Optimizing Cellular Automata through a Meta-model Assisted Memetic Algorithm

Donato D'Ambrosio[1], Rocco Rongo[2], William Spataro[1], and Giuseppe A. Trunfio[3]

[1] Department of Mathematics, University of Calabria, 87036 Rende (CS), Italy
[2] Department of Earth Sciences, University of Calabria, 87036 Rende (CS), Italy
[3] DADU, University of Sassari, 07041 Alghero (SS), Italy

Abstract. This paper investigates the advantages provided by a Meta-model Assisted Memetic Algorithm (MAMA) for the calibration of a Cellular Automata (CA) model. The proposed approach is based on the synergy between a global meta-model, based on a radial basis function network, and a local quadratic approximation of the fitness landscape. The calibration exercise presented here refers to SCIARA, a well-established CA for the simulation of lava flows. Compared with a standard Genetic Algorithm, the adopted MAMA provided much better results within the assigned computational budget.

Keywords: Cellular Automata, Model Calibration, Meta-modelling, Memetic Algorithms.

1 Introduction

Most applications of Cellular Automata (CA) models to the simulation of real complex systems, require a preliminary calibration process [1,2]. The latter consists of finding the unknown values of the model parameters in such a way that the outcomes of the model itself better correspond to the observed dynamics of the system under consideration. For such purpose, automated methods have been developed by defining calibration as a global optimization problem in which the solution in terms of parameter values must maximize a fitness measure [2,3]. Because of the size of the search space, such a process usually requires a large number of fitness evaluations, which consist of computationally expensive CA simulations. Hence, in dealing with CA calibration the use of parallel computing is often mandatory [2]. As shown in [4], an additional strategy for increasing the search efficiency may consists of the so-called meta-model assisted (or surrogate assisted) optimization [5], which is based on inexpensive surrogate functions able to approximate the fitness corresponding to the CA simulations.

However, in most cases the global search struggles to provide an accurate solution. This is often because search heuristics, for example based on the Genetic Algorithms (GA) operators, are more effective at exploring the search space

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 317–326, 2012.

rather than at the fine-tuning of a particular solution candidate. Therefore, a further enhancement of the classical calibration based on a pure global search approach, may be obtained introducing a local search (LS) phase, which in many applications proved to be capable of providing much more efficient and accurate global optimization processes. For example, in the hybrid GAs known as Memetic Algorithms [6] a sub-process of LS is introduced to refine individuals by more or less standard hill climbing procedures. As in the case of the meta-modelling approach, such hybridization has the main aim of increasing the overall efficiency of the optimization process (i.e., leading to better solutions within an assigned computational budget). In some recent applications, also Meta-model Assisted Memetic Algorithms (MAMAs) have been described and successfully applied to optimization problems [7,8]. However, to our knowledge the advantages provided by a MAMA for the calibration of CAs have not been explored. Some preliminary results in this direction are the object of this paper, in which a MAMA has been applied to the calibration of a well-established CA for the simulation of lava flows, namely the SCIARA model [2,9].

The paper is organized as follows. In section 2 the CA calibration problem is formalized. Section 3 describes in detail the tested MAMA. Section 4 illustrates the results of the numerical experiments and section 5 concludes the paper outlining possible future work.

2 Optimization of Cellular Automata

In many applications of the CA modelling approach the cells' transition function depends on a vector of constant parameters $\mathbf{p} = [p_1, \ldots, p_n]^T$, which belongs to a set Λ (e.g. [1,2]). In particular, the overall transition function Φ gives the global configuration $\Omega^{(t+1)}$ (i.e. the set of all cell states) at the step $t+1$ as:

$$\Omega^{(t+1)} = \Phi(\Omega^{(t)},\ \mathbf{p}) \tag{1}$$

The iterative application of Φ, starting from an initial configuration $\Omega^{(0)}$, leads to the CA simulation:

$$\Omega^{(0)} \xrightarrow{\Phi} \Omega^{(1)} \xrightarrow{\Phi} \cdots \xrightarrow{\Phi} \Omega^{(t)} \quad \Longrightarrow \quad \Omega^{(t)} = \Phi^t(\Omega^{(0)}, \mathbf{p}) \tag{2}$$

where the dependence of the automaton configuration at the time step t on both the initial configuration and the parameters is explicit, with the other automaton characteristics (i.e. the model structure) being fixed.

