Evolving Multi-creature Systems for All-to-All Communication

Rolf Hoffmann and Patrick Ediger

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

Introduction. Several creatures are moving around in a cellular automata grid. At a certain point of time all creatures want to exchange their information with all others (all-to-all communication). The goal is to find an optimal rule for the movement of the creatures in order to exchange their information as fast as possible. The information exchange is only possible when the creatures meet each other and when they form certain defined local patterns (communication situations). Possible communication situations are exemplarily shown in Fig. 1. In the cases a, b, c the creatures are directly in contact. But it is a matter of definition whether such situations allow communication. For this investigation we have defined the communication patterns d, e, f. A reason could be that communication can only take place if a mediator/negotiator is used between them. Furthermore the mediator may perform a particular computation (e. g., average, maximum, priority select). Such conflicts occur when creatures want to move to the same target position, like vehicles which are meeting in a cross-way. The center of the crossing can be interpreted as the mediator.

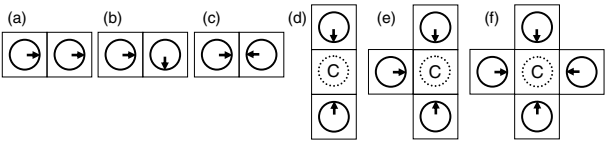

Fig. 1. Possible ommunication situations. Communication is only allowed for the situations d, e, f using a mediator C.

In former investigations [1] we have tried to find the best algorithms for the creature's exploration problem, in which the creatures have the task to visit all empty cells in shortest time. The presented problem is related to it in the way how creatures can move. But the task is different: The creatures shall exchange their information in shortest time taking advantage out of the conflicts which are the defined communication situations. – All-to-all communication is a very common task in distributed computing. The problem's specification can depend on many fixed or dynamic varying parameters like the number and location of nodes, the number and location of processes, the number, users and properties

H. Umeo et al. (Eds): ACRI 2008, LNCS 5191, pp. 550–554, 2008.

of the communication channels and so on. All-to-all communication in multi-creature systems is related to multi-agent problems like finding a consensus, synchronizing oscillators, flocking theory or rendezvous in space [2], or in general to distributed algorithms with robots [3].

Modeling. The whole system is modeled by cellular automata (CA). It consists of an environment ($n \times m$ grid) with borders or without borders (wrap-around) and k uniform creatures. A creature has a certain moving direction and it can only read the information from one cell ahead (target cell, front cell). If it detects a border cell or a creature in front or a conflict, it will turn right (R) or left (L). A conflict occurs when two or more creatures want to move to the same front cell (crossing point, cell in conflict, mediator). The conflict detection is realized by an arbitration logic [4] which is available in each cell. The arbitration logic evaluates the move requests coming from the creatures and replies asynchronously by an acknowledge signal in the same clock cycle. In the case that the creature can move forward it is simultaneously turning right (Rm) or to the left (Lm). Thus a creature performs the rule:

1. (*Evaluate move condition* x):
   If (front cell == OBSTACLE $\vee$ CREATURE $\vee$ CONFLICT) then $x = 0$ else $x = 1$
2. (*React*): If (x) then R/L else Rm/Lm

The decision which of the actions R/L respectively Rm/Lm will be performed depends on the behavior of the creature. The behavior (algorithm) of a creature is defined by a finite state machine. Input of the state machine is the move condition x. Output of the state machine is the signal y. The action R/L is performed if $y = 1/0$ and $x = 0$. The action Rm/Lm is performed if $y = 1/0$ and $x = 1$. The actions were defined in this way in order to keep the control automaton as simple as possible – A state machine is defined by a state transition table (Fig. 2) with current input x, current state s, next state s' and current output y. It is represented by concatenating the contents to a string, e. g.: 1R5L3L4R5R3L-1Lm2Rm3Rm4Lm5Rm0Lm or in a simplified form 1R5L3L4R5R3L-1L2R3R4L5R0L.

