

Is a non-uniform system of creatures more efficient than a uniform one?

Patrick Ediger, Rolf Hoffmann and Mathias Halbach

Technische Universität Darmstadt
FB Informatik, FG Rechnerarchitektur
Hochschulstraße 10, 64289 Darmstadt, Germany
{ediger, hoffmann, halbach}@ra.informatik.tu-darmstadt.de

Abstract

We have analyzed the creatures exploration problem with non-uniform creatures. The creatures' task is to visit all empty cells in an environment containing obstacles with a minimum number of steps. Ten different algorithms with good performance from former investigations were used on 16 environments. New metrics were defined for such a multi agent system, like the absolute and relative efficiency. The efficiency relates the work of an agent system to the work of a reference system. A reference system is such a system that can solve the problem with the lowest number of creatures with potentially different algorithms. It turned out that the system CG-32a (32 creatures, algorithms C and G, alternating placed initially) is 31% respectively 33% more efficient than the reference systems. The relative efficiency was even higher when systems using the same algorithms were compared. Mixing different types of creatures resulted in efficiencies higher than one in 6% of all non-uniform systems.

1 Introduction

The general goal of our project is to optimize the cooperative behavior of moving creatures in order to fulfill a certain global task in an artificial environment. A creature (another term: agent) behaves according to an algorithm which is stored in the creature.

The goal of this investigation was to find out for the creatures' exploration problem (explained below), which algorithms "harmonize" best, meaning which combinations of algorithms with how many creatures are the most efficient. Different measures for efficiency were defined and used to compare the different combinations of creatures. When we are speaking about efficiency you may think of cost (e. g., Euros) which you have to pay in total for the involved creatures to fulfill the task.

We distinguish *uniform* and *non-uniform* systems of creatures. A uniform system comprises creatures of one type (behavior, algorithm) only whilst a non-uniform system comprises creatures of different types.

We are modeling the behavior by a finite state machine (Section 2). In the past we have tried to find out the best algorithm for one creature by enumeration. The number of state machines which can be coded using a state table is $M = (\#s\#y)^{(\#s\#x)}$ where $n = \#s$ is the number of states, $\#x$ is the number of different input states and $\#y$ is the number of different output actions. Note that M increases dramatically, especially with $\#s$, which makes it very difficult or even impossible to check the quality of all algorithms by enumeration in reasonable time.

By hardware support (FPGA technology) we were able to simulate and evaluate all 12^{12} 6-state algorithms (including algorithms with less than 6 states and including redundant ones) for a test set of 5 initial configurations [6]. The 10 best algorithms (with respect to percentage of visited cells) were used in further investigations to evaluate the robustness (using additional 21 environments) and the efficiency of $k > 1$ creatures. It turned out that more than one creature may solve the problem with less cost than a single one [5].

Modeling the behavior with a state machine with a restricted number of states and evaluation by enumerations was also undertaken in SOS [9]. Additional work was done by these authors using genetic algorithms.

In this investigation we have concentrated on non-uniform systems using the previously found algorithms. This time we are using 16 new environments compared to the environments used before. Now we use a field of fixed size 35×35 with a fixed number (129) of obstacles. Thereby we are able to place the creatures at the beginning in regular formations and the number of obstacles becomes a constant which simplifies the analysis.

We have already started experiments using metaheuristics to optimize the algorithms. We are optimistic to find

agent algorithms for more complex agent problems and we will use a cluster of FPGAs for that purpose to exploit the inherent parallelism in the metaheuristics. Our current results also give a partial answer to the question: If algorithms are combined, what are the expectation rates for good or bad combinations of them?

The creatures' exploration problem based on our model was further investigated in [2]. Randomness was added which led to a higher degree of success.

Our research in general is related to works like: Evolving optimal rules for cellular automata (CA) [10, 11], finding out the center of gravity by marching pixels [3, 4, 8], evolving hardware [12], using evolutionary algorithms and metaheuristics [1].

The remainder of this paper is organized as follows. Section 2 describes how the problem is modeled in the CA model. Section 3 describes how well only one type of creatures (uniform system) under varying the number of creatures and the environment, solves the problem. Section 4 (non-uniform system) describes how efficient two types of creatures can solve the problem.