The CA model can be optimized with respect to $\mathbf{p}$ to maximise the agreement between the simulated patterns and those belonging to a spatio-temporal dataset $\bar{\mathcal{V}}$, which come from an experiment of the real system behaviour. In particular, let $\bar{\mathcal{V}}$ be composed by a sequence of q configurations:

$$\bar{\mathcal{V}} = \left\{ \bar{\Omega}^{(k)} : k \in \{0, \tau_1, \ldots, \tau_q\} \right\} \tag{3}$$

where $\tau_i \in \mathbb{N}$ indicates the time step in which a configuration is known. Starting from $\bar{\Omega}^{(0)}$, and given a vector $\mathbf{p}$ of parameters, the process (2) can be executed for the computation of the $q-1$ configurations:

$$\mathcal{V} = \left\{ \Omega^{(k)} : k \in \{\tau_1, \ldots, \tau_q\} \right\} \tag{4}$$

where $\Omega^{(j)} = \Phi^j(\bar{\Omega}^{(0)}, \mathbf{p})$. The agreement θ between the real and simulated processes is usually measured by a suitable fitness function:

$$\theta = \Theta\left(\bar{\mathcal{V}}, \mathcal{V}\right) = \Theta\left(\bar{\mathcal{V}}, \mathbf{p}\right) \tag{5}$$

Therefore, the calibration consists of the following maximisation problem:

$$\max_{\mathbf{p} \in \Lambda} \Theta\left(\bar{\mathcal{V}}, \mathcal{V}\right) \tag{6}$$

which involves finding a proper value of $\mathbf{p}$ that leads to the best agreement between the real and simulated spatio-temporal sequences.

Different heuristics have been used to tackle the automatic solution of problem (6) [2,3]. In this paper a MAMA has been adopted, which is designed according to some of the basic ideas described in [7]. The optimization process consists of a GA assisted by a fitness approximation model and endowed with a LS phase. The MAMA is used to evolve a population, whose generic *chromosome* is a n-dimensional vector $\mathbf{p}$. In the latter, the i-th element is obtained as the binary encoding of the parameter p_i. Each chromosome can be decoded back in a vector of parameters $\mathbf{p}$ and, through performing a CA simulation, the corresponding fitness can be computed.

3 A Meta-Model Assisted Memetic Algorithm

In the MAMA object of this paper the original fitness evaluations are partly replaced by the fitness estimates provided by an inexpensive model. This allows to reduce the number of CA simulations needed to evaluate the individuals generated by the genetic operators during the search. As detailed later, the CA simulations carried out during the optimization provide a training set $\mathcal{T}$:

$$\mathcal{T} = \left\{ \langle \theta^{(1)}, \mathbf{p}^{(1)} \rangle, \langle \theta^{(2)}, \mathbf{p}^{(2)} \rangle, \ldots, \langle \theta^{(n_t)}, \mathbf{p}^{(n_t)} \rangle \right\} \tag{7}$$

where each fitness value $\theta^{(i)}$ corresponds to a parameter vector $\mathbf{p}^{(i)}$. Thus, on the basis of the patterns in $\mathcal{T}$ a meta-model $\hat{\theta}$ is dynamically built for evaluating each candidate solution $\mathbf{p}$ through the estimated fitness value $\hat{\theta}(\mathbf{p})$.

It is worth noting that, given the patterns in $\mathcal{T}$, either a global meta-model or a local one can be trained [5,7]. For example, an *ad-hoc* surrogate of the real fitness can be constructed for each individual $\mathbf{p}$ to be evaluated using only the k nearest neighbours of $\mathbf{p}$ in $\mathcal{T}$. However, even if a local meta-model can potentially be more accurate than a global one, the cost of training a number of local surrogates should be compared with the cost of the true fitness evaluation.

