Routing Based on Evolved Agents

Patrick Ediger and Rolf Hoffmann
Motivation

- The goal of this investigation
  
  **Find an adaptive routing technique using multiple moving agents to transport messages in a regular 2D-grid.**

- Possible applications
  - New approach for designing networks on a chip
  - Model for transportation systems

- Questions in this work
  1) How can simple agents solve the routing task?
  2) How should a router based on agents be structured?
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

- **Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

**Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

- **Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

**Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

**Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

**Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

**Static Routing:** All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

**Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

**Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

- **Static Routing:** All agents (messages) are injected initially (barrier-synchronization).

![Diagram of 2D-Cellular Automaton with moving agents and target positions.](image)
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

**Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Problem Statement: Static Routing Task

- Given is a 2D-Cellular Automaton (CA) with moving agents.
- Each agent has the task to transport a message from a source position to a target position.
- Targets are also sources.
- Targets are assigned to sources mutually exclusive.

**Static Routing**: All agents (messages) are injected initially (barrier-synchronization).
Agents are directed: N, E, S, W

- front cell F reads and copies the agent
- current cell C deletes the agent from itself
Cellular Automata Model: Conflict Resolution

(1) agents can/must swap:

(2) each cell has a priority scheme:
Cellular Automata Model: Modeling Agent Behavior

- Agents react on inputs from the neighbor cells (adaptivity).
- Agents are controlled by finite state machines (FSM) with limited complexity.
- The output of the FSM activates an action, that is checked for conformity.
  - Turn **Right/Left/Back**, **Stay** (+ move ahead if possible)

![Diagram](attachment:image.png)
Cellular Automata Model: Inputs for the FSM

- Agent’s direction
- Agent’s assigned target cell (relative Cartesian coordinates)
- Reduced to 7 inputs, which indicate the sector of the target (relative to the direction of the agent).
Cellular Automata Model: Cell State

- **Cell type**
  - $\in \{\text{EMPTY, AGENT, OBSTACLE}\}$
  - Used only for borders

- **Direction**
  - $\in \{N, E, S, W\}$

- **Distance** = $(\Delta x, \Delta y)$
  - Relative distance to target, updated with movement

- **Control state**
  - behavior (FSM state)

- **Priority scheme**
  - used for conflict resolution
Evolving the Agents‘ Behavior

- The number of possible state transition tables for an FSM is

\[(#\text{states} \cdot #\text{outputs})^{(#\text{states} \cdot #\text{inputs})}\]

- Restriction to #states = 6 \(\rightarrow\) \(6 \cdot 4^6 \cdot 7 \approx 9 \cdot 10^{57}\) possible FSMs

- Searching for a good behavior with an Island Model Genetic Algorithm
  - Parameters: 5 islands with population size 100, immigration rate 2\%, mutation rate 0.9\%
  - 50 new FSMs per generation by uniform crossover with two parent FSMs: each entry in the transition table (a pair of next state and output) is taken from one of the parents.
1. A **Training Set** of few initial configurations of the grid was used to evolve FSMs.

2. After evolving the FSMs were ranked on a larger **Ranking Set** of initial configurations of the grid.
Evolving the Agents‘ Behavior: Fitness Function

- We want reliable (deadlock free) solutions
  - Capable of solving all initial configurations (placements of sources and permutations of assignment of sources and targets)
- We want fast solutions
  - Definition of “speed”: The inverse of the number of time steps needed to find all targets

The fitness function is a dominance relation:

1. # of resolved environments
2. # of found targets
3. Speed
Evolving Small Routers Separately

Best FSM was deadlock free for all possible permutations (3.9 time steps on average)

FSMs proven to be deadlock free, but slower (4.4 – 4.7 time steps).
Are free cells useful?

