The Role of the Substrate in CA-based Evolutionary Algorithms

Gloria Pietropolli
DIA - University of Trieste
Trieste, Italy
gloria.pietropolli@phd.units.it

Stefano Nichele
Østfold University College
Halden, Norway
Oslo Metropolitan University
Oslo, Norway
stefano.nichele@hiof.no

Eric Medvet
DIA - University of Trieste
Trieste, Italy
emedvet@units.it

ABSTRACT

Cellular automata (CA) are a convenient way to describe the distributed evolution of a dynamical system over discrete time and space. They can be used to express evolutionary algorithms (EAs), where the time is the flow of iterations and the space is where the population is hosted. When the CA evolves over a finite grid of cells, the *substrate*, each cell hosts an individual and the CA rule applies variation operators using the local and neighbor individuals. In this paper, we explore the possibility of enforcing a structure on the substrate. Instead of a flat toroidal grid, we use substrates where some empty cells never host individuals. These cells may act as barriers, slowing down the propagation of genetic traits and hence potentially improving the population diversity, eventually mitigating the risk of premature convergence. We experimentally evaluate the impact of these substrates using a simple CA-based EA on multi-modal and multi-objective problems. We find evidence of a positive impact in some circumstances; on multi-modal problems, convergence is slightly faster and the EA more often reaches all the targets.

CCS CONCEPTS

• **Theory of computation → Evolutionary algorithms**; Distributed computing models; • **Computing methodologies → Genetic algorithms**; *Genetic programming*.

KEYWORDS

Cellulara automata, quality-diversity, multi-modal optimization, multi-objective optimization, symbolic regression

ACM Reference Format:

Gloria Pietropolli, Stefano Nichele, and Eric Medvet. 2024. The Role of the Substrate in CA-based Evolutionary Algorithms. In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3638529.3654112

1 INTRODUCTION AND RELATED WORKS

The automatic exploration of search spaces to find complex solutions to different problems is one of the most active areas of research in machine learning and optimization [19, 45]. Among the different methods, evolutionary algorithms (EAs) [13] are particularly powerful population-based optimization algorithms that rely on a guided trial and error process to search for candidate solutions of increasing quality. Typically, in EAs all the candidate solutions compete with each other for survival. Therefore, EAs may suffer from convergence to sub-optimal solutions and lack of diversity in the population [19, 49].

Attempts to mitigate such effects have gained popularity [14, 15, 18, 26, 35, 43, 50]. In particular, quality-diversity (QD) algorithms [42] utilize a discretized parametrization of the search space to ensure localized competition and, in turn, promote diversity. Indeed, QD methods have been shown to be able of producing high-quality solutions and a better coverage of the search spaces. However, this comes at a cost, *i.e.*, one needs to define meaningful features (descriptors) that divide the search space.

An alternative to promote localized competition is to use distributed QD algorithms [6], where the level of distributedness may vary from distributed sub-populations interacting locally, *i.e.*, population islands in a graph connected with a certain topology, to a more regular lattice-distributed population, *i.e.*, cellular evolutionary algorithms (cEAs) [3]. In this last case, the population is in fact embedded within a cellular automaton (CA) substrate, and genetic operators act on local neighborhoods. This has the advantage of slowing down the diffusion of genetic traits and therefore favoring exploration, while exploitation is achieved by competition in local neighborhoods [27].

CA, initially proposed by Von Neumann and Burks [52], represent an abstract model of self-replication in biological systems [46]. Their effectiveness in distributed computing contributes to both efficiency and robustness when addressing complex computational tasks [33]. As such, they have been shown to be well suited as substrate for evolution.

cEAs have been also explored in the context of multi-objective problems [39] and with different tradeoffs of exploration/exploitation guided by different neighborhood radius and grid shapes [1]—for a comprehensive review of cEAs, refer to [2]. The concept of local evolution in cellular spaces has been additionally explored in the context of open-endedness in [21, 29, 30]. Recently, a CA-inspired modification of geometric semantic GP (GSGP) has been proposed in [5] to improve diversity.

However, to the best of our knowledge, an open problem is how to favor diversity further in cEAs without any need of additional descriptors to compartmentalize the search space, additional numerical fitness measures, or additional hyperparameters to be tuned (such as radius or grid shape). In this work, we introduce a cEA that incorporates empty cells into the cellular substrate, effectively

GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0494-9/24/07…$15.00
https://doi.org/10.1145/3638529.3654112

introducing barriers that hinder the rapid spread of genetic traits, thereby fostering diversity within the population. We rigorously test the proposed method on multi-modal and multi-objective optimization problems and we show that a substrate with barriers is indeed beneficial to the evolutionary search, in particular on multi-modal problems. By compartmentalizing the population into distinct regions separated by empty cells, our cEA is able to maintain a useful diversity which results in a slightly faster convergence and a higher likelihood of reaching multiple target solutions with respect to a flat substrate. Notably, this enhanced performance is achieved without imposing any additional constraints on the problem or demanding additional user input (as would be required for defining suitable solution descriptors). Consequently, our cEA retains the broad applicability of a standard genetic algorithm (GA) while simultaneously harnessing the benefits of QD search.

2 CELLULAR AUTOMATA

In brief and intuitive terms, a CA consists of a discrete space whose content evolves over discrete time. The content at each position of the space at a given time step is determined according to a transition rule that operates on the content of the neighboring positions at the previous time step.

The nature of the content, the structure of the space, and the rule depend on the kind of CA. They go from simple entities (nevertheless resulting in interesting and widely studied dynamics [47] and computation capabilities [10]) consisting of 1-D or 2-D grids of bits evolved with case-based rules, as in the classic example of Conway's game of life [9], to more complex cases where the content at each position is a possibly large numerical vector and the rule is a neural network, resulting in a CA able to self classify its overall state [38, 44], grow arbitrary shapes [34, 40], or control a robotic agent [37, 41, 51].

For the purpose of this study, we define a CA as a tuple (w, h, S, n, r, r_0) where $w \in \mathbb{N}$ and $h \in \mathbb{N}$ are the width and height of a bi-dimensional grid, S is the *set of states* each grid cell can take, $n : X_{w,h} \to X^*_{w,h}$ is the *neighborhood function*, with $X_{w,h} = \{1, \dots, w\} \times \{1, \dots, h\}$ being the set of coordinates of the grid, $r : S^* \to S$ is the *transition rule*, and $r_0 : X_{w,h} \to S$ is the *initialization* rule. The overall state of a CA at a given time step k is defined by the state of each cell of the grid: formally, the overall state maps a grid coordinate $\boldsymbol{x} \in X_{w,h}$ to a state $s^{(k)}_{\boldsymbol{x}} \in S$.

A CA (w, h, S, n, r, r_0) evolves over time as follows:

$$s^{(k+1)}_{\boldsymbol{x}} = \begin{cases} r_0(\boldsymbol{x}) & \text{if } k = 0 \\ r\left(s^{(k)}_{\boldsymbol{x}_{n,1}}, s^{(k)}_{\boldsymbol{x}_{n,2}}, \dots\right) & \text{otherwise,} \end{cases}$$

where $(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}, \dots) = n(\boldsymbol{x})$ is the neighborhood of $\boldsymbol{x}$.

In this study, we employ a square (*i.e.*, $w = h = \ell$) and toroidal grid and use the widely adopted Moore neighborhood n_{Moore} [24]. The neighborhood $n_{\text{Moore}}(\boldsymbol{x}, \ell, \rho)$ contains the $(2\rho + 1)^2$ positions which lay at a Manhattan distance not greater than ρ from $\boldsymbol{x}$, where $\rho \in \mathbb{N}$ is the radius of the neighborhood. By convention, we set $\boldsymbol{x}$ as the first element of the neighborhood $n(\boldsymbol{x})$, *i.e.*, the first argument of the transition rule r is $s^{(k-1)}_{\boldsymbol{x}_{n,1}} = s^{(k-1)}_{\boldsymbol{x}}$.