Since the meta-model only provides a more or less accurate approximation of the fitness landscape, to avoid convergence to false optima the surrogate-assisted optimization should also use, in some way, the true fitness function

[5]. On the other hand, the involvement of the latter should be minimized due to its high computational cost. A trade-off is provided by a suitable *evolution control* strategy. In particular, the approach adopted in this paper is the so-called *individual-based control*, which consists of using the true fitness function to evaluate at each generation some of the offspring individuals (i.e. the *controlled individuals*). The latter are chosen according to the so-called *best strategy*, in which the exact fitness value is assigned to some of the individuals that are the best according to the meta-model. For a detailed discussion about the commonly adopted evolution control strategies, the reader is referred to [5].

The pseudo-code of the corresponding MAMA is outlined in Figure 1. As in other elitist GAs, the optimization procedure begins with the initialization and exact evaluation of a population of individuals encoding CA vectors of parameters. The evolutionary search is iterated until the assigned budget n_{sim} of CA evaluations is exhausted. During the search, each CA simulation leads to a new element for the archive $\mathcal{T}$ of the training patterns. Usually, in the first GA generations the set $\mathcal{T}$ does not contains enough elements to build a reliable meta-model. Thus, while the current number of elements in $\mathcal{T}$ is less than the threshold $\rho_1\, n_{\text{sim}}$ (see line 6), where $\rho_1 \in [0, 1]$, the search consists of a standard GA, in which the fitness evaluations are carried out through CA simulations (see line 8). Subsequently, when $|\mathcal{T}| \geq \rho_1\, n_{\text{sim}}$ the meta-model $\hat{\theta}$ is built/updated at each generation (see line 10) in order to estimate the fitness of each individual belonging to the set of offspring $\mathcal{S}$. The adopted global surrogate is a Radial

```
1  Q ← populationInit();
2  for each q in Q do
3      simulateCAAndUpdateArchive(q, T);
4  while ( |T| < n_sim )
5      S ← crossoverAndMutation(Q);
6      if ( |T| < ρ1 n_sim )
7          for each q ∈ S do
8              simulateCAsAndUpdateArchive(q, T);
9      else
10         θ̂ ← createRBFN(T);
11         for each q ∈ S do
12             surrogateFitnessEvaluation(q, θ̂);
13         κ ← π |S|;
14         controlTheBestAndUpdateArchive(S, κ, T);
15         if ( |T| > ρ2 n_sim )
16             for i = 0 to κ do
17                 S[i] ←localSearchAndUpdateArchive(S[i], T);
18         end if
19     end if
20     Q ← elitistSelection(Q, S);
21 end while
```

Fig. 1. Outline of the meta-model assisted memetic CA optimization. The variable n_{sim} indicates the assigned budget of CA evaluations for the optimization process.

Basis Function Network (RBFN), a special type of artificial neural network that uses radial basis functions as activation functions [10]. The RBFN is often used as surrogate to assist optimizations because of its good generalization ability and because of its simpler topology compared to other networks [5,7]. Formally, the adopted RBFN can be expressed as:

$$\theta(\mathbf{p}) = \sum_{i=1}^{n_h} w_i \delta(\mathbf{p} - \mathbf{c}_i) \tag{8}$$

where n_h is the number of hidden neurons, $\delta(\mathbf{x})$ is the kernel function, $\mathbf{c}_i$ is the i-th center and w_i are the weights. The adopted kernel function is the Gaussian:

$$\delta(\mathbf{p} - \mathbf{c}_i) = \exp\left(-\frac{\|\mathbf{p} - \mathbf{c}_i\|^2}{2\sigma_i^2}\right) \tag{9}$$

where σ_i is the bandwidth assumed for the centre $\mathbf{c}_i$. In particular, the RBFN implementation has been based on the SHARK C++ library, a machine learning framework for regression and classification tasks including neural networks and kernel methods [11]. The first stage for building $\hat{\theta}$ consists of a fully unsupervised learning in which the centres and the corresponding bandwidths are determined. In particular: (i) first the RBFN centres $\mathbf{c}_i$ are obtained by few iterations of a k-means clustering algorithm on the set $\mathcal{T}$; (ii) then the value of σ_i for a given cluster centre $\mathbf{c}_i$ is set to the average Euclidean distance between $\mathbf{c}_i$ and the training vectors which belong to that cluster. Subsequently, the weights w_i need to be trained to achieve good generalization. In this work, the weights of $\hat{\theta}$ are trained using the iRprop algorithm [12] implemented in the SHARK library, which is quite fast and efficient.