2 CA model for the creatures' exploration problem

The problem is the following: p creatures are moving around in an environment that consists of empty cells and obstacle cells in order to visit all reachable empty cells in the shortest time. Creatures cannot move on obstacle cells, and only one creature can be on an empty cell at the same time. Creatures can look forward on the cell ahead which is in its moving direction. The creatures may perform four different actions:

- R (turn right only)
- L (turn left only)
- Rm (move forward and simultaneously turn right)
- Lm (move forward and simultaneously turn left)

If the "front cell" (the cell ahead) is not free, because it is an obstacle cell, another creature stands on it, or a collision conflict is anticipated, the action R or L is performed. In all other cases the action Lm or Rm is performed (Fig. 1).

The modeling of the behavior was done by implementing the rules with a state machine considered as a Mealy automaton with inputs (m, s) , next state s' and output d (Fig. 2). An algorithm is defined by the contents of a state table assigned to the state machine. We are coding an algorithm into a string representation or a simplified string representation by concatenating the contents line by line to a string or a corresponding number, e. g.,

$$1L2L0L4R5R3R-3Lm1Rm5Lm0Rm4Lm2Rm \\ = 1L2L0L4R5R3R-3L1R5L0R4L2R \text{ (short form).}$$

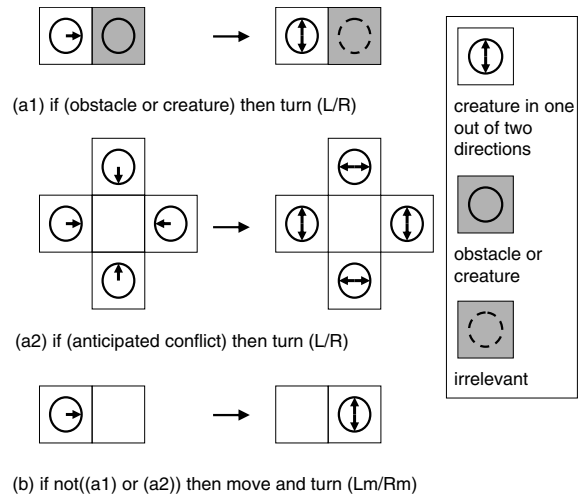


Figure 1. The conditions for the moving creatures' rules

The state table can be represented more clearly as a state graph (Fig. 2). If the state machine uses n states, we call such an algorithm n -state algorithm. If the automaton is considered as a Moore automaton instead of a Mealy automaton, the number of states will be the product $n \times \#r$, where $\#r$ is the number of possible directions (4 in our case).

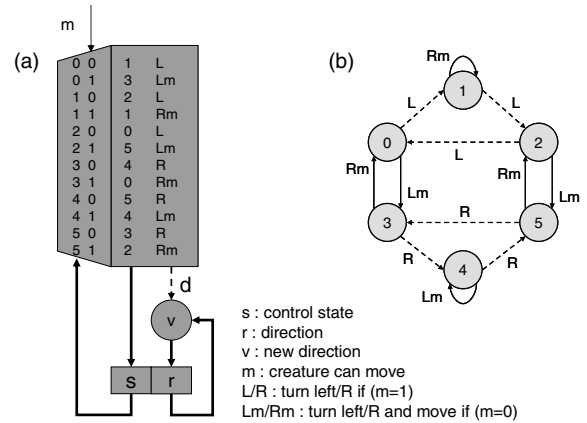


Figure 2. A state machine (a) models the behavior of a creature. Corresponding 6-state algorithm (b)

3 Uniform systems with one creature

In preceding investigations [6] we could discover and evaluate the best 6-state algorithms for one creature by the

