# Position Identification of Moving Agents in Networks

**Anna Černá, Jan Černý**

Faculty of Mangement, University of Economics, Prague
Jindřichův Hradec, Czech Republic
cerna@fm.vse.cz, cerny@fm.vse.cz

*Abstract: The paper introduces a group of problems connected with the position identification of an agent moving through a given network and starting in an unknown vertex or edge (arrow). It presents a classification of the problems, describes the application fields and outlines further possible developments of the theory.*

*Keywords: Agent; position; identification; network*

## 1 Introduction

Agent-based models represent a topic of wide and deep studies during the past two decades. They have been used in the simulation of road traffic [6], in grid calculation [4], in human systems [2], and in coalitions [7], and one can find books dealing with them, see e.g. [15]. Agents can possess different “cleverness”, as one can see in the agent typology presented in a paper published in Acta Polytechnica Hungarica in 2006 [12]. Their learning capability is described in [18].

Mathematical models describing the motion of agents through networks are described in, for example, [8], [13], [20], although in [8] and [20], the authors use the notion “robot” instead of “agent”. In contrast to [20], we shall deal with other approaches to “navigation”. On the one hand, we shall suppose that the agent moves through a given network; on the other hand the agent is supposed to be able to deal with other observations. Our approach represents a generalization of the one from [5], [14], and [19]. They found methods leading the agent to the target vertex when the network fulfils some particular constraint of “synchronizing colouring”. In that case, the “cleverness” of the agent may be rather low. We weaken the first feature and strengthen the second, achieving both in several different ways.

We shall present several methods of solving problems of agent position identification and we shall outline their possible connection to problems which have been known now for decades.

The problems of agent position identification in networks can be encountered in several branches of polytechnics, for example:

- We can meet objects moving through **telecommunication networks** in ways similar to the agents in our models. We can mention, for example, tokens in token rings, packets in packet mode networks and so forth, and identification of their positions is a natural issue.
- A robot inspecting a **pipeline network** from inside is another good example of application.
- A speleologist exploring a **cave network** can be regarded as an agent moving through it. However, our theory has a sense only in the case when the explorer cannot use a Tarry or Tremaux algorithm based on marking the ends of incident edges in vertices.
- **Transportation networks** represent probably the most important application field of the problems connected with the identification of and agent's positions.

The last-but-not-least thing to be mentioned is the fact that the models and methods used in automata theory studying synchronizing input sequences are closely related to the ones used in agent position identification theory. The bridge connecting these two branches can be seen in the problem of road colouring.

## 2 Types of Graphs Used

We expect the reader is familiar with the terminology of graph theory. We shall only specify several notions and denotations here.

First we mention that the **vertex set** $V$ is always finite. If $v \in V$ and $w \in V$, then we form two kinds of pairs:

- **non-oriented pair** $(v, w)$ without any order of $v$ and $w$, i.e. $(v, w) = (w, v)$,
- **oriented pair** $vw$ where $v$ is the beginning and $w$ is the end, i.e. $vw \neq wv$ for $v \neq w$.

The set of all non-oriented pairs $(v, w)$ of vertices from $V$ is denoted $VV$, and its elements are called **edges**. They are graphically expressed by abscissae connecting the symbols of vertices $v$ and $w$, usually linear, exceptionally curvilinear.

The set of all oriented pairs of vertices $vw$ from $V$ is denoted $V \times V$ or $V^2$ and its elements are called **arrows**. They are graphically expressed by arrows starting in the symbol of the vertex $v$ and ending in the symbol of $w$, usually linear, exceptionally curvilinear.

Note that in either $(v, w) \in E$ or in $vw \in A$ we allow the possibility $v = w$. In such a case we speak about **loops**, non-oriented or oriented respectively. Similarly, if we do not specify otherwise, we allow that a pair of vertices $v$, $w$ is connected by more than one edge $(v, w)$ or arrow $vw$. In such a case, either we distinguish between them by an auxiliary criterion, e.g. a colour or a position ("first to the left"), or we take an arbitrary one.

If either $(v, w) \in E$ or $vw \in A$, then we say that $v$ and $w$ are **incident** with $(v, w)$ or $vw$, respectively.

Finally, we denote $2^V = \{W\colon W \subset V\}$ as the group of all subsets of $V$.

**2.1** We speak about the **graph** $G = (V, E)$ if $E \subset VV$. The set $E$ is said to be a **set of edges** of $G$. In other words, we speak briefly about a graph instead of "non-oriented graph".