Once all the offspring are evaluated through $\hat{\theta}$, in order to avoid convergence towards false optima, the control strategy mentioned above is applied at line 14 by invoking the function *controlTheBestAndUpdateArchive*. In particular, the latter ensures that the fraction π of the individuals in $\mathcal{S}$ which are the best according to $\hat{\theta}$ are re-evaluated through CA simulations. As a further result of the function *controlTheBestAndUpdateArchive*, the first $\kappa = \pi\,|\mathcal{S}|$ offspring in $\mathcal{S}$ are sorted in descending order according to their fitness. Also, each CA simulation carried out during the control process contributes to the enrichment of the archive $\mathcal{T}$, which is used for future meta-model buildings/updates.

In an advanced stage of the optimization, in particular when $|\mathcal{T}| > \rho_2\, n_{\text{sim}}$, with $\rho_2 \in [0, 1]$ and $\rho_2 > \rho_1$, the κ controlled individuals are taken as starting point of the LS. The latter starts from each controlled individual $\mathbf{p}$ and is conducted on the local Quadratic Polynomial Approximation (QPA) defined as:

$$\theta(\mathbf{p}) = \beta_0 + \sum_{1 \le i \le n} \beta_i\, p_i + \sum_{1 \le i \le j \le n} \beta_{(n-1+i+j)}\, p_i\, p_j = \boldsymbol{\beta}^T\, \tilde{\mathbf{p}} \tag{10}$$

where:

$$\boldsymbol{\beta} = [\beta_0,\, \beta_1,\, \ldots,\, \beta_{n_v-1}]^T \tag{11}$$

is the vector collecting the $n_v = (n+1)(n+2)/2$ model coefficients and:

$$\tilde{\mathbf{p}} = [1, p_1, p_2, \ldots, p_1 p_2, \ldots, p_n^2]^T \tag{12}$$

is the vector of the CA parameters mapped into the polynomial model. In particular, in order to improve the LS reliability, an ad-hoc local QPA is built for each individual $\mathbf{p}$ on the basis of its $n_s \geq n_v$ nearest neighbours in $\mathcal{T}$. In this study, the model coefficients $\boldsymbol{\beta}$ are estimated using the least square method.

Even using an accurate QPA, the LS procedure may converge towards a point that does not represent an actual improvement of the starting individual $\mathbf{p}$. Hence, at the cost of some more CA simulations, the LS has been based on a trust-region approach [13]. In the latter, the LS iteratively operates on a region in which the accuracy of the QPA is verified by executing *ad-hoc* CA simulations. In particular, if the QPA accuracy is satisfying then the region is expanded; conversely, if the QPA accuracy is poor then the region is contracted. In practice, following the classical trust-region approach, the LS is structured in a sequence of subproblems as follows:

$$\begin{array}{ll} \max \hat{\psi}(\mathbf{p}^{(j)} + \mathbf{d}), & j = 0, 1, 2, \ldots, \lambda \\ \text{subject to } \|\mathbf{d}\| \leq r^{(j)} & \end{array} \tag{13}$$

where $\hat{\psi}(\mathbf{x})$ is the QPA meta-model, $\mathbf{p}^{(j)}$ is the starting point of the j-th iteration (i.e. $\mathbf{p}^{(0)}$ is the individual to optimize), $\mathbf{p}^{(j)} + \mathbf{d}$ represents a point within the current trust-region radius $r^{(j)}$. In this paper, the BLG code for solving an optimization problem with bound constraints through a gradient method, described in [14], is used for the trust-region subproblems. At the first sub-problem of the LS, the radius $r^{(0)}$ is initialized as the average of all the n_s nearest neighbours of $\mathbf{p}^{(0)}$ in $\mathcal{T}$. Then, the value of $r^{(j)}$ is determined for each of the following sub-problems on the basis of a parameter $\omega^{(j)}$, which is computed at the end of each subproblem as follows:

$$\omega^{(j)} = \frac{\theta(p^{(j)}) - \theta(p_{\text{opt}}^{(j)})}{\hat{\theta}(p^{(j)}) - \hat{\theta}(p_{\text{opt}}^{(j)})} \tag{14}$$

where each evaluation of the function $\theta(\mathbf{x})$ requires a CA simulation. Then, the trust region is contracted or expanded for low or high values of $\omega^{(j)}$ respectively, according to the empirical rule described in [7].