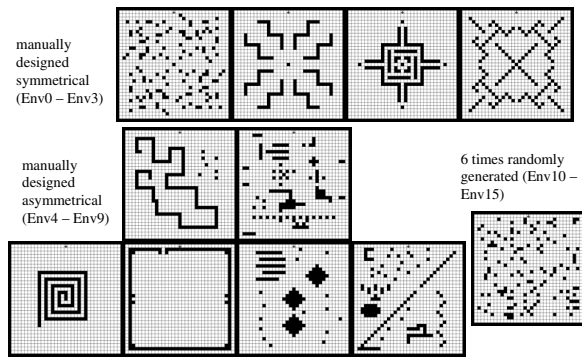


Figure 3. The 16 environments with 35x35 cells, manually designed or randomly generated. Each environment comprises $R = 960$ empty cells and 129 obstacles.

aid of special hardware. The behavior of all relevant algorithms was simulated and evaluated for 26 initial test configurations (they are different from the ones we are using here). The following 10 best algorithms with respect to (1.) success, (2.) coverage and (3.) speed are:

1. G: 1L2L0L4R5R3R-3L1R5L0R4L2R
2. B: 1R2R0R4L5L3L-3R1L5R0L4R2L
3. C: 1R2R0R4L5L3L-3R4R2L0L1L5R
4. A: 0R2R3R4L5L1L-1R5R4R0L2L3L
5. D: 1R2R3R1L5L1L-1R0L2L4R3L1L
6. E: 1R2L0R4L5L3L-3R4R5R0L1L2R
7. F: 1R2L0L4R5R3R-3L4L5L0R1L2R
8. H: 1L2L3R4L2R0L-2L4L0R3L5L4R
9. I: 1L2L3L4L2R0L-2L4L0R3R5L4R
10. J: 1R2R3R0R4L5L-4R5R3L2L0L1L

The following definitions and metrics are used:

- k := number of creatures
- R := number of empty cells
- g := generation (time steps)
- $r(g)$:= number of visited cells in generation g
- r_{\max} := the maximum number of cells which can be visited for $g \rightarrow \infty$
- g_{\max} := the first generation in which r_{\max} is achieved
- $e := r_{\max}/R[\%]$, the coverage or exploration rate, i. e. $\frac{\text{visited cells}}{\text{all empty cells}}$
- $speed := R/g_{\max}$ (only defined for successful algorithms)
- $step\ rate := \frac{1}{speed}$ (the number of cells visited in one generation)

In order to find an algorithm that is robust against changes of the environments, we have created a set of $I = 16$ environments which all contain 129 obstacle cells (Fig. 3). Then the algorithms A to J were simulated on them

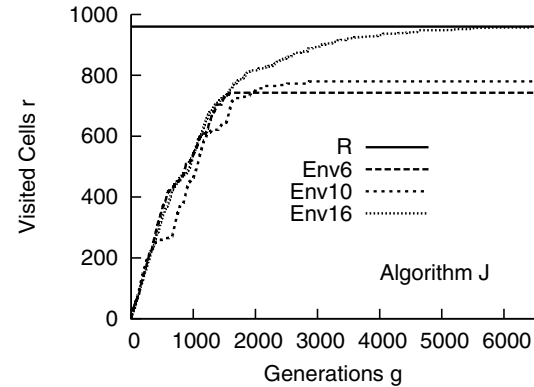
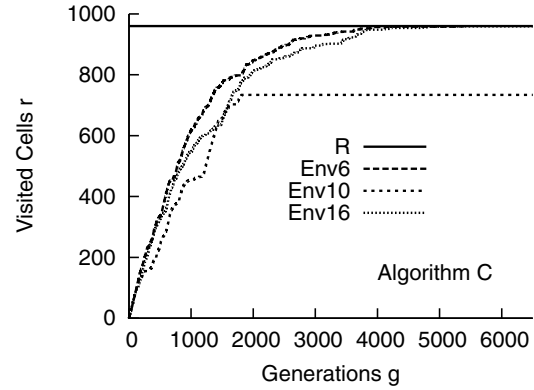
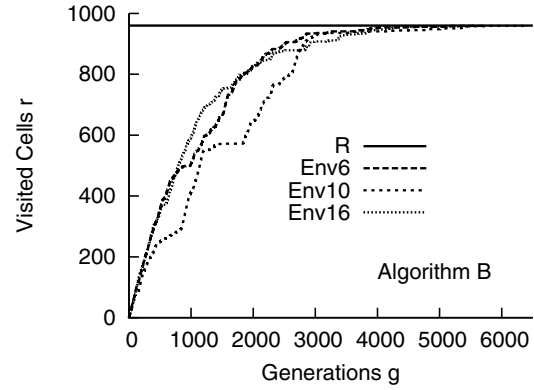


