Parallel Problem Solving from Nature, Springer (2000) 29–38
11. Iacca, G.: Introducing DOWSN: distributed optimization in wireless sensor networks. In: UK Workshop on Computational Intelligence, IEEE (2012) 1–8
12. Iacca, G.: Distributed optimization in wireless sensor networks: an island-model framework. Soft Computing **17**(12) (2013) 2257–2277
13. Johnson, D.M., Teredesai, A.M., Saltarelli, R.T.: Genetic programming in wireless sensor networks. In: European Conference on Genetic Programming, Springer (2005) 96–107
14. Valencia, P., Lindsay, P., Jurdak, R.: Distributed genetic evolution in WSN. In: International Conference on Information Processing in Sensor Networks, ACM/IEEE (2010) 13–23
15. Watson, R.A., Ficici, S.G., Pollack, J.B.: Embodied evolution: Distributing an evolutionary algorithm in a population of robots. Robotics and autonomous systems **39**(1) (2002) 1–18
16. Eiben, A., Haasdijk, E., Bredeche, N.: Embodied, on-line, on-board evolution for autonomous robotics. In: Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution, Springer (2010) 361–382
17. Bredeche, N., Haasdijk, E., Prieto, A.: Embodied evolution in collective robotics: A review. Frontiers in Robotics and AI **5** (2018) 12
18. Bredeche, N., Montanier, J.M., Liu, W., Winfield, A.F.: Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. Mathematical and Computer Modelling of Dynamical Systems **18**(1) (2012) 101–129
19. Haasdijk, E., Bredeche, N., Eiben, A.: Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics. PloS one **9**(6) (2014)

20. Montanier, J.M., Carrignon, S., Bredeche, N.: Behavioral specialization in embodied evolutionary robotics: why so difficult? Frontiers in Robotics and AI **3** (2016) 38
21. Hart, E., Steyven, A., Paechter, B.: Improving survivability in environment-driven distributed evolutionary algorithms through explicit relative fitness and fitness proportionate communication. In: Genetic and Evolutionary Computation Conference. (2015) 169–176
22. Zahadat, P., Hamann, H., Schmickl, T.: Evolving diverse collective behaviors independent of swarm density. In: Genetic and Evolutionary Computation Conference Companion. (2015) 1245–1246
23. Pérez, I.F., Sanchez, S.: Influence of local selection and robot swarm density on the distributed evolution of GRNs. In: International Conference on the Applications of Evolutionary Computation (Part of EvoStar), Springer (2019) 567–582
24. Trueba, P., Prieto, A., Bellas, F., Duro, R.J.: Embodied evolution for collective indoor surveillance and location. In: International Work-Conference on the Interplay Between Natural and Artificial Computation, Springer (2015) 138–147
25. Prieto, A., Bellas, F., Trueba, P., Duro, R.J.: Real-time optimization of dynamic problems through distributed embodied evolution. Integrated Computer-Aided Engineering **23**(3) (2016) 237–253
26. Gray, L.: A mathematician looks at Wolfram's new kind of science. Notices-American Mathematical Society **50**(2) (2003) 200–211
27. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324

A Additional results on the imitation problem

28 × 28 scenario. We have investigated the effect of cp and cr, which affect how frequently crossover occurs, and how much information is exchanged respectively. In Figure 10, we provide the average (across 10 runs per algorithm) F_g trends of XoverBest and XoverRand for all combinations of $cp \in \{0.2, 0.5, 1.0\}$ and $cr \in \{0.05, 0.2, 0.5\}$. In these experiments, we set $mr = 0.001$. We observe that in general XoverRand performs better than XoverBest. Also, the results indicate a better performance when crossover is performed frequently (higher cp) but exchanging small number of components (lower cr). In particular, we observe that the best result is achieved using XoverRandCP1CR005, which performs crossover with a randomly selected neighbor exchanging a very small number of components (the expected number of exchanged components per crossover is $0.005 \times 100 = 5$).