The LS process terminates when the maximum number of subproblems λ is reached. The latter parameter represents the individual learning intensity, that is the amount of computational budget in terms of CA simulations devoted on improving a single solution. At the end of each LS, any locally optimized vector of CA parameters is encoded back into the offspring according to a Lamarckian evolutionary approach [15].

4 Calibration Tests and Discussion

A Master-slaves parallel version of the MAMA described above has been developed and applied to the last release of SCIARA, a CA model for lava flows simulation. In the current implementation, based on the Message Passing paradigm, a

Table 1. Parameters object of calibration, explored ranges and target values

Parameter	Explored range for calibration	Target value
r_s	[0, 1]	0.096
r_v	[0, 1]	0.853
h_s $[m]$	[1, 50]	13.67
h_v $[m]$	[1, 50]	1.920
p_c	[0, 100]	8.460

master process executes the algorithm outlined in Figure 1, while the remaining processes carry out all the required CA simulations.

In the SCIARA model, which is described in detail in [9], a specific component of the transition function computes lava outflows from the central cell towards its neighbouring ones on the basis of the altitudes, lava thickness and temperatures in the neighbourhood. In the model, lava can flow out when its thickness overcomes a *critical height*, so that the basal stress exceeds the yield strength. The critical height mainly depends on the lava temperature according to a power law. Moreover, viscosity is accounted in terms of flow relaxation rate, being this latter the parameter of the distribution algorithm that influences the amount of lava that actually leaves the cell. At each time-step the new cell temperature is updated according to the mass and energy exchange between neighbouring cells and also by considering thermal energy loss due to lava surface irradiation. The temperature variation, besides the change of critical height, may lead to the lava solidification which, in turn, determines a change in the morphology. In SCIARA the transition function depends on the following scalar parameters: r_s, the relaxation rate at the temperature of solidification; r_v, the relaxation rate at the temperature of extrusion; h_s, the critical height at the temperature of solidification; h_v, the critical height at the temperature of extrusion; p_c, the "cooling parameter", which regulates the thermal energy loss due to lava surface irradiation. Once that the input to the model has been provided, such as parameter values, terrain topography, vents and the effusion rates as a function of time, SCIARA can simulate the lava flow. The simulation stops when the fluxes fall below a small threshold value. However, before using the model for predictive applications, the parameters must be optimized for a specific area and lava type. To this end, the following fitness measure was defined:

$$\theta = \frac{|R \cap S|}{|R \cup S|} \tag{15}$$

where R and S represent the areas affected by the real and simulated event, respectively. Note that $\theta \in [0,1]$; its value is 0 if the real and simulated events are completely disjoint, being $|R \cap S|=0$; it is 1 in case of perfect overlap, being $|R \cap S| = |R \cup S|$.

For the calibration task the MAMA was compared with the corresponding standard GA (SGA). In both algorithms a population of 100 bit-strings, each encoding a candidate solution $\mathbf{p} = [r_s, r_v, h_s, h_v, p_c]$, was evolved. In particular,

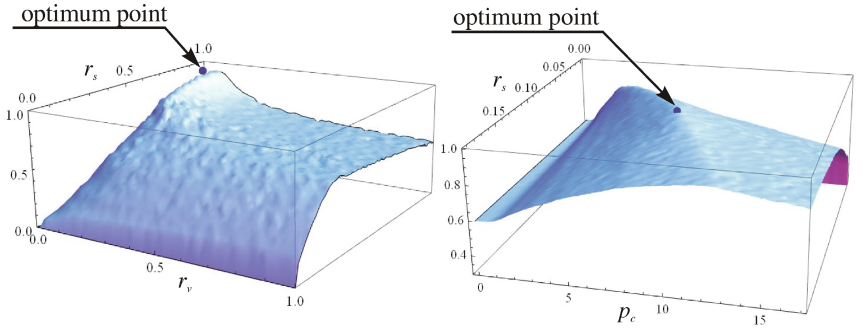

Fig. 2. The rugged fitness landscape generated by SCIARA

each of the SCIARA parameters was encoded on a string of 12 bits using the intervals shown in Table 1. As for the genetic operators, the standard 1-point crossover applied with probability $p_c = 1.0$ was adopted, while the mutation consisted of a bit flipping with probability $p_m = 1/n_b$, being n_b the number of bits per individual. Also, the standard Roulette Wheel Selection was applied together with an elitist replacement scheme.