Figure 4. Curves showing the number of visited cells $r(g)$ (generations g on the x-axis, visited cells r on the y-axis) for the algorithms B, C and J on the environments Env6, Env10 and Env16. $R = 960$.

with one creature. The results show that none of the algorithms is capable of solving every environment successfully (Tab. 1). The best algorithms with respect to these environments are the algorithms B, G and J. Algorithm B has a *mean speed* (mean speed per $k = 1$ creature, see definition in section 4) of 16.12% for the successful environments. The highest reachable mean speed would be 100% (visiting one new empty cell in every step). The *mean step rate*, which is the reciprocal of the mean speed, is 5.31 (every 5.31 steps an empty cell is visited). We can interpret the mean step rate as work or cost which has to be paid for a creature to visit a cell (e. g., 5.31€ per empty cell to visit, or to “clean” one “square meter”, if you think of cleaning a room). Fig. 4 shows as example how algorithms may behave in principle. Algorithm B is successful for the environments 6, 10 and 16 and the speed is comparable (around 960 / 5400). Algorithm J is not successful for the environments 6 and 10, but we will see later in section 4, that J is effective in combination with other algorithms.

4 Multi creature and non-uniform systems

In [5, 7] we have evaluated that increasing the number of creatures can lead to synergy effects, i. e., the uniform creatures can work more efficiently together than by their own. 1 to 64 creatures were arranged on the cellular field symmetrically side by side at the borders (Fig. 5). The same distribution was used for the following investigation with two types of creatures.

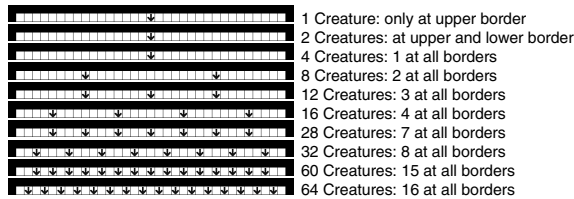


Figure 5. The arrangement of creatures in a multi creature system.

In order to investigate the performance of non-uniform systems, we have simulated all pairs (X, Y) of the former best single algorithms (A to J) on all 16 environments (Fig. 3) whereas the uniform pairs (X, X) are included for comparison. We used three different kinds of initial placement of the different types of creatures on the environment (Fig. 6):

- *alternating* (a), alternating clockwise, starting with the first algorithm in the left upper corner
- *cornerwise* (c), the first algorithm at the upper and right border, the second at the lower and left border

- *sidewise* (s), the first algorithm at the upper and lower border, the second at the left and right border

In our opinion the initial placement can have a significant effect on the performance, since the positive or negative effects of conflicts occur more or less frequently depending on the movements of creatures in the near surroundings. Regarding the distribution of creatures in Fig. 5 we exemplarily simulated every possible placement of 16 creatures (8 of each algorithm) with the algorithms E and A on environment $Env13$. The minimum value of generations needed to solve the environment is 472 and the maximum value is 4066, while 1098 is the average value. This shows that the initial placement can have a significant effect on the performance.

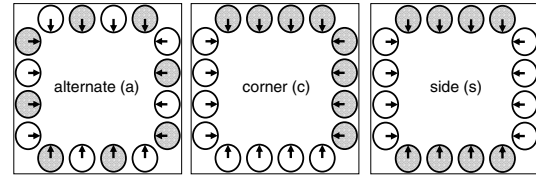


Figure 6. Types of placement of creatures in start configurations. Empty creatures symbolize creatures of the first algorithm, the shaded ones creatures of the second algorithm.

Two creatures. The results (Tab. 2) show that two creatures for the placement a and c are able to solve all 16 environments successfully only with the algorithm pairs (A, C) , (C, A) and (E, A) . The uniform pairs (X, X) are not successful. The placement s induces a uniform system (two opposite creatures with the same algorithm) which is not successful for all environments. The values in the diagonal are all less than 16.