To solve the problem very general either theoretical or practical with respect to all interesting parameters is too difficult. Therefore we have specialized our investigation. The grid size was set to 35 by 35. This size was taken over from former investigations allowing to distribute equally a varying number of creatures

x	0						1					
s	0	1	2	3	4	5	0	1	2	3	4	5
s',y	1,R	5,L	3,L	4,R	5,R	3,L	1,Lm	2,Rm	3,Rm	4,Lm	5,Rm	0,Lm
i	0	1	2	3	4	5	6	7	8	9	10	11
j	0,1	2,3	4,5	6,7	8,9	10,11	12,13	14,15	16,17	18,19	20,21	22,23

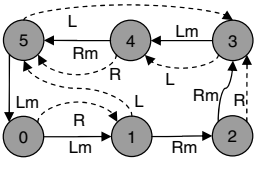

Fig. 2. A state table, defining the behavior (algorithm) of a creature and the corresponding state graph

at the borders. The number of creatures was set to $k = 16$. From former investigations in multi-creature systems we know that a number of creatures between approx. 8 and 64 can lead to good synergy effects and a sufficient number of conflicts which are required here. – The ultimate goal is to find the optimal behavior on average for all possible initial configurations. As we cannot test all possible configurations we will be satisfied if we can find the best behaviors for a first test set of 10 randomly generated initial configurations which have to confirm their quality using a second set of 100 random initial configurations. As the search space for different behaviors is very large we are not able to check all possible behaviors. Therefore we used a genetic procedure and tried to find the best behavior within a reasonable computational time limit. – The fitness of a multi-creature system is defined as the number of generations which are necessary to distribute (all-to-all) the information, averaged over all initial configurations (start positions and direction of the creatures) under test. In other words we search for state algorithms which can solve the problem with a minimum number of generations. – In order to model the distribution of information we are using a bit vector with k bits which is stored in each creature. At the beginning the bits are set mutually exclusive (bit(i)=1 for creature(i)). When two, three or four creatures form a communication situation they exchange their information by simply OR-ing their bit vectors together. The task is successfully solved when the bit vectors of all creatures obtain $11 \ldots 1$.

Genetic Procedure. The behavior is described by a state table (Fig. 2) using 6 control states, 2 inputs ($x = 0/1$) and 2 outputs ($y = 0/1$). The string representation of the state table defines the genome (individual, possible solution). P populations of N individuals are updated in each iteration (optimization cycle). During each cycle M offsprings are produced in each population. The union of the current N individuals and the M offsprings are sorted according to their fitness and the N best are selected to form the next population. An offspring is produced as follows: (1. Get Parents) Two parents are chosen for each population. Each parent is chosen from the own population with a probability of p_1 and from an arbitrary other population with the probability of $(1 - p_1)$. (2. Crossover) Each new component (s'_i, y_i) of the genome string is taken from either the first parent or the second parent with a probability of 50%. This means that the tuple (next state, output) for the position i=(input, state) is inherited from any parent. (3. Mutation) The string being modified by the crossover is afterwords mutated with a probabiltity of p_2. If a mutation shall be performed, an arbitrary position j is chosen and a new value (randomly choosen from the set of valid values) is replacing the existing one. Thereby either the next state or the output is changed at position i.

Results. The algorithms were optimized using $P = 7$ populations with $N = 100$ individuals each, $M = 10$ offsprings, $p_1 = 98\%$ and $p_2 = 9\%$. 800 iterations were performed starting with randomly generated state algorithms. The fitness was tested for each offspring by simulating the multi-creature systems with 10 initial random configurations. The 700 best behaviors which were produced after

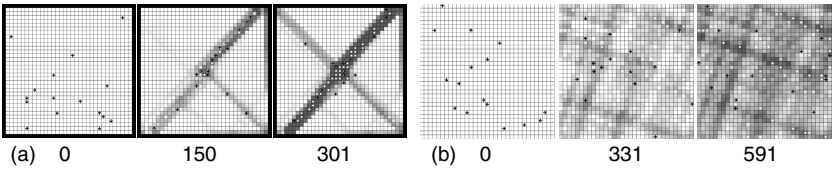

Fig. 3. Simulation patterns of the best algorithms A_b (a) respectively A_w (b) (see also Tab. 1) on an arbitrary start configuration. The numbers indicate the generations of the CA when the situations were observed. The rightmost images show the final generation when the algorithms accomplished all-to-all communication.