With a square grid and the Moore neighborhood of ρ radius, the CA employed in this work is uniquely defined by the tuple (ℓ, S, ρ, r, r_0).

3 CAbEA: A CA-BASED EA

3.1 Problem statement and requirements

In this study, we focus on optimization problems where candidate solutions can be compared based on a partial order.

More formally, we define a *problem* as a pair $(S, \prec)$, where S is the set of possible *solutions* (or solution space) and $\prec$ is a partial order defined on S. We write $s_1 \prec s_2$ if a solution s_1 is better than a solution s_2. Solving a problem $(S, \prec)$ means finding the set $S^\star \subseteq S$ of all the best solutions, *i.e.*, $S^\star = \{s \in S : \forall s' \in S, s' \not\prec s\}$.

We here propose a CA-based EA (CAbEA) as a way to solve any problem $(S, \prec)$ requiring only: (a) a way to randomly generate solutions, (b) a way to stochastically variate a solution, and (c) a way to stochastically combine two solutions. Formally, the first requirement is a probability distribution $B \in \mathcal{P}_S$ on S, where $\mathcal{P}_S$ denotes the set of probability distributions over S, the second is a *mutation* operator $o_{\text{mut}} : S \to \mathcal{P}_S$, and the third is a *crossover* operator $o_{\text{xover}} : S \times S \to \mathcal{P}_S$. We use $s \overset{\Delta}{\sim} P$ to indicate that $s \in S$ is sampled from the distribution $P_S \in \mathcal{P}_S$.

In evolutionary computation (EC) terms, B, o_{mut}, and o_{xover} are related to the representation of the solutions. Indeed, an optimization problem can been represented (in a less general and EC-friendly way) as a tuple $(G, P, F, \phi, f, \prec)$, where G is a genotype space, P is the phenotype space, F is the fitness space, $\phi : G \to P$ is a genotype-phenotype mapping function, $f : P \to F$ is a fitness function, and $\prec$ is a partial order defined over F. Solutions, which are triplets $s \in G \times P \times F$, are compared based on their fitness. Mutation and crossover alter genotypes, with ϕ and f generating the phenotype and fitness. The mapping ϕ has to be consistent with B, o_{mut}, and o_{xover} as they all operate within G.

Despite the formal complexity, B, o_{mut}, and o_{xover} are basic and fundamental components commonly required by many EAs. CAbEA, requiring only these components and imposing no constraints on the solution space S (hence neither on G, P, and F), has a very broad and general applicability, the same of a standard GA. In particular, both CAbEA and GA do not require S (or G) to be a numerical space (like most variants of evolutionary strategies (ES) [22], particle swarm optimization (PSO) [48], differential evolution (DE) [16], and many others), nor the fitness to be a single number or a few numbers (like nondominated sorting genetic algorithm (NSGA-II) [14]); neither they require additional components, as, e.g., a set of descriptors to map solutions to a further numerical space (like multi-dimensional archive of phenotypic elites (MAP-Elites) [35]), or a similarity metric over S (or P).

3.2 Working of CAbEA

CAbEA iteratively evolves a population of solutions which lay in a CA grid, aligning each iteration of the EA with a single time step of the CA. Formally, CAbEA can be expressed as a CA (ℓ, S, ρ, r, r_0) where ℓ and ρ are parameters of the EA, S is the set of possible solutions (which hence coincides with the set of states of the CA), the initialization rule is $r_0(\boldsymbol{x}) = s$, with $s \overset{\Delta}{\sim} B$, and the transition rule is:

$$r(\boldsymbol{x}) = \begin{cases} \max_{\prec}\left(s^{(k)}_{\boldsymbol{x}}, o_{\text{mut}}\left(s^{(k)}_{\boldsymbol{x}}\right)\right) & \text{if } r \overset{\Delta}{\sim} U([0,1]) \geq p_{\text{xover}} \\ \max_{\prec}\left(o_{\text{xover}}\left(s^{(k)}_{\boldsymbol{x}}, s^{(k)}_{\boldsymbol{x}_{n,j}}\right)\right) & \text{otherwise,} \end{cases}$$

where $\max_{\prec}(s, s')$ is s' if $s' \prec s$ or s otherwise and j is randomly sampled as $j \triangleq U(\{2, \ldots, |n(\boldsymbol{x})|\})$.

In other words, at the beginning of the execution of CAbEA, each one of the ℓ^2 grid cells is populated with a solution sampled randomly from B. Then, at each iteration, CAbEA updates each cell as follows: with $1 - p_{\text{xover}}$ probability, it applies mutation operator to the solution in the cell at $\boldsymbol{x}$ and replaces the solution only if the mutated one is better. Otherwise, with p_{xover}, it applies the crossover operator to the solution at $\boldsymbol{x}$ and one randomly chosen solution in the Moore neighborhood $n_{\text{Moore}}(\boldsymbol{x}, \ell, \rho)$ of $\boldsymbol{x}$ and replaces the solution only if the one obtained with crossover is better. The process repeats until n_{evals} comparisons using $\prec$ are completed, with each iteration involving ℓ^2 comparisons. Upon the last iteration, CAbEA returns all the solutions in the grid. p_{xover} and n_{evals} are parameters of the EA, together with ℓ and ρ.

We remark that, being based on the distributed computing paradigm of the CA, the execution of CAbEA is largely parallelizable.

Interpretation. CAbEA shares the basic requirements and, hence, the broad applicability of a standard GA. By standard GA we refer to an EA where, at each iteration, n_{pop} solutions are generated from the existing n_{pop} ones by stochastically applying either mutation (with a probability $1-p_{\text{xover}}$) or crossover (with a probability p_{xover}) based on a specific reproduction selection criterion. Afterwards, the population is reduced back to n_{pop} solutions including new and previous solutions according to some other removal selection criterion. Tournament selection, which imposes no additional constraints on the problem or its representation, is often used as reproduction selection criterion. Similarly, a common criterion for trimming the population that does not limit the EA applicability is randomly removing a solution that is not better than any other.

The key difference between CAbEA and a standard GA lies in the selection of partners during the crossover phase. Given a parent s_1, while in GA the other parent s_2 may be any solution in the population (potentially even s_1 itself), in CAbEA s_2 is chosen from the neighborhood of the grid cell containing s_1, ensuring proximity between parents. In other words, CAbEA, by storing the population in a grid, progressively induces a sort of spatial structuring of the solutions throughout the evolutionary process. Such a spatial structure may allow for the emergence and maintenance of multiple and different optimal solutions, hence mitigating the risk of premature convergence.

The idea of using a grid structure and a CA mechanism to host and evolve population, and the fact that it can be beneficial for the evolutionary process, in particular for balancing exploitation and exploration [11] through diversity [49], is not new [4, 17]. In this paper, we propose to further strengthen this capability of CAbEA by employing a structured grid, as explained in the next section.

3.3 Non-flat substrate

To further slow down the spread of locally dominant solutions in the population, we have modified CAbEA to operate on a grid where specific cells are left empty through the entire evolutionary process. These empty cells act as barriers, preventing the spread of traits from a solution (when it acts as the second parent in a crossover) to neighboring cells. Moreover, the strategic location of these empty cells can promote the development and coexistence of multiple high-quality solutions. These solutions, being located in different regions of the grid separated by these barriers, can evolve independently without competing with each other.