No deadlock/livelock free FSM found
Unavoidable Deadlock

- Deadlock cannot be prevented in some cases

- All agents perform the same action, because they all start in state 0 and all have the same inputs and control automaton.
Unavoidable Deadlock

Deadlock cannot be prevented in some cases

- All agents perform the same action, because they all start in state 0 and all have the same inputs and control automaton.
Unavoidable Deadlock

Deadlock cannot be prevented in some cases

- All agents perform the same action, because they all start in state 0 and all have the same inputs and control automaton.
Unavoidable Deadlock

Deadlock cannot be prevented in some cases

- All agents perform the same action, because they all start in state 0 and all have the same inputs and control automaton.
Large Router Structures

- Large cases L evolved for 16, 64, 144 and 256 agents
  - \( L_n-z \): \( n \) sources placed in a square of size \( \sqrt{n} \cdot \sqrt{n} \)
  - \( z \) free lines around the sources block
  - \( L_n-zR \): same grid size as \( L_n-z \), but sources placed randomly
Reliability of Evolved FSMs

- L16, L64 and L144 not reliable on the ranking set
  - Should be evolved with larger training set
- Reliability of the top 100 L256 for other numbers of agents
  - Many of the FSMs evolved for L256 are operational for the other cases
- In general, FSMs evolved for a large number of agents can be used for a smaller number of agents

<table>
<thead>
<tr>
<th>Tested on Ranking Set</th>
<th>Reliable FSMs out of Top100 L256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranking Set L256</td>
<td>28/100</td>
</tr>
<tr>
<td>Ranking Set L144</td>
<td>55/100</td>
</tr>
<tr>
<td>Ranking Set L64</td>
<td>63/100</td>
</tr>
<tr>
<td>Ranking Set L16</td>
<td>61/100</td>
</tr>
<tr>
<td>All Ranking Sets</td>
<td>4/100</td>
</tr>
</tbody>
</table>
Speed of the Agents

- Optimum = maximum of all shortest paths between an agent and its target
- General Observation: FSMs that are not always reliable are faster.
- The best FSMs are about three times slower than the optimum, example:

<table>
<thead>
<tr>
<th></th>
<th>Optimum</th>
<th>Steps needed by best evolved FSM L256 on average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranking Set L256</td>
<td>28.28</td>
<td>87.6</td>
</tr>
</tbody>
</table>
Optimal Amount of Free Cells (Randomly Distributed Agents)

- Increasing amount of free cells → increasing optimum, decreasing congestion → speed approaches optimum
- The FSMs work best with 1-2 free cells per agent

256 agents and 420 free cells, ratio 1.64 free cells per agent
Strategy of the Agents

steps: 0 15 30 45 60 75 90 105 120 135
Conclusion and Future Work

**Conclusion**
- A Multi-Agent System for Routing was modeled in CA.
- It’s possible to implement routers with FSM controlled agents.
- Deadlock free routing for some structures found.
- The amount of free cells plays an important role for “speed”
  - 1-2 free cells per agent are optimal.

**Future work**
- More complex agents.
- Other grid structures (hexavalent, 3D).
- Optimizing the Genetic Algorithm.
- Using hardware support (FPGAs).
Thank you for your attention!

image source: unknown
Appendix

The following slides were not used in the talk
Conflict resolution requires an extended neighborhood (Manhattan Distance 2).

Deleting by current cell C and copying by the front cell F must be consistent and thus based on the same information.
4 desired actions: R (right), L (left), B (back), S (stay)

Control automaton (FSM) → Check for conformity:

if (conflict) then $X#$, else $Xm$

conflict conditions

Always walk ahead, if possible

8 actions:
Rm, R#, Lm, L#, Bm, B#, Sm, S#

Always walk ahead, if possible

$Xm = \text{move ahead to the front cell and turn}$

$X# = \text{stay on the current cell and turn}$
Cellular Automata Model:

Cell Rule

If (cell type == EMPTY):
- Find neighboring AGENT with direction to “me” with highest priority.
- Copy agents’ control state, distance and direction and perform FSM transition.
- Update distance and direction, change cell type to AGENT.

If (cell type == AGENT):
- detect possibility of movement (obstacle, conflicts, swap, priority).
- If possible, change cell type to EMPTY
- If not, perform FSM transition and update direction and control state.

If (cell type == OBSTACLE):
- do nothing