**2.2** We speak about the **digraph** $D = (V, A)$ if $A \subset V^2$. The set $A$ is said to be a **set of arrows** of $D$. In other words, we speak briefly about a digraph instead of "oriented graph". Let $k > 1$. A vertex $v \in V$ is said to be $k$-**merging** if there exist $k$ different arrows $u_1x, \ldots, u_kx$.

**2.3** Let $\Gamma$ be a finite set of **colours**. The graph $G = (V, E)$ or the digraph $D = (V, A)$ is said to be **coloured** if each $e \in E$ or $a \in A$ is assigned a colour $\gamma(e) \in \Gamma$ or $\gamma(a) \in \Gamma$, respectively. The mapping $\gamma$ is said to be **colouring**. That is, speaking on colouring, we mean the colouring of edges or arrows. If we need to speak about the colouring of vertices, we will specify so. If we do not specify otherwise, we shall suppose that the colouring is a mapping **onto** the set $\Gamma$ i.e. for any colour $c \in \Gamma$ there exist $x \in E$ or $x \in A$ such that $c = \gamma(x)$.

Let $D = (V, A)$ be a digraph, $\gamma \in \Gamma$ and $k > 1$. A vertex $v \in V$ is said to be $k$-**merging** for $\gamma$ if there exist $k$ different arrows $u_1x, \ldots, u_kx$ with the colour $\gamma$.

**2.4** The digraph $D = (V, A)$ is said to be **derived** from the graph $G = (V, E)$ if exactly one of the relations $vw \in A$, $wv \in A$ holds for each $(v, w) \in E$. Moreover, if $G$ is coloured, then $D$ is coloured as well and $\gamma(vw) = \gamma((v, w))$ for each $vw \in A$. Note that $D$ arises from $G$ by the replacing of each edge by an arrow connecting the same vertices and it is of the same colour if $G$ was coloured.

**2.5** The digraph $DG = (V, AG)$ is said to be **equivalent** to the graph $G = (V, E)$ if $vw \in AG$ and $wv \in AG$ if and only if $(v, w) \in E$. Moreover, if $G$ is coloured, then $DG$ is coloured as well and $\gamma(vw) = \gamma(wv) = \gamma((v, w))$ for each $(v, w) \in E$. Note that $DG$ arises from $G$ by the replacing of all edges by pairs of counter arrows and they are of the same colour if $G$ was coloured.

**2.6** Let $D = (V, A)$ be a digraph and let $AD \subset VV \times VV$. Let $vw \in A$, $xy \in A$ if and only if $(v, x)(w, y) \in AD$. Then the digraph $DD = (VV, AD)$ is said to be the **double digraph** derived from $D$.

**2.7** Let $D = (V, A)$ be a digraph and let $AT \subset 2^V \times 2^V$. Let $U \subset V$, $W \subset V$. Let $UW \in AT$ if and only if $W = \{w \in V$: there exists $u \in U$ such that $uw \in A\}$. Then the digraph $DT = (2^V, AT)$ is said to be the **total digraph** derived from $D$.

**2.8** Let $D = (V, A)$ be a digraph, let $v \in V$ and let $AR \subset 2^{V-\{v\}} \times 2^{V-\{v\}}$. Let $U \subset V-\{v\}$, $W \subset V-\{v\}$. Let $UW \in AR$ if and only if $W = \{w \in V-\{v\}$: there exists $u \in U$ such that $uw \in A\}$. Then the digraph $DRv = (2^{V-\{v\}}, AR)$ is said to be the **reduced total digraph** derived from $D$ and $v$.

**2.9** Let $C$ be a finite set called **set of commands**. Let $D = (V, A)$ be a digraph and let $\delta : A \times C \to A$ be a mapping. We say that the arc $\delta(a, c)$ **follows** after the arc $a$ **due to the command** $c$. Let $A\delta = \{ab : a \in A, b \in A$ and there exists $c \in C$, $\delta(a, c) = b\}$. Let $D\delta C = (A, A\delta)$ be a coloured digraph such that $\delta(a, c) = b$ implies that the arrow $ab$ has the colour $c$. Then the digraph $D\delta C$ is called the **transition digraph** corresponding to the digraph $D$ and the set of commands $C$. We note that the vertices of $D\delta C$ are arrows of the former digraph $D$ and the arrows of $D\delta C$ express the choice of next arrow in $D$ due to the received command.