The calibration exercise concerns a real event occurred on Mt. Etna (Sicily, Italy) in 2001 which is described in details in [2]. However, the target final configuration was obtained with SCIARA itself, using the set of parameters shown in Table 1. This guarantees the existence of a zero-error solution of the calibration problem, thus allowing for a more objective evaluation of the calibration procedures. In Figure 2 the landscape generated by the fitness defined in Equation (15) is depicted. In particular, the two surfaces were obtained executing a number of SCIARA simulations on a grid covering the whole search space, with a refinement in a neighbourhood of the target point shown in Table 1. The ruggedness of the fitness landscape, which can be observed in Figure 2, is known as one of the causes of slow convergence when using most optimization heuristics. In these cases, it is known that using global meta-models can help on smoothing the fitness landscape, thus speeding-up the optimization convergence. In the preliminary experiments presented here, besides the overall effectiveness of the

Table 2. Overview of the calibration results obtained assigning to each search algorithm a budget of 1000 SCIARA evaluations. The statistics were computed on 10 independent run of each algorithm.

	λ	Average	Min	Max	Std. Dev.
SGA	-	0.821	0.740	0.910	0.048
MAMA	0	0.918	0.901	0.950	0.015
	2	0.894	0.872	0.925	0.017
	4	0.910	0.862	0.966	0.035
	6	0.939	0.912	0.971	0.019
	10	0.901	0.860	0.921	0.019

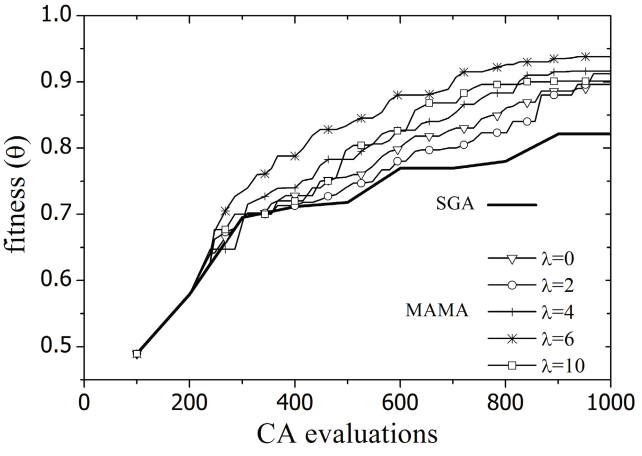

Fig. 3. Average behaviour of the optimization heuristics SGA and MAMA. In the latter, different learning intensities were tested.

MAMA, also the influence of the individual learning intensity (i.e., the parameter λ) was investigated. In particular, five different values of λ were considered, namely 0, 2, 4, 6 and 10. To all runs, a budget of $n_{\text{sim}} = 1000$ CA simulations was assigned. In the MAMA, the remaining parameters were $\rho_1 = 0.2$, $\rho_2 = 0.3$ and $\pi = 0.1$. Therefore, up to 200 CA simulations the MAMA worked as the SGA. Starting from 200 CA simulations, the MAMA operated exploiting the RBFN as fitness surrogate. Only after the first 300 CA simulations, the LS was applied to about 10 individuals per generation. For each type of heuristic search, 10 independent runs were carried out. In Table 2 an overview of the results is shown. Within the limited budget of 1000 CA evaluations, the SGA achieved an average fitness value of $\theta \approx 0.82$ and a maximum of $\theta \approx 0.91$. As expected, the MAMA outperformed the SGA providing the best result for $\lambda = 6$, that is a final average $\theta \approx 0.94$ and a maximum $\theta \approx 0.97$. Figure 3-*a* shows the average behaviour of the algorithms during the search process. Interestingly, for any number of SCIARA simulations and regardless of the learning intensity, the MAMA attained an average fitness significantly higher than that of the SGA. In particular, the MAMA with $\lambda = 6$ reached a significant average speed of convergence, by requiring only about one half of the computational budget to achieve the same fitness given by the SGA at the end of the process. Since each CA evaluation takes several minutes on a standard PC, the MAMA can thus provide the same results of a SGA saving a few hours of computation.