Metrics for non-uniform systems with more than two creatures. In the following we will denote a system in the way $XY-kp$, where XY is the pair of algorithms, k the number of agents and p the type of placement (optional). E. g., AB-8a is a system with algorithm pair (A,B) comprising 8 creatures placed alternating.

To be able to evaluate and compare non-uniform systems (including uniform systems) with a different number of creatures we have defined additional metrics:

- *mean speed per creature* = $ms(k) = \frac{\sum_i r_{\max,i}}{k \cdot \sum_i g_{\max,i}}$. The speed $ms(k)$ is an average over all environments i and is related to one creature. This measure expresses how fast a creature can visit a cell on average (maximum is 100%). This measure should not be used if any environment can not be successfully visited because then

Algorithm	Success Env0	Success Env1	Success Env2	Success Env3	Success Env4	Success Env5	Success Env6	Success Env7	Success Env8	Success Env9	Success Env10	Success Env11	Success Env12	Success Env13	Success Env14	Success Env15	T =	No. Successful	total visiting percentage	mean g_max (successful)	mean r_max	mean speed (successful)	mean step rate
B	O	O	O	O	X	O	X	X	X	O	O	X	O	X	X	O	7	88.74%	5956	852	16.12%	5.31	
G	O	O	O	O	X	O	X	X	X	O	O	X	O	X	X	O	7	87.32%	5998	838	16.01%	5.10	
J	O	O	O	O	O	X	O	X	O	X	X	O	X	X	X	O	7	85.89%	6813	825	14.09%	5.78	
C	X	O	O	O	X	O	X	X	O	O	O	O	O	X	X	O	6	86.92%	5626	834	17.06%	4.86	
F	O	O	O	O	O	X	O	X	O	X	O	O	O	X	O	O	4	83.42%	6664	801	14.41%	5.20	
A	O	O	X	X	O	O	O	O	O	O	O	O	O	O	O	X	3	82.38%	14297	791	6.71%	8.32	
F	O	O	O	O	O	O	O	O	O	X	O	O	O	X	O	O	2	73.08%	6405	702	14.99%	4.60	
D	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	0	47.55%	-	456	-	2.96	
H	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	0	36.20%	-	348	-	4.69	
I	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	0	36.20%	-	348	-	4.75	

Table 1. Results of the simulation of one creature. The values are averaged over all environments, respectively of the T successfully visited environments in the case of “mean g_{max} ” and “mean speed”. An X in the columns “Success Env n ” indicates that the environment was successfully solved, an O that it was not.

		ALGORITHM 1										Σ
		A	B	C	D	E	F	G	H	I	J	
ALGORITHM 2	A	12	15	16	12	16	12	13	11	12	14	133
	B	14	11	13	10	15	13	12	11	9	13	121
	C	16	13	12	5	14	13	14	10	8	15	120
	D	14	8	7	0	7	7	7	1	2	7	60
	E	15	15	15	7	8	13	15	9	10	15	122
	F	13	14	14	6	12	9	13	6	9	15	111
	G	13	13	14	11	15	14	11	11	13	14	129
	H	13	13	11	1	7	6	11	1	2	12	77
	I	11	10	5	3	10	7	10	2	1	9	68
	J	15	14	15	10	15	14	13	11	10	12	129
Σ	136	126	122	65	119	108	119	73	76	126		

Table 2. The number of successfully solved environments by two agents with placement a for each pair of algorithms. The placement c would create the same table with rows and columns transposed. In the case of two creatures the placement s induces a uniform system whose results are the values in the diagonal of this table.

the mean speed might be believed higher than reasonable.

- *mean normalized work* = $mw(k) = \frac{k \cdot \sum g_{\max,i}}{\sum r_{\max,i}} = \frac{1}{ms(k)}$. This value represents the work which is necessary, or the costs one has to pay for one creature to visit a cell.
- *relative efficiency* = $\frac{ms(XY-k)}{ms(XY-k_{\min})}$. First a reference system $XY-k_{\min}$ has to be found which can solve the problem with the lowest number k_{\min} of creatures. The relative efficiency relates the mean speed of the system $XY-k$ to the mean speed of the reference system $XY-k_{\min}$. This measure compares the costs of the reference system $XY-k_{\min}$ with the cost of the system $XY-k$. If the relative efficiency is higher than one, the work can be done cheaper with the system $XY-k$.