**Remark:** Commands from the set $C$ serve as colours of arrows in the graph $D\delta C$.

**2.10** A graph $G = (V, E)$ or a digraph $D = (V, A)$ is said to be **locally planar** if it is embedded in a smooth surface. This embedding determines cyclic ordering of edges or arrows incident with $v$ for each $v \in V$. Since different embeddings of the same graph or digraph may determine different cyclic orderings, say "Given locally planar graph/digraph", we shall mean that it is given together with its embedding in a given surface, i.e. together with the cyclic orderings.

We point out that the cyclic ordering is normally meant in a counter clockwise sense. It is similar to the ordering of adjacent roads in a roundabout (continental European, not British, where the motion is clockwise).

**2.11** Usually, for a motion of an agent through a graph or a digraph, it is important to note if there exists a path from $v$ to $w$ for each pair of vertices $v$, $w$. A graph $G = (V, E)$ having that property is said to be **connected**; a similar digraph $D = (V, A)$ is said to **strongly connected**. If in the digraph there exists a vertex $v$ such that there exists a path from any $w \in V$ to $v$, then the digraph is said to be **weakly connected.** If we do not specify otherwise, we shall automatically suppose that the agent moves through a network which is connected in the non-oriented case and weakly connected in the oriented one.

## 3 Moving Agents

**3.1** An **agent** is defined as a finite discrete system $\mathscr{S}$ composed from two finite automata, namely an executor $\mathscr{A}$ performing the moving and a controller $\mathscr{A}'$ giving commands to $\mathscr{A}$. That is, from the point of view of the classification of agents, presented e.g. in [12], the agents we shall deal with belong to the class of **hysteretic** agents. Of course, in the further development of the theory, a role of "less clever" reactive agents or "more clever" deliberative and, maybe mainly, attentive ones will be studied as well.

We can imagine that the agent is specified and constructed by some external client, after which the agent enters the network in an "unknown" position, and from that moment it moves step by step in the network autonomously until the controller orders it to stop (usually in a precisely identified position).

**3.2** The **executor** $\mathscr{A} = (Q, C, \delta)$ is without output. **Internal states** $q \in Q$ represent **positions** of the agent; **inputs** $c \in C$ represent **commands** coming from $\mathscr{A}'$; $\delta$ represents a **transition mapping**; that is, if the agent is in the position $q$ and $\mathscr{A}$ receives a command $c$, then it moves to the new position $\delta(q, c)$.

We must admit that the executor $\mathscr{A}$ can only be **partially defined**, or in other words. the next position $\delta(q, c)$ need not be defined for some position $q \in Q$ and a command $c \in C$. Then the agent remains in the position $q$ and waits for the next command. In other words, it behaves like $\delta(q, c) = q$ in that case like it was fully defined. This approach differs from the one of e.g. [16], where it is supposed that the input command $c$ cannot come in such a case.

We suppose that exactly one agent moves through the given graph or digraph and its movement, initiated by a sequence of commands $\mathbf{s} = c_1, \ldots, c_n$ , represents a discrete process $q_0, q_1, \ldots, q_n$, where $q_0$ is the initial position and for $i = 1, 2, \ldots, n$ we have $q_i = \delta(q_{i-1}, c_i)$. If the command is $c = \text{stop}$ then $\delta(q, c) = q$.

Obviously, our approach possesses an extension to the case of more than one moving agent, but we shall not study it now.

**3.3** The **transition mapping** $\delta$ can be **extended** and **modified** in several senses.

First, the previously-mentioned relations can be expressed also by the formulae:

$$q_n = \delta(q_0, \mathbf{s}) = \delta(q_0, c_1, \ldots, c_n) = \delta(\ldots\ \delta(\delta(q_0, c_1), c_2), \ldots, c_n)$$

$$q_0, q_1, \ldots, q_n = \delta^*(q_0, \mathbf{s}) = \delta^*(q_0, c_1, \ldots, c_n) =$$

$$= q_0, \delta(q_0, c_1), \ldots, \delta(q_0, c_1, \ldots, c_n)$$

The indexes of positions and commands need not be concordant; i.e. we can write

$$q_{m+n} = \delta(q_m, \mathbf{s}) = \delta(q_m, c_1, \ldots, c_n)$$

$$q_m, q_{m+1}, \ldots, q_n = \delta^*(q_m, \mathbf{s}) = \delta^*(q_m, c_1, \ldots, c_n)$$

**3.4** The **controller** $\mathscr{A}' = (Q', X, C, \delta', \lambda', q_0')$ is a finite state transducer (= "complete" initial finite automaton – see 5.1) having the output set $C$. Its input set $X$ represents the possible observations of some data in the current positions of the agent.