7 × 7 scenario. Figure 11 shows the average (across 10 runs per algorithm) F_g trends of a subset of the various versions of the algorithm, with $mr = 0.001$ and selected combinations of $cp \in \{0.2, 0.5, 1.0\}$ and $cr \in \{0.01, 0.1, 0.5\}$. Similarly to the 28×28 scenario, HillClimbing and, in general, the versions of the algorithm that do not employ crossover (not shown in the figure), perform worse than those that use it. This shows that even if the optimal parameters of the neighbor agents are quite different (as we have shown in Figure 4), sharing behavior parameters with neighbors helps the optimization process. Also in this scenario, better performance are obtained with high values of cp and lower values of cr (XoverRandCP1CR001).

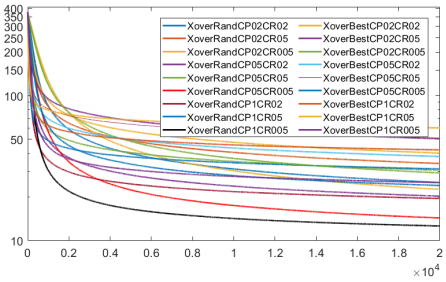


Fig. 10. Imitation problem (28×28 scenario): Results of different versions of the algorithm with $mr = 0.001$ and different values of cp and cr.

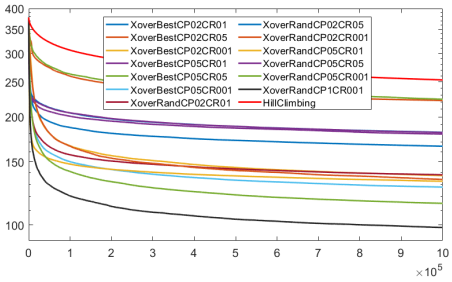


Fig. 11. Imitation problem (7×7 scenario): Results of different versions of the algorithm with $mr = 0.001$ and different values of cp and cr.

B Illumination problem

Single parameter scenario. Figure 12 shows a visualization of the results obtained by HillClimbing, CopyBest and XoverRandCP02 ($mr = 0.05$) during the evolutionary process. We observe that with no information sharing (HillClimbing), the results are very noisy, with several isolated agents that are non-optimal (Figures 12a-12d). With crossover, results are clearly better, see Figures 12i-12l. However, we observe in the results of XoverRand that at $g = 100$ there are many non-optimal agents at the right and left edges of the image, see Figure 12i. This is due to the problem formulation and the arithmetic crossover operator: in fact, there are two optimal values for the left and right edge of the image (0 and 50), due to the periodicity of the sine function. Thus, agents are optimized towards one of these values, which may be different from that of their neighbors. Moreover, arithmetic crossover in this case does not help because the fitness of the average of these two values is worse.

Vector representation scenario. Figure 13 shows a visualization of the results obtained by HillClimbing, CopyBest and XoverRandCP02CR05 ($mr = 0.001$) during the evolutionary process[2]. The results of HillClimbing appear more noisy, with many isolated, not well adapted agents. Also, we observe vertical "stripes" in the results of CopyBest: this is due to the fact that the optimal parameters of the agents in each column are the same, therefore they tend to share the same parameters. Overall, we observe that crossover helps the optimization process even though the optimal parameters of neighboring agents are different.

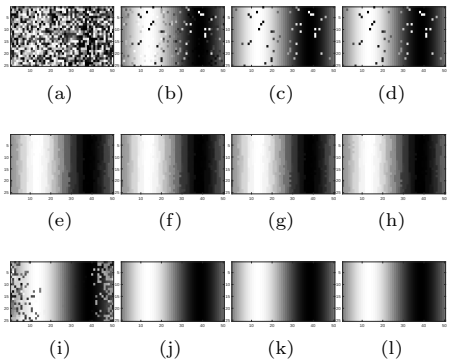


Fig. 12. Illumination problem (single parameter scenario): Visualization of the evolutionary process at generations $g \in \{100, 1000, 3000, 5000\}$ in each row from left to right (ground truth shown in Figure 3a). The rows present the results of HillClimbing, CopyBest and XoverRandCP02 ($mr = 0.05$) respectively.