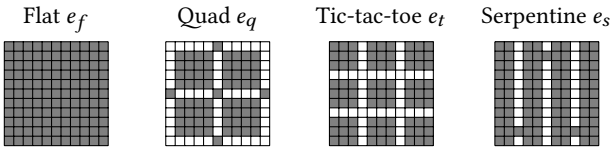

Figure 1: The four substrates considered in this study: the white background denotes empty cells.

Formally, we call *substrate* a predicate $e : \{1, \ldots, \ell\} \times \{1, \ldots, \ell\} \to \{\text{true}, \text{false}\}$. For a given cell coordinate $\boldsymbol{x}$, the corresponding cell is empty if $e(\boldsymbol{x})$ is true. In this modified version of CAbEA, the state of the CA is represented by $S \cup \{\varnothing\}$ instead of just S, where $\varnothing$ indicates an empty cell. Once initialized, empty cells are never updated. Moreover, the neighborhood does not take into consideration empty cells, *i.e.*, $n(\boldsymbol{x}) = \{\boldsymbol{x}' \in n_{\text{Moore},\ell,\rho}(\boldsymbol{x}) : \neg e(\boldsymbol{x}')\}$, possibly resulting in a smaller local set of candidate parents.

When the substrate e is such that $\forall \boldsymbol{x}, e(\boldsymbol{x}) = \text{true}$, CAbEA works as described in the previous section. We call *flat* this substrate and denote it by e_f. We show in Figure 1 (for $\ell = 11$) the four substrates we experimented with in this study.

One important consequence of not updating empty cells is that the number of $\prec$ comparisons (and hence fitness evaluations, in the case the problem includes f, F) at each iteration is lower than ℓ^2.

4 EXPERIMENTAL EVALUATION

To verify our hypothesis that a substrate with empty cells increases CAbEA ability to solve optimization problems where diversification is useful, we conducted experiments focusing on two problem types: multi-modal (MM) problems and multi-objective (MO) problems. Both are particular instances of the general case $(S, \prec)$ and in both cases the set $S^\star$ of best solutions is not a singleton. Therefore, we expect that an EA like CAbEA, which can diversify the exploration of the search space thanks to the non-flat substrate, will outperform an EA lacking this ability.

We considered a synthetic MM and a synthetic MO problem, each one instantiated twice with different representations. To contextualize CAbEA performances, we also tested other EAs, namely, standard GA, MAP-Elites, PSO (namely, the standard variant presented in [8]), DE (namely, the DE/rand/1 used in [32]), and NSGA-II. Furthermore, to verify in practice the wide applicability of CAbEA, we evaluated six symbolic regression (SR) problems, using a different representation than the one employed in the MM and MO synthetic problems.

4.1 Performance on MM problems

4.1.1 Synthetic MM problem. We considered a synthetic MM problem where the goal is to find the h target solutions denoted as $\{s_1^\star, \ldots, s_h^\star\} = S^\star$. Here, the partial order $\prec$ is defined as a total order based on the proximity to the closest target solution. Formally, a solution s is considered better than another s', *i.e.*, $s \prec s'$, if and only if $\min_{s^\star \in S^\star} d(s, s^\star) < \min_{s^\star \in S^\star} d(s', s^\star)$, where d is a distance defined on S. In other words, this problem involves a

fitness function $f : S \to \mathbb{R}^+$ where $f(s) = \min_{s^\star \in S^\star} d(s, s^\star)$, and the goal is to minimize f.

To evaluate the EA effectiveness in solving this MM problem, *i.e.*, its ability to find all the targets, we also measured, besides the fitness $f(s^*)$ of the best solution s^*, the *overall distance to targets* of the set S' of solutions generated by the EA. Formally, given a set $\hat{S} \subseteq S$, we define the overall distance to targets of $\hat{S}$ as $\bar{d}\left(\hat{S}\right) = \frac{1}{h}\sum_{i=1}^{i=h} \min_{s\in\hat{S}} d\left(s, s_i^\star\right)$. This measure represents the average distance of each target to its closest solution in $\hat{S}$. For both f and $\bar{d}$, the lower, the better.

We instantiated the proposed MM problem twice, with two different representation. In the *discrete* representation, a solution is a string of integers with length p, *i.e.*, $S = \{0, \dots, u\}^p$. Here, the distance d is the normalized Hamming distance (*i.e.*, the Hamming distance divided by p), and consequently, the codomain is $[0, 1]$ for both f and $\bar{d}$. The h targets are defined as $\boldsymbol{s}_j^\star = (s_{j,1}^\star, \dots, s_{j,p}^\star)$ with $s_{j,i}^\star = u$ if $\left\lceil \frac{ih}{p} \right\rceil = j$ and 0 otherwise. For example, for $u = 9$, $p = 6$, and $h = 3$, the three targets are $\boldsymbol{s}_1^\star = (9, 9, 0, 0, 0, 0)$, $\boldsymbol{s}_2^\star = (0, 0, 9, 9, 0, 0)$, and $\boldsymbol{s}_3^\star = (0, 0, 0, 0, 9, 9)$. We experimented with $u = 9$ and different values for p and h. We denote by MM-D-p-h the variants of this problem.

In the *continuous* representation of the synthetic MM problem, a solution is a string of real numbers of length p, *i.e.*, $S = \mathbb{R}^p$. Here, the distance d is the Euclidean distance, and consequently, the codomain is $\mathbb{R}^+$ for both f and $\bar{d}$. Each of the h targets is defined as $\boldsymbol{s}_j^\star = \mathbf{1} + \frac{4\boldsymbol{r}}{\|\boldsymbol{r}\|_2}$, where $\mathbf{1}$ is the p-long vector of ones and $\boldsymbol{r}$ is a random vector in $\mathbb{R}^p$ where each element is sampled from the Gaussian distribution $N(0, 1)$—*i.e.*, the targets lay on a sphere of radius 4 centered in $\mathbf{1}$. To ensure consistency, we used the same random generator seed for sampling $\boldsymbol{r}$ in all the experiments, hence having always the same targets for each p, h. We denote by MM-C-p-h the variants of this problem.

4.1.2 *Experimental procedure and baselines.*

We selected different sets of EAs for the MM-D and the MM-C problem. Specifically, we tested CAbEA, GA, and MAP-Elites for the discrete problem. Regarding the continuous case, we additionally compared CAbEA with PSO and DE.

For MM-D we defined B, *i.e.*, the initial population sampling probability distribution, as a uniform random selection in $\{0, \dots, u\}$ for each element of the integer string. As mutation o_{mut}, we used the "bit-flip" mutation adapted to deal with more than two symbols, *i.e.*, we changed with probability $\frac{1}{p}$ each element of the integer string to a random value in the proper domain. Finally, as crossover o_{xover}, we used the uniform crossover, *i.e.*, we set each element of the integer string to the corresponding element of one of the parents, chosen with uniform probability. After every crossover, we applied the mutation described above.

For MM-C we used as B the random sampling in $U([-1, 1])$ for each element of the numerical vector. As o_{mut}, we employed the Gaussian mutation where we added a noise sampled from $N(0, \sigma_{\text{mut}})$ to each element of the vector, with $\sigma_{\text{mut}} = 0.35$. As o_{xover}, we used the segment crossover, where $\boldsymbol{s}' = \alpha\boldsymbol{s}_1 + (1 - \alpha)\boldsymbol{s}_2$, with $\alpha \overset{\Delta}{\sim} U([0, 1])$, followed by a Gaussian mutation.