The simplest controller subsequently emits the "a priori" given sequence $c_1, \ldots, c_n$. It has the set of internal states $Q' = C$, the initial state $q_0' = c_1$, the input set $X = \varnothing$, and the output set $Y = C$, the transition mapping non depending on input: $\delta(c_i) = c_{i+1}$ for $i = 1, \ldots, n - 1$ and $\delta(c_n) = c_n$. The output mapping $\lambda$ is identical, i.e. $\lambda(c_i) = c_i$. This controller realizes the situation when the sequence of commands $\mathbf{s} = c_1, \ldots, c_n$ is known at the beginning and does not depend on the observations during the movement of the agent. Such a controller is called **trivial.**

More "smart" controller deduces the emitted commands as from observed current data $x$ as from the previous steps "reflected" in the state $q'$.

**3.5** Usually, if the agent moves in the digraph $D = (V, A)$ then we put $Q = V$, i.e. the positions are in vertices and arrows serve to transitions between vertices only. In this case we speak about "**vertex bound positions**".

On the other hand, if it moves in the graph $G = (V, E)$, then we model it in the equivalent digraph $DG = (V, AG)$ and we put $Q = AG$, i.e. the positions are in edges of $G$ in a given direction of motion, and the vertices serve to transition between edges only. We speak about "**edge bound positions**" even though we model them as "arrow bound".

Exceptionally, these choices can be interposed, that is, we can put $Q = V$ (vertex bound) for the agent moving in $G = (V, E)$ and $Q = A$ (arrow bound) for the agent moving in $D = (V, A)$. However, these possibilities will not be in the centre of our focus.

**3.6** The main problem we are focused on is the **identification of the agent's final state** by the controller. The controller manifests that fact via emitting the command "stop".

The simplest case is when it is "a priori" known that after emitting the sequence of commands $\mathbf{s} = c_1, \ldots, c_n$, the trivial controller will lead the executor to stop in the given position $q$ independent of the initial position $q_0$. Then we say that $\mathbf{s} = c_1, \ldots, c_n$ is the **synchronizing sequence of commands**.

In more complicated cases, the controller observes the data of the passed vertices and edges or arrows and, on the basis of this, it identifies the current position and eventually decides to emit the command "stop".

**3.7** Usually, there is given a **target** vertex or edge or arrow to the controller at the beginning. Then the controller compares the identifications with the target and in the case of their being equal emits "stop".

We have to note that this is exactly characterized if the target is of the same type as the position, e.g. both are vertices or both are arrows, etc. And, fortunately, this is the main case. The situation when the target is an arrow and positions are vertices is practically impossible. Thus, we must say what happens when the target is a vertex *v* but the agent's positions are in arrows, or in edges in given directions, which is nearly the same. Then the desired final positions of the agent are all of ending in the vertex *v*. For example, in the graph in Fig. 1, the target vertex *x* implies the desired final positions *vx*, *yx* and *zx*.

If the induced path passes or ends in the target for the given sequence $s = c_1, \ldots, c_n$ of commands and for any initial position, then we say that the sequence *s* is **target finding**.

An agent can **observe** some data characterizing its position.

The notion of observability in graphs was studied e.g. in [11]. Our point of view is the following:

**3.8** In the sequel we shall distinguish the following data, assumed to be available (i.e. "observable" by the agent) for each vertex $v \in V$:

D0 Beginnings of outgoing edges or arrows and their colours (if coloured). Moreover, if it is not the first position of the agent, then the end of the edge/arrow the agent comes in the vertex *v* belongs here as well.

D1 Degree $d(v)$ of the vertex *v* in the case of (non-oriented) graph.

D2 Number $d(v, c)$ of incident edges of the vertex *v* with the colour *c* in the case of (non-oriented) graph.

D3 Out-degree $d^+(v)$ of the vertex *v* in the case of digraph.

D4 Ends of ingoing arrows, their colours (if coloured) and in-degree $d^-(v)$ of the vertex *v*.

D5 Number $d^+(v, c)$ of outgoing arrows from the vertex *v* with the colour *c* in the case of digraph.

D6 Number $d^-(v, c)$ of ingoing arrows to the vertex *v* with the colour *c* in the case of digraph.