Two similar measures can be defined if the uniform reference system $X-k_{\min}$ or the uniform reference system $Y-k_{\min}$ is chosen instead of $XY-k_{\min}$.

- *absolute efficiency* = $\frac{ms(XY-k)}{ms(UV-k_{\min})}$. In distinction to the relative efficiency another reference algorithm pair is used. The reference is the fastest of any algorithm pair UV (including the uniform “pairs” UU) which can solve the problem with a minimum number of creatures.

A similar measure can be defined if the fastest of any uniform reference systems $U-k_{\min}$ is chosen instead of $UV-k_{\min}$.

The efficiency measures are only defined if a k_{\min} exists and if the system $XY-k$ solves all 16 environments successfully. We have assumed for the reference algorithm an alternating initial placement in order to simplify the comparison.

The three successful systems XY-2. The comparison of the three best (most robust) systems for $k = 2$ shows that there is a significant interspace between their costs (work) (Tab. 3). The system EA-2 costs 22,50€ per visited cell compared to 14,80€ for the system CA-2.

Successful systems, depending on the number of creatures. Increasing the number of creatures the success rate also increases (Tab. 4). With 12 creatures, more than 81% (228/280) of the algorithm pairs can solve all environments. Even more than 97% can do it with 64 creatures. The most successful algorithms as a component of a non-uniform system are A, J, C, E, F, G, B, H, I, D in that order, averaged over placement (c, a, s), over the 16 environments and over the number of creatures (2, 4, 8, ... 64).

The fastest systems. Because of the high success rates of systems with many creatures we will discuss the effectiveness more than the robustness in the following evaluations. Therefore we will only take into account those systems that were successful on all environments.

Algorithm X	Algorithm Y	mean g_max	mean speed per creature	mean normalized work	absolute efficiency compared to J-8	absolute efficiency compared to CA-2
C	A	7105	6.76%	14.80 €	1.017	1.000
A	C	8291	5.79%	17.27 €	0.872	0.857
E	A	10800	4.44%	22.50 €	0.669	0.658

Table 3. The three pairs of algorithms that successfully solve all 16 environments with two creatures placed alternating.

The system BJ-64a is the most efficient, needing only 193 generations on average to solve the problem (Tab. 5). Except for one occurrence of the algorithms F in the system FC-64a (in rank 24), all pairs of the top 25 (only the top 10 are shown in Tab. 5) are a combination of the best 4 uniform algorithms for one creature (B, C, G and J).

No. of Agents (k)	Algorithm X	Algorithm Y	Placement	mean g_max	mean speed per creature	mean normalized work	absolute efficiency compared to J-8	absolute efficiency compared to CA-2
64	B	J	a	193	7.79%	12.84 €	1.173	1.153
60	C	J	c	203	7.89%	12.68 €	1.188	1.168
64	J	C	c	210	7.15%	13.98 €	1.077	1.059
64	B	J	c	211	7.13%	14.03 €	1.073	1.055
64	J	C	s	212	7.09%	14.10 €	1.068	1.050
64	C	J	c	213	7.05%	14.18 €	1.062	1.044
64	C	B	s	213	7.05%	14.19 €	1.061	1.043
64	G	J	s	214	7.02%	14.25 €	1.057	1.039
64	C	J	s	218	6.89%	14.51 €	1.038	1.020
60	B	C	c	220	7.26%	13.77 €	1.093	1.075

Table 5. The 10 absolute fastest systems (sorted ascending by “mean g_{\max} ”) that solve all 16 environments.