Regarding the representation agnostic EA parameters, we set them as follows. For GA, we used a population of $n_{\text{pop}} = 100$ individuals, we produced the offspring using crossover with probability $p_{\text{xover}} = 0.8$ and mutation with probability $1 - p_{\text{xover}}$, and we selected parents with tournament selection of size $n_{\text{tour}} = 5$. For MAP-Elites, we used two descriptors which, given a solution $\boldsymbol{s}$, measure the average of the even or odd elements, respectively. For each descriptor, we considered 20 bins, resulting in a square grid archive of 400 cells; we generated 100 new individuals at each iteration. While we acknowledge that the descriptors play a crucial role in the performance of MAP-Elites, choosing the best descriptors for the two synthetic problems considered here was not a goal of this study. However, we remark that the chosen descriptors are (a) correlated with the fitness and (b) mutually independent, hence allowing for full coverage of the archive. For PSO, we used a standard implementation with $n_{\text{pop}} = 100$ particles, the cognitive and social coefficients $\phi_p = \phi_g = 1.5$, and the inertia weight $w = 0.8$. For DE, we used a standard implementation with $n_{\text{pop}} = 15$ individuals, the crossover probability $p_{\text{xover}} = \text{CR} = 0.8$, and the differential weight $d_w = \text{F} = 0.5$. Finally, for CAbEA we experimented with $\rho = 1$, $p_{\text{xover}} = \frac{1}{3}$, two values for ℓ, 11 and 21, and two substrates, the *flat* one, where $e_f(\boldsymbol{x}) = \text{false}$, and one corresponding to a sort of macro-grid of four square groups of cells (*quad*, denoted by e_q), shown in Figure 1 for $\ell = 11$—when scaled to $\ell = 21$, we kept the width of empty cells barriers to 1. Note that the five non-empty cells in e_q (at the center of the grid and at the midpoints of the sides) allow for the transfer of genetic material (through crossover and thanks to the Moore neighborhood of radius $\rho = 1$) between the four groups of cells, which are hence not isolated. We denote each of the four variants of CAbEA with CAbEA-ℓ-e, where e represents the substrate chosen. For all the EAs, we set the termination criterion to be the completion of $n_{\text{evals}} = 50\,000$ solution comparisons, which is equivalent to the number of fitness evaluations performed.

We executed the experiments with JGEA [31]. We made the code for reproducing the experiments publicly available at https://github.com/gpietrop/CAbEA.

We executed 30 independent runs for each (problem, EA) pair, by varying the random seed. When comparing CAbEA variants with same ℓ and different substrates, we tested for statistical significant difference of means with the Wilcoxon rank-sum test for pairwise comparisons, setting the significance level at $\alpha = 0.05$.

4.1.3 *Results and discussion.*

Discrete representation. Figure 2 shows the results for the MM-D-p-3 problems (with $p \in \{25, 100, 250\}$), *i.e.*, the synthetic MM problem with the discrete representation and 3 targets. The figure shows on the top the fitness $f(s^*)$ of the best individual s^* during the evolution and on the bottom the overall target distance $\bar{d}(S')$ of the entire population S' during the evolution. As specified above, for this problem, we compared CAbEA, GA, and MAP-Elites.

The principal observation we draw from Figure 2 is that CAbEA-11 (*i.e.*, with $\ell = 11$) always achieves the best overall target distance $\bar{d}(S')$, regardless of the substrate being used. For the smallest problem, MM-D-25-3, both CAbEA-11 and CAbEA-21-e_q (non-flat substrate) often find all the three targets (*i.e.*, achieve $\bar{d}(S') = 0$), while CAbEA-21-e_f (flat substrate) does not.

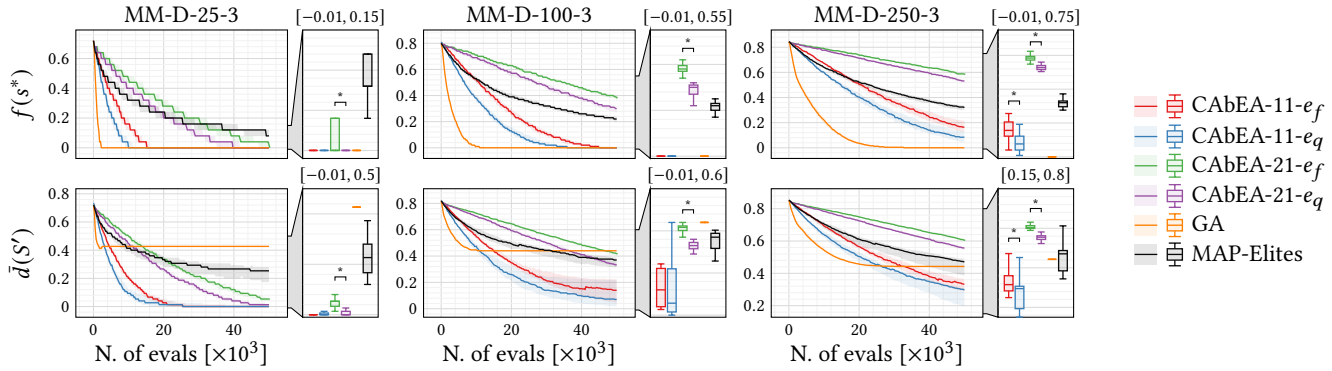

Figure 2: Results on the MM-D-p-3 problem: fitness $f(s^*)$ of the best individual s^* and overall target distance $\bar{d}(S')$ of the population S' during the evolution. For both, the line plots show the median and the interquartile range across the 30 runs and the boxplots show the final distribution. An asterisk $\overset{*}{\frown}$ above a pair of boxplots related to CAbEA-ℓ represents statistically significant difference of means.

Regarding the specific research question of this study, the results reported in Figure 2 do not allow to draw sharp conclusions. We observed that the e_q substrate systematically allows CAbEA to achieve faster convergence of both $f(s^*)$ and $\bar{d}(S')$ compared to the e_f substrate, regardless of ℓ or the specific problem. The final value of the overall target distance is significantly better in four on six cases for e_q, and never worse.

From a broader point of view, Figure 2 shows that CAbEA achieves a better overall target distance at the expense of the efficiency in reaching the perfect fitness when compared against GA. For all the problems, GA achieves $f(s^*) = 0$ more quickly than CAbEA (and than MAP-Elites): this is the other side of the coin of slowing the spread of good solutions traits in the population and is a consequence of how the crossover operator acts in GA and CAbEA. While in the former the tournament selection tends to choose more frequently the best individuals, regardless of their "position" in the population, in CAbEA the crossover only involves neighbor individuals. MAP-Elites, which is a mutation-only EA, is even slower in convergence. However, due to its intrinsic ability to maintain diversity, MAP-Elites surpasses GA in $\bar{d}(S')$, like CAbEA. We remark, however, that MAP-Elites requires to define the descriptors to promote diversity, while CAbEA does not impose any requirement.

To gain deeper insights into the CAbEA population dynamics, particularly regarding the impact of using a non-flat substrate, we analyzed in detail the population at different stages of the evolution with CAbEA-21. Specifically, Figure 3 shows the CAbEA variant operating with a flat substrate (top) and with a substrate including empty cells (bottom). Each colormap is a snapshot of the CA at a given iteration: in each cell of the grid, the color represents the fitness $f(s)$ (top row of plots) of the hosted solution or its distance $d(s, s_i^\star)$ (bottom rows) to the i-th target. These plots confirm that solutions hosted in different regions of the substrate tend to "cluster" in terms of which targets they are aiming at. Moreover, the comparison in Figure 3 between CAbEA-21-e_q with CAbEA-21-e_f, shows that in the former, those regions appear partially delimited by the barriers constituted by the empty cells. *I.e.*, the non-flat substrate appears to play a role in the evolutionary process.