D7 Counter clockwise or clockwise cyclic ordering of incident edges or arrows with a vertex in the case of planar or locally planar graphs.

D8 Mark $\mu(v)$ of the target vertex *v*.

**3.9** As concerns edges or arrows, we shall distinguish the following data, assumed to be available for each edge $e \in E$ or each arrow $a \in A$:

D9 Colour $\gamma(e)$ or $\gamma(a)$ if it is coloured.

D10 Orientation (= the allowed direction of moving) of *a*.

D11 Mark $\mu(e)$ of the target edge *e* or $\mu(a)$ of the target arrow *a*.

**3.10** If positions of the agent are vertex bound then each position may be characterized by data selected from D1-D8. If they are edge bound then we add D9, D11 and for arrow bound we add D9, D10 and D11 to the previous ones. We say "add", not "replace", because the agent, moving through an edge or arrow, is able to observe data characterizing the final vertex before taking the decision concerning the next position.

Among the data D0-D10, we shall emphasise the ones that are needed in each case of agent motion we can meet. We call them **basic data**. If we deal with graphs and vertex bound agent positions, then D0, D1 and D2 belong to them; if it is edge bound then we add D9. If we deal with digraphs and vertex bound agent positions, then D0, D3 and D5 belong to them; if it is arrow bound then we add D9 and D10.

A bit beyond this frame are D4 and D6 for digraphs. If we add them to D0, D3 and D5, we speak about **extended basic data**. We note that in some applications it is not possible to observe that in a vertex there are ends of incoming arrows. Of course, extended basic data are observable, for example, in the major part of transportation networks.

On the other hand, in transportation networks and in some electric flex networks the assumption of local planarity is quite natural and then we can observe the data D7 there.

Finally, we can imagine many possible marks applied to vertices, edges or arrows. Their technical principles may vary very widely depending on the essence of graph or digraph in practice.

# 4 Problems Connected with the Identification of the Agent's Position

## 4.1 Main Problem

**4.1.1** As we see from the title of the paper, our main task is to formulate and solve various identification problems of the current position of agents moving in networks. However, in connection with this, some "auxiliary" problems arise, for example, how to choose the orientation of a given graph not yet oriented, how to choose the colouring of a given graph or digraph if not yet coloured, etc. Even though these problems are interesting as well, we shall start with the **general position identification problem** – in short, **GPIP**: Disregarding the (unknown at the beginning) initial position of the agent, to identify its current position after performing a given sequence of commands. This identification can be

- **full**, i.e. the current position is unambiguously determined as a concrete $q_n \in Q$,

- **partial**, i.e. a subset $Q_n \subset Q$ is found (as small as possible) such that the current position belongs to it.

**4.1.2 GPIP** can be divided into the following sub-problems:

**GPIP-1:** To find the partial or full identification of the initial position $q_0$ of the agent at the beginning of its movement.

**GPIP-2:** If the initial position is not fully identified, i.e. $q_0 \in Q_0$, $|Q_0| > 1$, to find such sequence $\mathbf{s} = c_1,\dots,c_n$ of input commands that the position $q_m \in \delta(\dots\delta(Q_0, c_1),\dots,c_m)$ is unambiguously identifiable for some $m \leq n$.

**GPIP-3:** Among the solutions $\mathbf{s}$ of **GPIP-2** to find the shortest one.

## 4.2 X/Y/Z Classification of Systems

**4.2.1** The solution of an agent position identification problem strongly depends on the type of system, consisting of the agent and the network (i.e. graph or digraph). We shall classify them using three symbols *X/Y/Z* expressing the following features:

**X)** **type of network and position of agent**: Da – digraph, position in arrows (= "arrow bound"), Dv – digraph, position in vertices, Ne/Nv – graph (non-oriented), position in edges/vertices respectively,

**Y)** possibility of **additional observations** beyond basic data (see 3.8-3.10): B – basic data only, E – extended basic data in digraphs, L – locally planar graph/digraph with cyclic ordering of edges in vertices, Tv – target vertex marking, Te – target edge marking,

**Z)** occurrence of **edge colouring**: C – coloured, N – non-coloured.

For instance, the class Ne/Tv/C contains full or partial identification of marked target vertex in non-oriented coloured graphs with positions of the agent in edges.