As can be seen on Table 2, in the present application the trade-off between exploration and exploitation regulated by λ had a limited influence (i.e. about 5% at most) on the achieved optimum. Probably, in this case the beneficial smoothing effects provided by the global meta-model plays a major role on speeding-up the optimization. However, it is important to remark that a small gain in the fitness defined by Equation (15) corresponds to a significant difference in the final map of the lava invasion.

5 Conclusions and Future Work

The preliminary results of this study indicate that the automatic optimization of CA models can greatly benefit by the use of a MAMA. Future work will focus on more sophisticated strategies for choosing the individuals on which it is worth investing CA simulations for a Lamarckian learning. In particular, an interesting direction to explore is that proposed in [7], where a pre-selection criterion based on a *probability of improvement* was adopted to rank the promising individuals.

References

1. Di Gregorio, S., Serra, R.: An empirical method for modelling and simulating some complex macroscopic phenomena by cellular automata. Future Generation Computer Systems 16, 259–271 (1999)
2. Rongo, R., Spataro, W., D'Ambrosio, D., Avolio, M.V., Trunfio, G.A., Gregorio, S.D.: Lava flow hazard evaluation through cellular automata and genetic algorithms: an application to mt etna volcano. Fundam. Inform. 87, 247–267 (2008)
3. Blecic, I., Cecchini, A., Trunfio, G.A.: A Comparison of Evolutionary Algorithms for Automatic Calibration of Constrained Cellular Automata. In: Taniar, D., Gervasi, O., Murgante, B., Pardede, E., Apduhan, B.O. (eds.) ICCSA 2010, Part I. LNCS, vol. 6016, pp. 166–181. Springer, Heidelberg (2010)
4. D'Ambrosio, D., Rongo, R., Spataro, W., Trunfio, G.A.: Meta-model Assisted Evolutionary Optimization of Cellular Automata: An Application to the SCIARA Model. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2011, Part II. LNCS, vol. 7204, pp. 533–542. Springer, Heidelberg (2012)
5. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Comput. 9, 3–12 (2005)
6. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms (1989)
7. Zhou, Z., Ong, Y.S., Nair, P.B., Keane, A.J., Lum, K.Y.: Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part C 37, 66–76 (2007)
8. Lim, D., Jin, Y., Ong, Y.S., Sendhoff, B.: Generalizing surrogate-assisted evolutionary computation. IEEE Trans. Evolutionary Computation 14, 329–355 (2010)
9. Spataro, W., Avolio, M.V., Lupiano, V., Trunfio, G.A., Rongo, R., D'Ambrosio, D.: The latest release of the lava flows simulation model sciara: First application to Mt Etna (Italy) and solution of the anisotropic flow direction problem on an ideal surface. Procedia CS 1, 17–26 (2010)
10. Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. Neural Comput. 3, 246–257 (1991)
11. Igel, C., Heidrich-Meisner, V., Glasmachers, T.: Shark. Journal of Machine Learning Research 9, 993–996 (2008)
12. Igel, C., Hüsken, M.: Empirical evaluation of the improved Rprop learning algorithm. Neurocomputing 50 (2003)
13. Celis, M.R., Dennis Jr., J.E., Tapia, R.A.: A trust region strategy for nonlinear equality constrained optimization. In: Boggs, P., Byrd, R., Schnabel, R. (eds.) Numerical Optimization 1984, pp. 71–82. SIAM Publications (1985)
14. Hager, W.W., Zhang, H.: A new active set algorithm for box constrained optimization. SIAM Journal on Optimization 17, 526–557 (2006)
15. Ong, Y., Keane, A.J.: Meta-lamarckian learning in memetic algorithms. IEEE Transactions on Evolutionary Computation 8, 99–110 (2004)