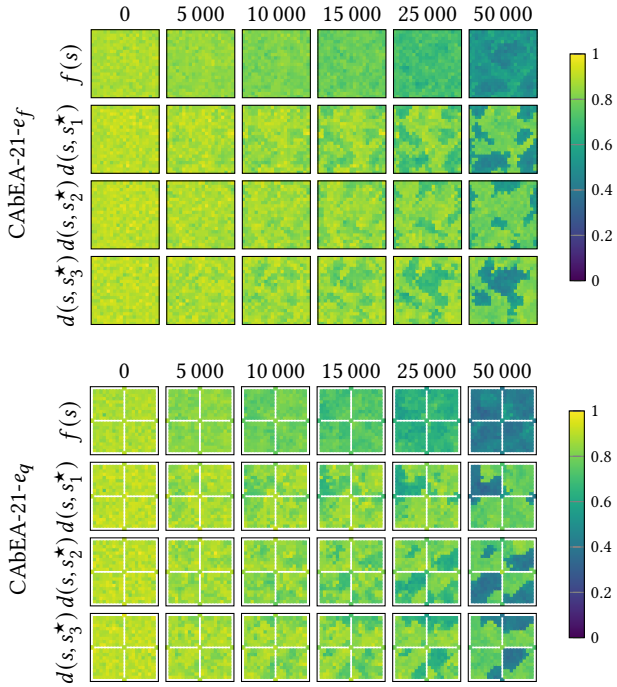

Figure 3: CAbEA-21 on the MM-D-100-3 problem. Each colormap is a snapshot of the CA at a given iteration (the corresponding number of $<$ evaluations is shown on the top of each column of plots): in each cell of the grid the color represents the fitness $f(s)$ of the hosted solution or its distance $d(s, s_i^\star)$ to the i-th target.

Continuous representation. Figure 4 shows the results for the MM-C-p-3 problems (with $p \in \{10, 25, 50\}$), with the same structure of Figure 2.

The results shown in Figure 4 tell a different story compared to the ones of the discrete results. The ability to reach all the targets is here clearly affected by the problem size p. In MM-C-10-3 the only

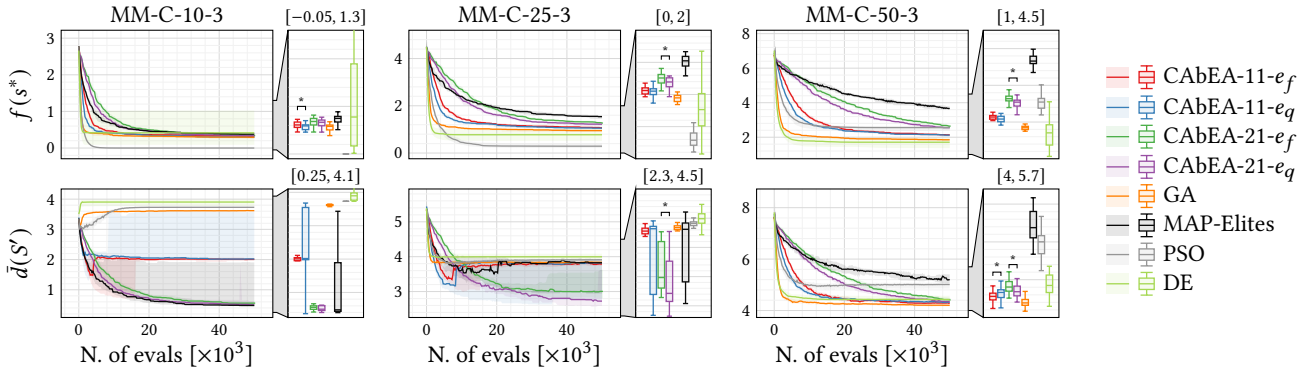

Figure 4: Results on the MM-C-p-3 problem (same visual syntax of Figure 2).

EAs that consistently achieve values of the overall target distance $\bar{d}(S')$ close to 0 are CAbEA-21 (with both substrates) and MAP-Elites. CAbEA-11 appears to reach only two of the three targets (with CAbEA-11-e_q sometimes reaching all of them, likely thanks to the non-flat substrate). All the other EAs, *i.e.*, GA, PSO, and DE, reach only one target. Interestingly, CAbEA-11-e_f initially (up to $\approx$ 8 000 fitness evaluations) appears to be able to track three targets, but then loses one of them. Regarding the mid-size problem MM-C-25-3, only CAbEA-21 is able to often reach two on three targets, while the other EAs fail to reach more than one of them—with CAbEA-11-e_q and MAP-Elites occasionally hitting two. Finally, in the MM-C-50-3 none of the EAs is able to reach more than one target.

The different findings on MM-D and MM-O problems can be attributed to the relative position of the targets and the initial distribution of the solutions (determined by the same B for all the EAs). In MM-D the targets are at the same distance from the mode of the distribution (*i.e.*, $\left(\left\lfloor \frac{u}{2} \right\rfloor, \ldots, \left\lfloor \frac{u}{2} \right\rfloor\right) \in \mathbb{N}^p$), and hence maintaining a population able to aim at all of them at the same time is easy. Differently, in MM-C some targets are closer to the mode of the distribution (*i.e.*, $\mathbf{0} \in \mathbb{R}^p$): the population tends hence to converge to the closest target and maintaining diversity is harder. Nevertheless, CAbEA outperforms other EAs in reaching multiple targets at the same time; in terms of final $\bar{d}(S')$, e_q is significantly better than e_f in two on six cases and worse in one.

As for the discrete case, we show in Figure 5 a representation of the population at different stages of the evolution using CAbEA-11 on MM-C-25-3. For CAbEA-11-e_f, there is no evident pattern in the substrate: all the solutions are indeed converging to $s_2^\star$. In contrast, the CAbEA-11-e_q variant benefits from the presence of the empty cells, which permit a small subset of the solutions to converge to $s_1^\star$, while all the remaining ones converge to $s_2^\star$. In both figures, it can be seen that the solutions are farther to $s_3^\star$ than to the other two targets since the early stages of the evolution: this further confirms why all the EAs struggle in reaching this target.

4.1.4 More substrates. Since the experiments with MM-D and with MM-C suggested that the substrate plays some role when CAbEA faces MM problems, we experimented with two other substrates and with different numbers of targets. For this analysis, we considered exclusively the MM-D-100-h problems, with $h \in \{2, 3, 5, 8\}$, $\ell = 11$,

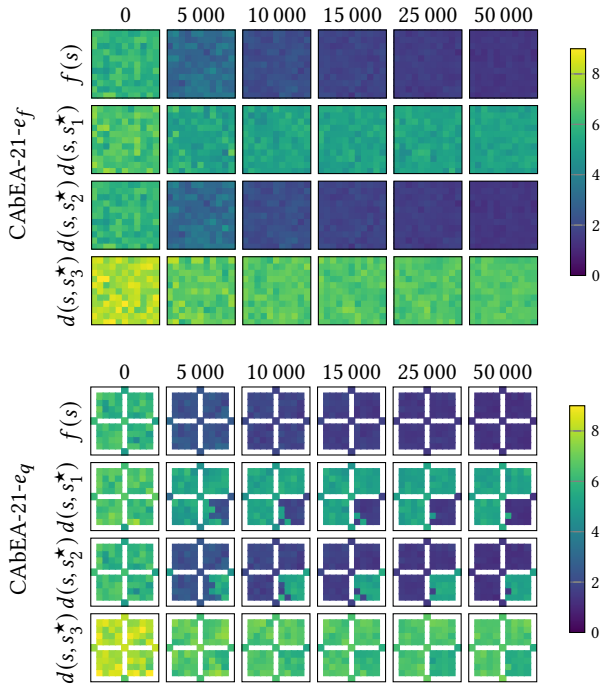

Figure 5: CAbEA-11 on the MM-C-25-3 problem (same visual syntax of Figure 3).

and two other substrates (shown in Figure 1): a *tic-tac-toe*-like substrate e_t, where the grid is divided by two vertical and two horizontal lines of empty cells, and a *serpentine* substrate e_s, where the grid is divided into four vertical stripes by three lines of empty cells interrupted in one cell on the top or bottom of the grid on even or odd lines, respectively.