**4.2.2** Dv/B/C represents a deeply explored and well known case of synchronizing colouring [5], [14], [19]. It concerns digraphs with proper colouring, i.e. no outgoing arrows from the same vertex have equal colours. The command "c" determines the colour of the arrow to the next position, hence $C = \Gamma$. The trivial controller subsequently emits the "a priori" given sequence $\mathbf{s} = c_1, \dots, c_n$. The **sequence** $\mathbf{s}$ is called **synchronizing** if $|\delta(V, \mathbf{s})| = 1$. If there exists a synchronizing sequence then the **colouring** is called **synchronizing** as well.

These notions are important mainly for the case when each vertex has the same outdegree $|\Gamma|$ i.e. for each vertex and each colour there exist exactly one outgoing arrow having that colour.

**4.2.3** We admit more than one symbol in the second position, e.g. Dv/BTv/C means that both basic data and target vertex mark are observable.

### 4.3 Auxiliary Problems

It may happen that the given network is modified before the agent starts its motion in it. In our case, the goal of such modifications is to facilitate the consecutive identification of moving agent's position.

We need to emphasise that the problems mentioned henceforth will be fully formulated and able to be solved only if the *X/Y/Z* type of system is specified.

**4.3.1** One of the auxiliary problems is the following: Given a locally planar connected graph $G = (V, E)$. To find a digraph $D = (V, A)$ derived from $G$ such that it is synchronizing with respect to the commands of the type "$c$" where $c$ is a positive integer and the command means "when you come to the vertex $w$ by the arrow $vw$, then continue through the $c^{th}$ outgoing arrow, counting in clockwise sense from $q$".

**4.3.2** **Another** auxiliary problem starts with a digraph $D = (V, A)$ and a set of colours $\Gamma$. Then one has to find a synchronizing colouring of $D$ in the system Dv/B/C. This is the well known **road colouring problem** – see e.g. [1], [19] studied mainly in the case of a constant outdegree of vertices in $D$.

## 5 Applications and Connection with Other Theories

### 5.1 Finite Automata

**5.1.1** As we remember from the "classic" publications, a **finite automaton** (of the Mealy type) is defined as a quintuple $\mathcal{A} = (Q, X, Y, \delta, \lambda)$ where $Q$ is a finite set of internal states, $X$ is a finite set of inputs, $Y$ is a finite set of outputs, further $\delta$: $Q \times X \to Q$ is a transition mapping determining the next state, i.e. $q_{n+1} = \delta(q_n, x_{n+1})$, $\lambda$: $Q \times X \to Y$ is an output mapping determining the next output, i.e. $y_{n+1} = \lambda(q_n, x_{n+1})$. We note that in more recent papers this notion is often named the "**finite state transducer**".

**5.1.2** The automaton $\mathcal{A}$ can be represented by a "coloured" digraph $D = (Q, A)$ with the "colour" set $\Gamma = X$ and auxiliary marking with the mark set $Y$. The arrow $pq \in A$ with the colour $\gamma(pq) = x$ and the mark $\mu(pq) = y$ if and only if $q = \delta(p, x)$ and $y = \lambda(p, x)$.

We see that the colouring of the digraph $D$ is proper and the outdegree of each vertex equals to the number of colours $|X|$.

Note that $D$ may be a multigraph; there may exist more than one arrow connecting two adjacent vertices $p$, $q$. Of course, they are distinguished by their colours.

**5.1.3** We can say that the question of final state identification is almost as old as the finite automata theory itself. As a "black box", the automaton does not allow for the observation and to identification of the internal state directly. In late 50's, e.g. in [9], there were studied so called **experiments**, i.e. observations of output sequence $\boldsymbol{o} = y_1, \ldots, y_n$ corresponding to a given input sequence $\boldsymbol{s} = c_1, \ldots, c_n$ in order to identify the final state. Saying by the "moving agent through network" language, the agent receives the command sequence $\boldsymbol{s} = c_1, \ldots, c_n$, passes through a path observing the marks $\boldsymbol{o} = y_1, \ldots, y_n$ and identifies the final vertex. If there is a target final vertex then we say the sequence $\boldsymbol{s} = c_1, \ldots, c_n$ is target finding.

**5.1.4** A decade later, in [3], the notion of synchronizing sequence $\boldsymbol{s} = c_1, \ldots, c_n$ was introduced using the name "directing word". There $|\delta(Q, \boldsymbol{s})| = 1$, i.e. the final state is unambiguously determined without any observation. That means, the automaton is thought as a triplet $\mathscr{A} = (Q, X, \delta)$, the corresponding coloured digraph has no marks and the synchronizing property enables identification of the final state anyway. The minimal length of the synchronizing sequence $\boldsymbol{s}$ was also studied and estimated, as seen e.g. in [10] and [17].