The most efficient systems and synergy. The system CG-32a is the most efficient (absolute efficiency) considering the costs per visited cell (Tab. 6). It is 30.9% more efficient than the reference system CA-2a and 33.2% more efficient than the reference system J-8. This answers the question posed in the title of the paper. We may interpret this effect as synergy. In total 6% (104 out of 1734) of the

NO. OF CREATURES	SUCCESSFUL (UNIFORM ONLY)	NO. OF SUCCESSFUL SYSTEMS / NO. OF COMBINATIONS NON-UNIFORM (INCL. UNIFORM)			
		a+c+s	alternate (a)	corner (c)	side (s)
1	-	0/10	0/10	0/10	0/10
2	-	3/100	3/100	3/100	0/10
4	-	46/190	19/100	27/100	19/100
8	J	142/280	22/100	71/100	51/100
12	B,C,J	228/280	76/100	85/100	73/100
16	B,C,E,G,J	237/280	65/100	91/100	91/100
28	B,C,D,E,I,J	269/280	93/100	94/100	94/100
32	C,D,E,F,G,H,I,J	265/280	91/100	95/100	95/100
60	B,C,D,E,F,G,H,I,J	272/280	97/100	97/100	96/100
64	B,C,D,E,F,G,H,I,J	272/280	97/100	97/100	96/100
total	41/100	1734/2260	563/910	660/910	615/820

Table 4. Percentage of algorithm pairs that solve successfully all 16 environments

No. of Agents (k)	Algorithm X	Algorithm Y	Placement	mean g_max	mean speed per creature	mean normalized work	mean conflicts (c) per creature	mean conflicts (b) per creature	mean conflicts (a) per creature	mean conflicts per creature	conflicts per visit	relative efficiency (uniform X)	relative efficiency (uniform Y)	absolute efficiency compared to J-8	relative efficiency (XY)	absolute efficiency compared to CA-2	k_min AlgX	k_min AlgY	k_min AlgX-AlgY
32	C	G	a	339	8.84%	11.31 €	10	6	50	66	2.19	1.721	1.451	1.332	1.290	1.309	12	16	12
28	J	G	s	415	8.26%	12.10 €	9	7	63	80	2.32	1.244	1.355	1.244	1.190	1.223	8	16	8
16	J	B	s	738	8.13%	12.30 €	8	6	114	128	2.14	1.224	1.949	1.224	1.154	1.203	8	12	4
32	J	G	a	371	8.09%	12.37 €	10	6	58	74	2.48	1.218	1.326	1.218	1.165	1.197	8	16	8
8	J	C	s	1486	8.08%	12.38 €	11	9	225	245	2.04	1.216	1.571	1.216	1.316	1.196	8	12	4
28	J	B	a	433	7.91%	12.64 €	9	7	65	81	2.36	1.191	1.897	1.191	1.124	1.171	8	12	4
28	C	J	s	434	7.90%	12.66 €	10	8	69	86	2.51	1.537	1.190	1.190	1.143	1.170	12	8	4
60	C	J	c	203	7.89%	12.68 €	8	9	32	49	3.07	1.535	1.188	1.188	1.142	1.168	12	8	4
16	B	J	c	761	7.89%	12.68 €	9	6	117	132	2.19	1.890	1.187	1.187	1.127	1.167	12	8	4
32	J	J	*	382	7.86%	12.72 €	8	7	60	74	2.47	1.184	1.184	1.184	1.184	1.164	8	8	8

Table 6. The top 10 most absolute efficient (lowest total costs) non-uniform systems. Constraint: all algorithm pairs are successful in all 16 environments (visiting percentage = 100%). "*" in placement means that all three types of distribution are identical in that case.

systems reach an absolute efficiency of more than 1. This result gives a hint what will happen when good agents are randomly combined, e. g., during heuristic methods.

With only one exception, the top 25 most efficient algorithm pairs (only the top 10 are shown in Tab. 6) are combinations of B, C, G and J, like for the top 25 fastest.

The table (Tab. 6) includes also values for the relative efficiencies which became even higher. Considering the top 25 absolute most efficient systems, the relative efficiencies are 1.89 for BJ-16c/B-12 and 1.949 for JB-16s/B-12 and 1.316 for JC-8s/JC-4a.