Figure 6 shows the results of this experiment: since all the variants of CAbEA behave similarly for what concerns $f(s^*)$, and due to space constraints, we report here only the overall target distance $\bar{d}(S')$.

While Figure 6 appears to confirm the general tendency of the e_q substrate to converge slightly faster than e_f, also with different numbers of targets, it can also be seen that there are no significant differences between substrates in the final value of $\bar{d}(S')$ in the

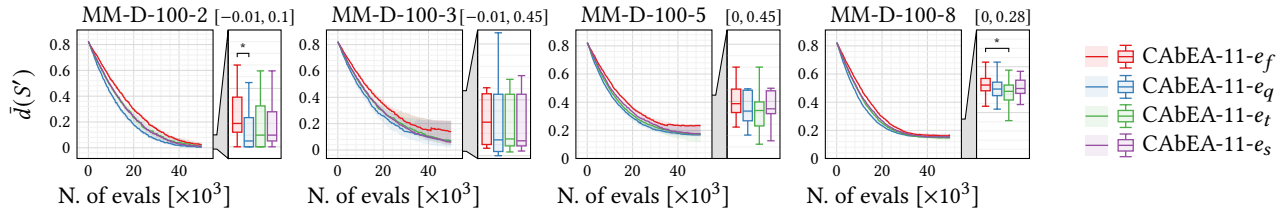

Figure 6: Results of CAbEA-11 on the MM-D-100-h problem (same visual syntax of Figure 2).

vast majority of cases. The flat substrate appears to be the slowest in all the problems and the one with the worst median final $\bar{d}(S')$. Interestingly, the convergence speed of the four variants aligns with the number of non-empty cells within their substrates: 69, 85, 91, and 121 respectively for e_q, e_t, e_s, and e_f. While this finding is in general consistent with the widespread belief that a smaller population in an EA favors exploitation [7], and hence speed of convergence, in CAbEA and according to our experiments, this is not obtained at the expenses of exploration ability.

4.2 Performance on MO problems

4.2.1 Synthetic MO problem. We considered two synthetic MO problems derived from the MM-D and MM-C problems defined in Section 4.1.1. Namely, for a given a MM-r-p-h problem, with r representing the discrete (D) or continuous (C) representation, we built a MO-r-p-h problem where S is equal to the one of the corresponding MM problem, $f : S \to \mathbb{R}^h$ maps a solution s to the tuple $f(s) = (f_1(s), \ldots, f_h(s)) = (d(s, s_1^\star), \ldots, d(s, s_h^\star))$ of its distances to the h targets, and $\prec$ is the Pareto-dominance partial order on $\mathbb{R}^h$. We experimented with $h = 2$ and the same values of p used in the previous experiments.

For measuring the capability of an EA to find solutions that perform well across both objectives f_1 and f_2, we considered the hyper-volume of the non-dominated solutions in the population. Formally, given a set $\hat{S} \subseteq S$ of solutions and two reference points $\boldsymbol{f}_{\min}, \boldsymbol{f}_{\max} \in \mathbb{R}^2$, we computed the hyper-volume $v(\hat{S})$ by first isolating the non-dominated solutions $\hat{S}' \subset \hat{S}$. Then we computed the area of the polygon delimited by $\boldsymbol{f}_{\max}$, $(f_{\max,1}, f_{\min,2})$, $(f_{\min,1}, f_{\max,2})$ and the points given by applying f to the solutions in $\hat{S}'$. We used $\boldsymbol{f}_{\min} = (0, 0)$ for both representations, $\boldsymbol{f}_{\max} = (1, 1)$ for the discrete representation, and $\boldsymbol{f}_{\max} = (7, 7)$ for the continuous one—the latter chosen considering the position of the target solutions and the initial distribution of the solutions given by B. For the hyper-volume, the greater, the better.

4.2.2 Experimental procedure and baselines. We followed the same procedure of Section 4.1.2, but using different EAs for the comparison of the performances, suitable for a MO problem. Namely, we considered CAbEA, GA, MAP-Elites, and NSGA-II, for which we used the same genetic operators of GA. We remark that NSGA-II requires the fitness to be defined in $\mathbb{R}^h$, to sort solutions according to their crowding distance; the other three EAs just require a partial order $\prec$ on S (with MAP-Elites further requiring the descriptors).

4.2.3 Results and discussion.

Discrete representation. Figure 7 shows the results for the MO-D-p-2 problems, with $p \in \{25, 100, 250\}$.

There are two principal observations we draw based on the plots in Figure 7. Similar to findings in the MM context, the e_q substrate in CAbEA generally leads to quicker convergence for all the tested ℓ values. However, this difference does not consistently correspond to a significantly larger final hyper-volume: e_q is better in four and worse in two cases on six.

Second, comparing CAbEA against the other EAs, CAbEA-11 outperforms MAP-Elites consistently and GA for $p = 25$ and $p = 100$, whereas, for $p = 250$, no variant of CAbEA reaches a plateau in $v(S')$ and only CAbEA-11-e_q matches the performance of GA. On the other hand, NSGA-II achieves the largest hyper-volume $v(S')$ in all the cases. This is expected since NSGA-II is the only EA that is designed to exploit the distribution of solutions in the fitness space, *i.e.*, the space where the hyper-volume is computed.

Continuous representation. Figure 8 shows the results for the MO-C-p-2 problems, with $p \in \{10, 25, 50\}$.

Here, the impact of the substrate on the ability to achieve a large hyper-volume appears negligible and there are no statistical differences between e_q and e_f. Similarly, regarding the speed of convergence, the advantage of using a non-flat substrate appears thin.

Finally, and maybe more importantly, CAbEA appears to be worse than GA (but not worse than MAP-Elites) on the largest problem (MO-C-50-2) despite reaching a plateau in $v(S')$.

4.3 Beyond synthetic problems

We performed a further set of experiments on problems based on a different representation, also to confirm the wide applicability of CAbEA.

We considered six variants of SR problem. Formally, in SR S is the set of mathematical expressions built using a set O of (unary or binary) mathematical operators, a set V of variables, and a set $C \subseteq \mathbb{R}$ of constants. Given a dataset $D = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{i=n}$, with $\boldsymbol{x}^{(i)} \in \mathbb{R}^p$ and $y \in \mathbb{R}$, the fitness of a solution s is its mean squared error (MSE) on D, *i.e.*, $f(s) = \mathrm{MSE}(s, D) = \frac{1}{n}\sum_{i=1}^{i=n}(s(\boldsymbol{x}^{(i)}) - y^{(i)})^2$, and $\prec$ is the natural (total) order in $\mathbb{R}$ applied to values of $f(s)$.