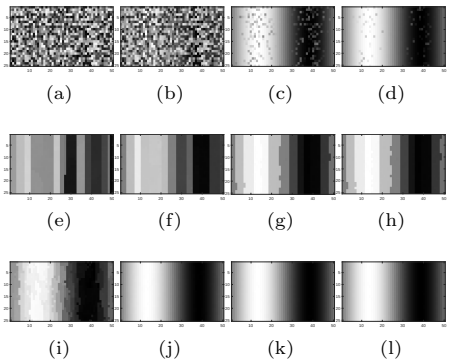


Fig. 13. Illumination problem (vector representation): Visualization of the evolutionary process at generations $g \in \{100, 1000, 10000, 20000\}$ for each row from left to right (ground truth shown in Figure 3a). The rows present the results of HillClimbing, CopyBest and XoverRandCP02CR05 ($mr = 0.001$) resp.

[2] Video of the evolutionary process available at: `https://youtu.be/5HZ-SMLyv_E`.

C Statistical analysis

In the following, we report the results of the pairwise comparisons (based on Wilcoxon Rank-Sum test, $\alpha = 0.05$), visually represented in a symmetric matrix format, and a post-hoc multiple-comparisons analysis (based on Nemenyi test, $\alpha = 0.05$), visually represented with Critical Difference plots. We performed both tests on each group of algorithm versions compared in Figure 5, 8, and 9.

In Table 1, 2 and 3 it can be noted that, apart from a few occasional cases, the null-hypothesis H_0 (statistical equivalence) is rejected in most cases, highlighting the fact that the different parametrization and algorithm versions are indeed different (pairwise) in statistically significant terms.

In Figure 14, 15, and 16, algorithms that are considered statistically equivalent w.r.t. the Critical Difference (CD) calculated by the Nemenyi test (depicted as a segment on top of the plot) are graphically connected by a thick black line. Also, better algorithms get a lower rank. With this notation in mind, the main results of the Nemenyi test can be summarized as follows:

- In Figure 14 (imitation problem, 28×28 scenario), XoverRand versions consistently rank before XoverBest, CopyRand/CopyBest and HillClimbing versions, in this order. Furthermore, all XoverRand versions and two XoverBest versions result statistically equivalent w.r.t. the CD. Also, the versions without crossover (CopyRand/CopyBest and HillClimbing) perform better (i.e. they get lower ranks) with higher mr values, while the contrary happens on the versions with crossover.
- In Figure 15 (illumination problem, single parameter scenario), CopyRand with $mr = 0.005$ ranks first, but it results statistically equivalent (w.r.t. the CD) to CopyRand with $mr = 0.05$ and $mr = 0.5$, as well as XoverBest and XoverRand versions with mr larger than 0.005. Also in this case, HillClimbing (with the lowest mr value) obtains the worst rank.
- In Figure 16 (illumination problem, vector representation scenario), most of the XoverRand versions rank before XoverBest and the versions without crossover (although the general trend is somehow less clear w.r.t. the two previous cases).

Overall, the above observations obtained from the multiple-comparisons analysis are in line what we discussed in Section 4 of the main text.

Table 1. Wilcoxon Rank-sum test ($\alpha = 0.05$) on the pairwise comparisons between the algorithm versions compared in Figure 5. "=" indicates that the null-hypothesis H_0 (statistical equivalence) is accepted (omitted on the diagonal cells). Empty cells indicate that the null-hypothesis is rejected.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
HillClimbingMR01	**1**																											
CopyBestMR01	**2**											=																
CopyRandMR01	**3**																											
XoverBestCP1CR05MR01	**4**																											
XoverBestCP05CR05MR01	**5**																							=				
XoverBestCP02CR05MR01	**6**																							=				
XoverRandCP1CR05MR01	**7**																											
XoverRandCP05CR05MR01	**8**																											
XoverRandCP02CR05MR01	**9**																							=				
HillClimbingMR001	**10**																											
CopyBestMR001	**11**		=																									
CopyRandMR001	**12**																											
XoverBestCP1CR05MR001	**13**																											
XoverBestCP05CR05MR001	**14**															=												
XoverBestCP02CR05MR001	**15**														=													
XoverRandCP1CR05MR001	**16**																									=		
XoverRandCP05CR05MR001	**17**																											
XoverRandCP02CR05MR001	**18**																											
HillClimbingMR0001	**19**																											
CopyBestMR0001	**20**																											
CopyRandMR0001	**21**																											
XoverBestCP1CR05MR0001	**22**																								=			
XoverBestCP05CR05MR0001	**23**					=	=			=																		
XoverBestCP02CR05MR0001	**24**																						=					
XoverRandCP1CR05MR0001	**25**																=											
XoverRandCP05CR05MR0001	**26**																											
XoverRandCP02CR05MR0001	**27**																											