## 5.2 Transportation

The authors are convinced that **transportation networks** represent the most important application field of the problems connected with identification of an agent's position in networks. Of course, the moving agents are persons in this case. Primarily, we shall consider road networks, bicycle path networks and pedestrian or hiking path networks. One can be sure that these studies have practical sense in spite of GPS devises for several reasons. Somewhere or sometimes there may be problems with signal, some area is not depicted well in GPS and mainly, many agents (drivers, cyclists, hikers) do not possess GPS.

Dealing with diverse transportation networks we can meet many variants of our main problem described in 4.1 and not less auxiliary problems presented in 4.3. The systems we shall deal with may belong to many classes in our triple X/Y/Z classification.

**5.2.1 Example:** Let $G = (V, E)$ be the graph from Fig. 1 representing a very simple network of cycle paths. Let us suppose that the moving agent's name is Clark. If it is a system of the type Nv/B/N, even though Clark can distinguish the vertices in according to the degrees, $d(v) = d(z) = 2$, $d(x) = d(y) = 3$, he has **no chance to identify the current state;** neither will he follow the simple rule represented by a sequence of commands $\boldsymbol{s} = c_1, \ldots, c_n$ nor a more complex rule represented by a controller $\mathscr{A}' = (Q', X, C, \delta', \lambda', q_0')$. The vertices $v$ and $z$ are undistinguishable observing basic data only. The same remains true even in the cases Nv/BL/N and Ne/BL/N as one can verify easily.

**5.2.2 Example continuation:** On the other hand, the systems Ne/BLTv/N and Ne/BLTe/N enable to find “target finding” sequence $\boldsymbol{s} = c_1, \ldots, c_n$.

In the graph $G = (V, E)$ from Fig. 1 we have $Q = \{vx, vy, xv, xy, xz, yv, yx, yz, zx, zy\}$. Since the graph is locally planar the command set may be $C = \{\text{stop}, 1, 2, 3\}$ where $m = 1, 2, 3$ means: “take the $m^{\text{th}}$ exit anticlockwise. If the marked vertex is $x$ then the target $x$ finding sequence of commands is 1, 1, 1:

- The initial positions *vx, yx* and *zx* have the observable marked target vertex *x* immediately **at the end of the edge**.
- The initial positions *vy, yv* and *yz* have the observable marked target vertex *x* **at the end of the next position after applying the command** “1”.
- The initial positions *xv, xy* and *zy* have the observable marked target vertex *x* **at the end of the reached position after applying the subsequence** “1,1”, e.g. $xy \rightarrow yz \rightarrow zx$.
- The initial position *xz* has the observable marked target vertex *x* **at the end of the reached position after applying the sequence** “1, 1, 1”: $xz \rightarrow zy \rightarrow yv \rightarrow vx$.

We have just shown that regardless of the initial position, applying the sequence “1, 1, 1” certainly enables the agent to pass through the target vertex *x*.

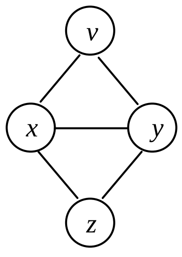

Figure 1
Original graph

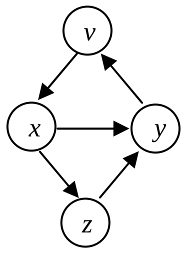

Figure 2
Derived digraph

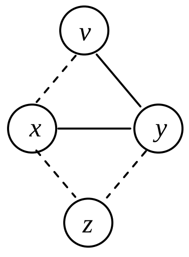

Figure 3
Coloured graph

If the marked target position is *vx* then the the target finding sequence of commands is 1, 1, 1, 2, 1, 1, 1, 2, 1, 1 as can be easily seen.

**Real world interpretation.** Let us suppose that the graph from Fig. 1 represents a cycle path network in a forest. Clark knows that his friend Xanthia is waiting for him sitting at a vertex (junction) *x*. He comes to the forest from outside and finds a path not knowing where he is and which direction to go in. Following the command sequence “1, 1, 1” he is sure that he will find Xanthia at the latest at the third passed vertex.

**5.2.3 Example.** We can ask whether the given graph $G = (V, E)$ may be replaced by a digraph $D = (V, A)$ derived from the graph $G$ (see 2.4) in such a manner that $D$ is synchronizing as a locally planar graph. That is, we consider the agent moving in $D$ as a system Da/BL/N.