Table 1. The top 3 fastest alg. with and without borders averaged over 100 env

				not visited	Communication Situations		
		Algorithm	Generations	cells	d	e	f
with borders	A_b	2L5L5R1R3L4L-2R3R3L4R3L1R	391.76	29.43%	73.16	0.49	0.02
	B_b	2L5L5L1R3L4L-2R3R3L4R3L1R	392.41	29.39%	72.72	0.51	0.02
	C_b	2L5L3R1R3L2L-4R3R3L4R3L1R	394.06	29.48%	71.39	0.6	0.03
without borders	A_w	1R5L3L4R5R3L-1L2R3R4L5R0L	707.09	0.017%	82.68	0.35	0
	B_w	4L2R3L1R5R3L-1L2R3R4L5R0L	722.08	0.031%	88.03	0.43	0
	C_w	4R5L3L1R5R0L-1L2R3R4L5R0L	723.22	0.013%	83.58	0.34	0

800 iterations were used for an additional test set consisting of 100 initial configurations, also randomly generated in advance. – All algorithms which could successfully communicate in all the 100 environments show a similar global behavior: The creatures first move to a corner and from there on a diagonal path back and forth (Fig. 3(a)). Thereby two trails of communication are established, forming a cross. When a creature meets another creature it is slightly deviated, but manages to move back to a diagonal trail. The points of communication mainly lie on these trails and they are marked in Fig. 3 as communication spots in white. – We were not satisfied by this result when we realized that the communication trails were induced by the shape of the borders, in particular by the corners. Therefore we started another experiment: We removed the borders by wrap-around in order to get more interesting results to be independent of the border's shape. By the use of the same genetic procedure we could also evolve a bundle of good algorithms (the three best with borders and with wrap-around are shown in Tab. 1). Now the communication trails form a sort of grid, which is slightly turned to the right or to the left (Fig. 3(b)). Similar patterns have emerged for all the algorithms out of the top 100 we have checked. The best algorithm A_w is shown in Fig. 2. Analyzing the behavior of a single creature showed that it is following a trail which forms a spiral with wrap-around in the torus. – Future work: Further investigations are considered with different actions, a different number of control states, time-shuffled algorithms, non-uniform creatures [5], specialized fixed communication cells, and complete quality tests or formal proves for small environments.

References

1. Halbach, M., Hoffmann, R.: Optimal behavior of a moving creature in the cellular automata model. In: Malyshkin, V.E. (ed.) PaCT 2005. LNCS, vol. 3606, pp. 129–140. Springer, Heidelberg (2005)
2. Lin, J., Morse, A.S., Anderson, B.D.O.: The Multi-Agent Rendezvous Problem. An Extended Summary. In: Cooperative Control. LNCS, vol. 309, pp. 257–289. Springer, Heidelberg (2005)
3. Principe, G., Santoro, N.: Distributed Algorithms for Autonomous Mobile Robots. In: 4th IFIP Int. Conf. on TCS. IFIP, vol. 209, pp. 47–62. Springer, Heidelberg (2006)
4. Halbach, M., Hoffmann, R., Both, L.: Optimal 6-state algorithms for the behavior of several moving creatures. In: El Yacoubi, S., Chopard, B., Bandini, S. (eds.) ACRI 2006. LNCS, vol. 4173, pp. 571–581. Springer, Heidelberg (2006)
5. Ediger, P., Hoffmann, R., Halbach, M.: Is a non-uniform system of creatures more efficient than a uniform one? In: NIDISC at IPDPS Proceedings. IEEE, Los Alamitos (2008)