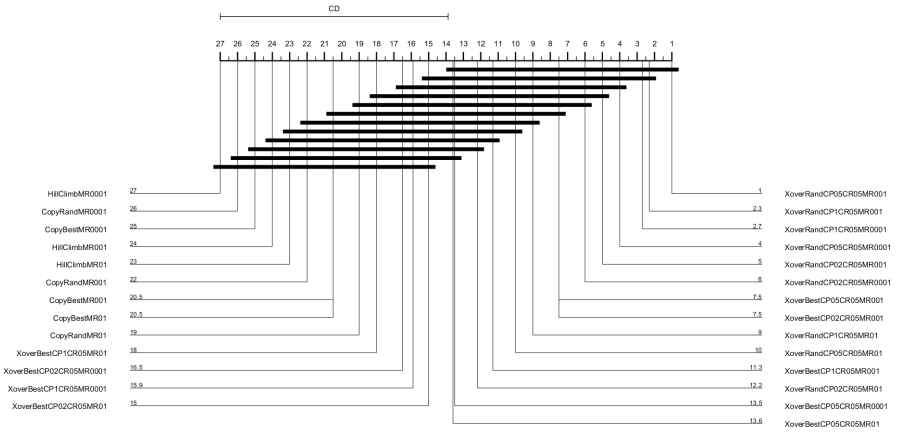


Fig. 14. Critical Difference plot based on the Nemenyi post-hoc test ($\alpha = 0.05$) on the algorithm versions compared in Figure 5.

Table 2. Wilcoxon Rank-sum test ($\alpha = 0.05$) on the pairwise comparisons between the algorithm versions compared in Figure 8. "=" indicates that the null-hypothesis H_0 (statistical equivalence) is accepted (omitted on the diagonal cells). Empty cells indicate that the null-hypothesis is rejected.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
HillClimbingMR05	**1**																											
CopyBestMR05	**2**																											
CopyRandMR05	**3**																											
XoverBestCP1MR05	**4**																											
XoverBestCP05MR05	**5**																											
XoverBestCP02MR05	**6**									=																		
XoverRandCP1MR05	**7**																											
XoverRandCP05MR05	**8**																											
XoverRandCP02MR05	**9**						=																					
HillClimbingMR005	**10**																											
CopyBestMR005	**11**																				=			=				
CopyRandMR005	**12**																											
XoverBestCP1MR005	**13**																						=					
XoverBestCP05MR005	**14**																											
XoverBestCP02MR005	**15**																		=									
XoverRandCP1MR005	**16**																											
XoverRandCP05MR005	**17**																											
XoverRandCP02MR005	**18**															=												
HillClimbingMR0005	**19**																											
CopyBestMR0005	**20**											=												=				
CopyRandMR0005	**21**																											
XoverBestCP1MR0005	**22**													=														
XoverBestCP05MR0005	**23**											=									=							
XoverBestCP02MR0005	**24**																											
XoverRandCP1MR0005	**25**																											
XoverRandCP05MR0005	**26**																											
XoverRandCP02MR0005	**27**																											