We compared CAbEA and GA, the two EAs with the widest applicability and least requirements, and employed a tree-based representation: a solution s is structured as a tree where terminal nodes are labeled with $V \cup C$ elements and non-terminal nodes are labeled with O elements and have the appropriate number of child nodes. We defined the operator set as $O = \{+, -, \times, \div^*, \log^*\}$, with $\div^*$ and $\log^*$ being the protected versions of their counterparts, the constant set as $C = \{0.1, 1, 10\}$, and the variable set as $V = \{x_1, \ldots, x_p\}$, p being problem-dependent. We used the standard subtree mutation as o_{mut} and the standard subtree crossover as

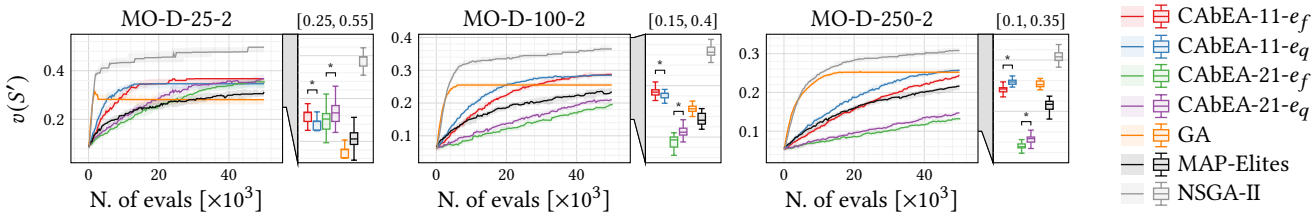

Figure 7: Results on the MO-D-p-2 problem (same visual syntax of Figure 2).

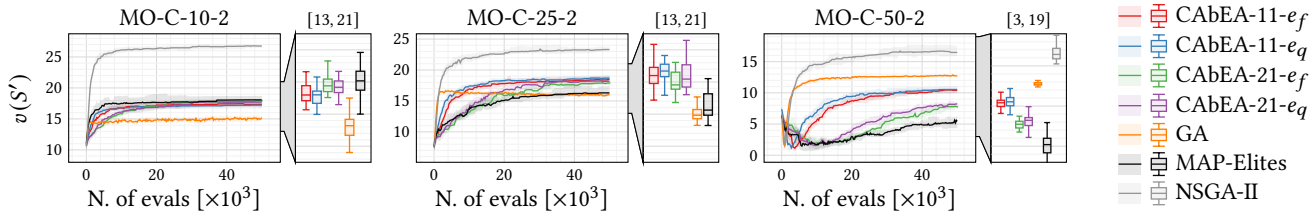

Figure 8: Results on the MO-C-p-2 problem (same visual syntax of Figure 2).

o_{xover}; for the probability distribution B, we relied on the ramped half-and-half procedure. For all o_{mut}, o_{xover}, and B, we constrained the tree depth within the range $[4, 10]$. When coupled with our GA, this representation corresponds to genetic programming (GP) [25].

Concerning the problems, we considered three SR problems where D is obtained by sampling a target function $s^\star$ and three additional problems where D is derived from a real-world dataset—we chose all of them based on the considerations of [53]. In the former group, we included Keijzer-6 ($p = 1$, $|D| = 50$), Vladislavleva-4 ($p = 5$, $|D| = 1024$), and Nguyen-7 ($p = 1$, $|D| = 20$). In the latter, we included the "concrete" dataset ($p = 8$, $|D| = 825$, $|D'| = 206$), the "energy efficiency" dataset ($p = 8$, $|D| = 615$, $|D'| = 154$), and the "wine" dataset ($p = 11$, $|D| = 3919$, $|D'| = 980$). For the problems based on real-world datasets, besides measuring the fitness $f(s^*)$ of the best solution, we also measured its MSE on a separate holdout dataset D', disjoint from D, to investigate the solutions generalization capabilities.

For this experiment, we set $n_{\text{evals}} = 10\,000$. Figure 9 shows the results in terms of the final $f(s^*)$ (for target-based problems) or $\text{MSE}(s^*, D')$ (for the dataset-based problems—we verified that they are qualitatively similar to those of $f(s^*)$).

Experimental results on the SR problems are consistent with those observed in previous experiments. The non-flat substrate e_q leads to a slightly faster convergence (for space constraints, we do not show the plots with $f(s^*)$ and $\text{MSE}(s^*, D')$ during the evolution) and marginally better results than e_f: however, the difference is almost never significant. Moreover, Figure 9 confirms the gap of performance of CAbEA with respect ot GA (here as GP), and the advantage of CAbEA-11 with respect to CAbEA-21.

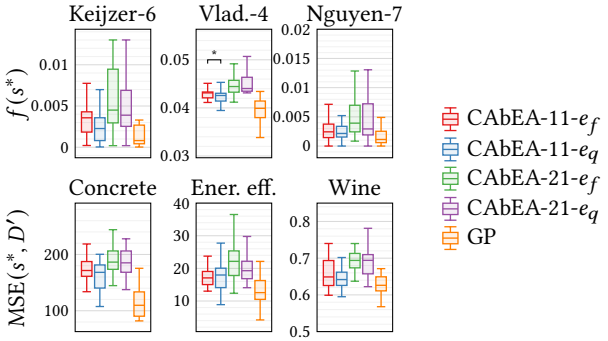

Figure 9: Results on SR problems: final distribution of the fitness $f(s^*)$ (for target-based problems) or the MSE(s^*, D') on the holdout dataset (for the dataset-based problems) of the best individual s^*.

5 CONCLUSIONS AND FUTURE WORK

In this study, we explore the impact of incorporating empty cells as barriers within a CAbEA substrate to promote diversity and improve performances in the evolutionary search. To this aim, we conducted a comprehensive experimental campaign to assess the influence of the substrate on MM, MO, and SR problems.

This research paves the way for multiple possible future developments. Exploring the effectiveness of CAbEA in real-world problems, such as evolutionary robotics [36] and policy search [28], is an interesting direction, given the success of QD algorithms such as MAP-Elites in this areas [12]. We also aim at comparing our proposed method with EAs that explore different subpopulations, such as the island model [54]. We envision that CAbEA may be applied to other cellular GA approaches, including cGA [23] or TBGA [20], and we aim at an extensive comparison with those methods in the future.

Additionally, refining CAbEA initialization method to strategically place solutions throughout the grid could improve its performance. Instead of the current approach of random placement, this method would aim to distribute solutions across distinct zones of the grid (especially in zones divided by empty cells), promoting initial diversity and potentially contributing to a more robust evolutionary process. Finally, incorporating open-ended evolution techniques into CAbEA might also allow for the continuous generation of novel and diverse solutions, expanding its capabilities.

ACKNOWLEDGMENTS

The authors wish to thank Luca Manzoni for their insights and discussions on preliminary versions of this work. This work was partially financed by the Research Council of Norway's DeepCA project, grant agreement 286558 and by the Oslo Metropolitan University (Norway), and partially carried out within the PNRR research activities of the consortium iNEST (Interconnected North-East Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) - Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS_00000043).