We have also counted the different types of conflicts (Tab. 6): (a) static obstacle, (b) conflict when a creature is on the front cell (dynamic obstacle), (c) anticipated conflicts (2, 3 or 4 creatures want to visit the same cell). At the moment it is not clear, whether there is a significant relation between the efficiency and the types and frequency of the conflicts.

5 Conclusion

The creatures' exploration problem was investigated in the CA model for multiple creatures using combinations of 10 algorithms (behaviors). These algorithms had shown a good performance in former investigations. The analysis was performed for 16 new environments of size 35×35 and 129 obstacles each.

New metrics have been defined for such multi creature systems, especially the *mean speed*, the *relative efficiency* (comparing the work of a system with an algorithmic similar system using the minimum number of creatures which can solve the problem), and the *absolute efficiency* (comparing the work of a system with an algorithmic potentially different system using the minimum number of creatures which can solve the problem).

A single creature is not successful for all environments. Two creatures are not successful in the uniform case, but three non-uniform systems are successful. The overall fastest system is BJ-64a but it is not the most efficient. The system CG-32a is 31% absolute more efficient than CA-2a and 33% more efficient than J-8. The highest relative efficiency reached was 1.95 for JC-8s compared to JC-4a.

Our future work is directed to investigate the relation between conflicts and efficiency, mixing algorithms in time instead of space, and to optimize the behavior through metaheuristics.

References

- [1] E. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley & Sons, NJ, USA, Aug. 2005.
- [2] B. N. Di Stefano and A. T. Lawniczak. Autonomous roving object's coverage of its universe. In *CCECE*, pages 1591–1594. IEEE, 2006.
- [3] D. Fey, M. Komann, F. Schurz, and A. Loos. An organic computing architecture for visual microprocessors based on marching pixels. In *ISCAS*, pages 2686–2689. IEEE, 2007.
- [4] D. Fey and D. Schmidt. Marching-pixels: a new organic computing paradigm for smart sensor processor arrays. In N. Bagherzadeh, M. Valero, and A. Ramírez, editors, *Conf. Computing Frontiers*, pages 1–9. ACM, 2005.
- [5] M. Halbach and R. Hoffmann. Solving the exploration's problem with several creatures more efficiently. In R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, editors, *EUROCAST*, volume 4739 of *Lecture Notes in Computer Science*, pages 596–603. Springer, 2007.
- [6] M. Halbach, R. Hoffmann, and L. Both. Optimal 6-state algorithms for the behavior of several moving creatures. In S. E. Yacoubi, B. Chopard, and S. Bandini, editors, *ACRI*, volume 4173 of *Lecture Notes in Computer Science*, pages 571–581. Springer, 2006.
- [7] R. Hoffmann and M. Halbach. Are several creatures more efficient than a single one? In S. E. Yacoubi, B. Chopard, and S. Bandini, editors, *ACRI*, volume 4173 of *Lecture Notes in Computer Science*, pages 707–711. Springer, 2006.
- [8] M. Komann, A. Mainka, and D. Fey. Comparison of evolving uniform, non-uniform cellular automaton, and genetic programming for centroid detection with hardware agents. In V. E. Malyshkin, editor, *PaCT*, volume 4671 of *Lecture Notes in Computer Science*, pages 432–441. Springer, 2007.
- [9] B. Mesot, E. Sanchez, C.-A. Peña, and A. Perez-Uribe. SOS++: Finding smart behaviors using learning and evolution. In R. Standish, M. Bedau, and H. Abbass, editors, *Artificial Life VIII: The 8th International Conference on Artificial Life*, pages 264–273, Cambridge, Massachusetts, 2002. MIT Press.
- [10] M. Sipper. *Evolution of Parallel Cellular Machines, The Cellular Programming Approach*, volume 1194 of *Lecture Notes in Computer Science*. Springer, 1997.
- [11] M. Sipper and M. Tomassini. Computation in artificially evolved, non-uniform cellular automata. *Theor. Comput. Sci.*, 217(1):81–98, 1999.
- [12] A. Upegui and E. Sanchez. On-chip and on-line self-reconfigurable adaptable platform: the non-uniform cellular automata case. In *IPDPS*. IEEE, 2006.