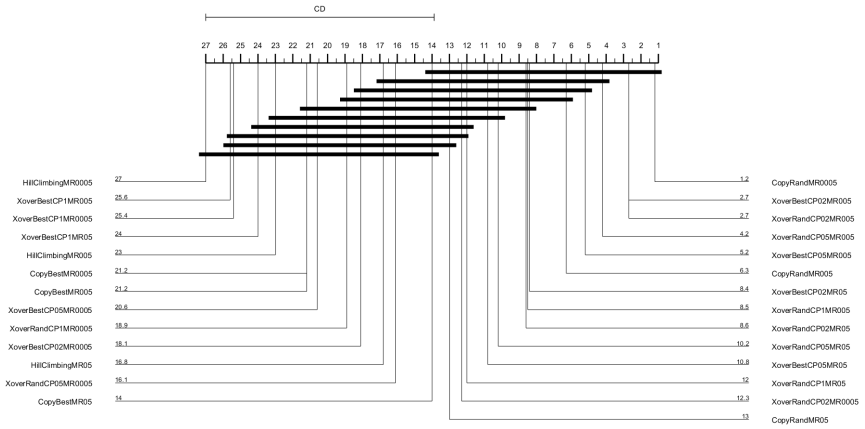


Fig. 15. Critical Difference plot based on the Nemenyi post-hoc test ($\alpha = 0.05$) on the algorithm versions compared in Figure 8.

Table 3. Wilcoxon Rank-sum test ($\alpha = 0.05$) on the pairwise comparisons between the algorithm versions compared in Figure 9. "=" indicates that the null-hypothesis H_0 (statistical equivalence) is accepted (omitted on the diagonal cells). Empty cells indicate that the null-hypothesis is rejected.

		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
HillClimbingMR01	**1**																											
CopyBestMR01	**2**																											
CopyRandMR01	**3**																											
XoverBestCP1MR01	**4**																											
XoverBestCP05MR01	**5**																											
XoverBestCP02MR01	**6**																											
XoverRandCP1MR01	**7**																											
XoverRandCP05MR01	**8**																											
XoverRandCP02MR01	**9**																											
HillClimbingMR001	**10**																											
CopyBestMR001	**11**																											
CopyRandMR001	**12**																								=			
XoverBestCP1MR001	**13**																						=					
XoverBestCP05MR001	**14**															=												
XoverBestCP02MR001	**15**														=													
XoverRandCP1MR001	**16**																											
XoverRandCP05MR001	**17**																											
XoverRandCP02MR001	**18**																											
HillClimbingMR0001	**19**																											
CopyBestMR0001	**20**																											
CopyRandMR0001	**21**																											
XoverBestCP1MR0001	**22**													=														
XoverBestCP05MR0001	**23**																											
XoverBestCP02MR0001	**24**												=															
XoverRandCP1MR0001	**25**																										=	=
XoverRandCP05MR0001	**26**																									=		=
XoverRandCP02MR0001	**27**																									=	=	

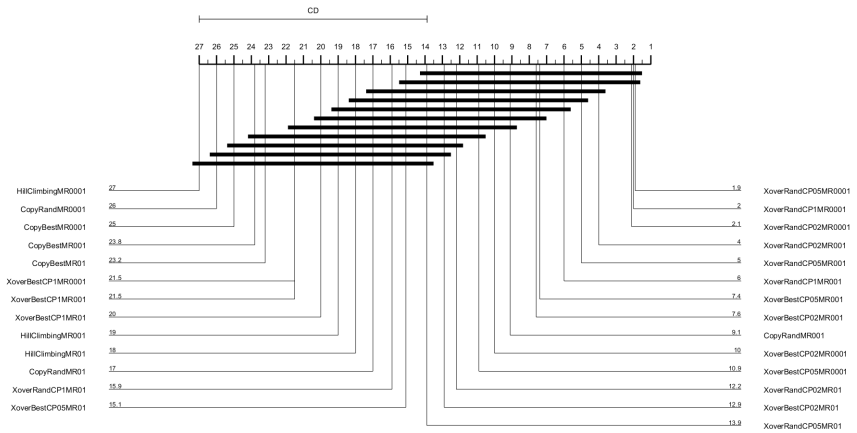


Fig. 16. Critical Difference plot based on the Nemenyi post-hoc test ($\alpha = 0.05$) on the algorithm versions compared in Figure 9.