REFERENCES

[1] Enrique Alba and Bernabé Dorronsoro. 2005. The exploration/exploitation trade-off in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation* 9, 2 (2005), 126–142.
[2] Enrique Alba and Bernabé Dorronsoro. 2008. Introduction to cellular genetic algorithms. In *Cellular Genetic Algorithms*. Springer, 3–20.
[3] Enrique Alba and Marco Tomassini. 2002. Parallelism and evolutionary algorithms. *IEEE transactions on evolutionary computation* 6, 5 (2002), 443–462.
[4] Enrique Alba and José Ma Troya. 2000. Cellular evolutionary algorithms: Evaluating the influence of ratio. In *International Conference on Parallel Problem Solving from Nature*. Springer, 29–38.
[5] Lorenzo Bonin, Luigi Rovito, Andrea De Lorenzo, and Luca Manzoni. 2024. Cellular geometric semantic genetic programming. *Genetic Programming and Evolvable Machines* 25, 1 (2024), 1–32.
[6] Erick Cantu-Paz. 2000. *Efficient and accurate parallel genetic algorithms*. Vol. 1. Springer Science & Business Media.
[7] Mauro Castelli, Luca Manzoni, Sara Silva, Leonardo Vanneschi, and Aleš Popovič. 2017. The influence of population size in geometric semantic GP. *Swarm and Evolutionary Computation* 32 (2017), 110–120.
[8] Maurice Clerc. 2012. Beyond standard particle swarm optimisation. In *Innovations and Developments of Swarm Intelligence Applications*. IGI Global, 1–19.
[9] John Conway. 1970. The game of life. *Scientific American* 223, 4 (1970), 4.
[10] Matthew Cook. 2004. Universality in elementary cellular automata. *Complex systems* 15, 1 (2004), 1–40.
[11] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR)* 45, 3 (2013), 1–33.
[12] Antoine Cully and Yiannis Demiris. 2017. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation* 22, 2 (2017), 245–259.
[13] KA De Jong. 2002. Evolutionary computation: a unified perspective.
[14] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.
[15] Anh Viet Do, Mingyu Guo, Aneta Neumann, and Frank Neumann. 2024. Evolutionary Multi-Objective Diversity Optimization. *arXiv preprint arXiv:2401.07454* (2024).
[16] Vitaliy Feoktistov. 2006. *Differential evolution*. Springer.
[17] Carlos M Fernandes, Nuno Fachada, Juan LJ Laredo, JJ Merelo, and Agostinho C Rosa. 2020. Population sizing of cellular evolutionary algorithms. *Swarm and Evolutionary Computation* 58 (2020), 100721.
[18] José Ferreira, Mauro Castelli, Luca Manzoni, and Gloria Pietropolli. 2023. A Self-Adaptive Approach to Exploit Topological Properties of Different GAs' Crossover Operators. In *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 3–18.
[19] Dario Floreano and Claudio Mattiussi. 2008. *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT press.
[20] V Scott Gordon, Rebecca Pirie, Adam Wachter, and Scottie Sharp. 1999. Terrain-based genetic algorithm (tbga) modeling parameter space as terrain. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*. Citeseer, 229–235.
[21] Karol Gregor and Frederic Besse. 2021. Self-organizing intelligent matter: A blueprint for an ai generating algorithm. *arXiv preprint arXiv:2101.07627* (2021).
[22] Nikolaus Hansen, Dirk V Arnold, and Anne Auger. 2015. Evolution strategies. *Springer handbook of computational intelligence* (2015), 871–898.
[23] Georges R Harik, Fernando G Lobo, and David E Goldberg. 1999. The compact genetic algorithm. *IEEE transactions on evolutionary computation* 3, 4 (1999), 287–297.
[24] Jarkko Kari. 2005. Theory of cellular automata: A survey. *Theoretical computer science* 334, 1-3 (2005), 3–33.
[25] John R Koza. 1994. *Genetic programming II*. Vol. 17. MIT press Cambridge.
[26] Joel Lehman and Kenneth O Stanley. 2011. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 211–218.
[27] Bernard Manderick and Piet Spiessens. 1989. Fine-grained parallel genetic algorithms. In *Proc. 3rd International Conference on Genetic Algorithms*. 428–433.
[28] Francesco Marchetti, Gloria Pietropolli, Federico Julian Camerota Verdù, Mauro Castelli, and Edmondo Minisci. [n. d.]. Control Law Automatic Design Through Parametrized Genetic Programming with Adjoint State Method Gradient Evaluation. *Available at SSRN 4490005* ([n. d.]).
[29] John S McCaskill and Norman H Packard. 2019. Analysing emergent dynamics of evolving computation in 2D cellular automata. In *Theory and Practice of Natural Computing: 8th International Conference, TPNC 2019, Kingston, ON, Canada, December 9–11, 2019, Proceedings 8*. Springer, 3–40.
[30] David Medernach, Taras Kowaliw, Conor Ryan, and René Doursat. 2013. Long-term evolutionary dynamics in heterogeneous cellular automata. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. 231–238.
[31] Eric Medvet, Giorgia Nadizar, and Luca Manzoni. 2022. JGEA: a Modular Java Framework for Experimenting with Evolutionary Computation. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2009–2018.
[32] Eric Medvet, Stefano Seriani, Alberto Bartoli, and Paolo Gallina. 2020. Evolutionary optimization of sliding contact positions in powered floor systems for mobile robots. *at-Automatisierungstechnik* 68, 2 (2020), 97–109.
[33] Melanie Mitchell. 2005. Computation in Cellular Automata: A Selected Review. *Non-standard computation* (2005), 95–140.
[34] Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. 2020. Growing neural cellular automata. *Distill* 5, 2 (2020), e23.
[35] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).
[36] Giorgia Nadizar, Eric Medvet, and Karine Miras. 2022. On the schedule for morphological development of evolved modular soft robots. In *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 146–161.
[37] Giorgia Nadizar, Eric Medvet, Stefano Nichele, and Sidney Pontes-Filho. 2022. Collective control of modular soft robots via embodied Spiking Neural Cellular Automata. *arXiv preprint arXiv:2204.02099* (2022).
[38] Giorgia Nadizar, Eric Medvet, Kathryn Walker, and Sebastian Risi. 2023. A Fully-Distributed Shape-Aware Neural Controller for Modular Robots. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Lisbon, Portugal) *(GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 184–192.
[39] Antonio J Nebro, Juan J Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. 2009. Mocell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems* 24, 7 (2009), 726–746.
[40] Stefano Nichele, Mathias Berild Ose, Sebastian Risi, and Gunnar Tufte. 2017. Ca-neat: evolved compositional pattern producing networks for cellular automata morphogenesis and replication. *IEEE Transactions on Cognitive and Developmental Systems* 10, 3 (2017), 687–700.
[41] Sidney Pontes-Filho, Kathryn Walker, Elias Najarro, Stefano Nichele, and Sebastian Risi. 2022. A single neural cellular automaton for body-brain co-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 148–151.
[42] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI* 3 (2016), 40.
[43] Justin K Pugh, Lisa B Soros, Paul A Szerlip, and Kenneth O Stanley. 2015. Confronting the challenge of quality diversity. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. 967–974.
[44] Ettore Randazzo, Alexander Mordvintsev, Eyvind Niklasson, Michael Levin, and Sam Greydanus. 2020. Self-classifying MNIST digits. *Distill* 5, 8 (2020), e00027–002.
[45] Stuart J Russell and Peter Norvig. 2010. *Artificial intelligence a modern approach*. London.
[46] Palash Sarkar. 2000. A brief history of cellular automata. *Acm computing surveys (csur)* 32, 1 (2000), 80–107.
[47] Hiroki Sayama. 2023. Evolution of Self-Replicators within Cellular Automata: 25 Years After Evoloops. In *Proceedings of The Distributed Ghost Cellular Automata, Distributed Dynamical Systems, and Their Applications to Intelligence*.
[48] Yuhui Shi. 2004. Particle swarm optimization. *IEEE connections* 2, 1 (2004), 8–13.
[49] Giovanni Squillero and Alberto Tonda. 2016. Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization. *Information Sciences* 329 (2016), 782–799.
[50] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.
[51] Alexandre Variengien, Stefano Nichele, Tom Glover, and Sidney Pontes-Filho. 2021. Towards self-organized control: Using neural cellular automata to robustly control a cart-pole agent. *Innovations in Machine Intelligence* 1 (2021).
[52] John Von Neumann and Arthur W Burks. 1966. *Theory of self-reproducing automata*. University of Illinois Press.

[53] David R White, James McDermott, Mauro Castelli, Luca Manzoni, Brian W Goldman, Gabriel Kronberger, Wojciech Jaśkowski, Una-May O'Reilly, and Sean Luke. 2013. Better GP benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines* 14 (2013), 3–29.

[54] Darrell Whitley, Soraya Rana, and Robert B Heckendorn. 1999. The island model genetic algorithm: On separability, population size and convergence. *Journal of computing and information technology* 7, 1 (1999), 33–47.