For the graph $G$ from Figure 1 the answer is positive, the derived digraph $D$ is in Figure 2. It is easy to see that the sequence $\boldsymbol{s}$ = 2, 1, 2, 2, 2, 1, 2, 2, 1, stop synchronizes the digraph $D$ into the position $q = vx$. E.g. $xz$ 2→ $zy$ 1→ $yv$ 2→ $vx$ 2→ $xy$ 2→ $yv$ 1→ $vx$ 2→ $xy$ 2→ $yv$ 1→ $vx$ stop.

**Real world interpretation.** Let us suppose that the supervisor of the cycle network is considering converting all paths to a one-way type. Existence of a synchronizing sequence of commands may be a welcome feature.

That was "a priori" a given sequence $\boldsymbol{s}$ applied by a trivial controller. Let us consider a non-trivial controller $\mathscr{A}' = (Q', X', C, \delta', \lambda', q_0')$ where $Q' = \{p, r\}$, $X' = \{1, 2\}$, $C = \{1, \text{stop}\}$, $\delta'(p, 1) = p$, $\delta'(p, 2) = \delta'(r, 1) = \delta'(r, 2) = r$, $\lambda'(p,1) = 1$, $\lambda'(p, 2) = \lambda'(r, 1) = \lambda'(r, 2) = \text{stop}$, $q_0 = p$. When the agent's position is $uw$, then the current input $x'$ of the controller is equal to the value of outdegree $d^+(w)$. Hence, the final position $vx$ is reached by the agent in at most 3 steps; e.g. $q_0 = xy$, $q_0' = p$, $x_1' = d^+(y) = 1$, $q_1' = \delta'(p, 1) = p$, $c_1 = \lambda'(p,1) = 1$, $q_1 = \delta(xy, 1) = yv$, $x_2' = d^+(v) = 1$, $q_2' = \delta'(p, 1) = p$, $c_2 = \lambda'(p,1) = 1$, $q_2 = \delta(yv, 1) = vx$, $x_3' = d^+(x) = 2$, $q_3' = \delta'(p, 2) = r$, $c_3 = \lambda'(p,2) = \text{stop}$, $q_3 = \delta(vx, \text{stop}) = vx$ (repeated, i.e. the agent stops there).

We note that the vertex $y$ is 2-merging (see 2.2). Obviously, existence of such a vertex is a necessary condition for the synchronizing property.

**5.2.4 Example.** We can put another question then in 5.2.3. We shall ask whether the given graph $G = (V, E)$ may be coloured in such a way that it becomes synchronizing as a locally planar graph. That is, we consider the agent moving in coloured $G$ as a system Ne/BL/C.

**Real world interpretation.** Let us suppose that the supervisor of the cycle network is considering "colouring" the paths in the network, distributing coloured spur stones. Existence of synchronizing sequence is welcome as well.

For the graph $G$ from Fig. 1 the answer is positive, the coloured graph $G$ is in Fig. 3. Dashed lines represent the colour "b" (blue), solid lines the colour "r" (red). The possible commands we shall use are the following: 1r (first red to the right, i.e. in a counter clockwise direction), 2r, 1b (first blue to the right) and 2b. The set of all positions is $Q = \{vx, vy, xv, xy, xz, yv, yx, yz, zx, zy\}$. It is easy to prove that $\delta(q_0, \boldsymbol{s}) = xy$ for the sequence $\boldsymbol{s}$ = 1r, 1b, 1r, 1b, 1r, 1r, 1b, 1r and for each $q_0 \in Q$. E.g. $q_0 = xz$ 1r→ $xz$ 1b→ $zy$ 1r→$yv$1b→ $vx$ 1r→ $xy$ 1r→ $yv$ 1b→ $vx$ 1r→ $xy$.

We can note that having the desired final vertex $x$, the sufficient synchronizing sequence is $\boldsymbol{s}$ without the last "1r", i.e. 1r, 1b, 1r, 1b, 1r, 1r, 1b.

Similarly as in example 5.2.3, it is possible to look for some non-trivial controller here as well. Its construction can result from the fact that all vertices are distinguishable by their cyclic ordering of the colours of incident edges: $v \sim$ b-r, $x \sim$ b-b-r, $y \sim$ b-r-r, $z \sim$ r-r. Hence we can put $X = \{v, x, y, z\}$ as observable input data for the controller $\mathscr{A}' = (Q', X, C, \delta', \lambda', q_0')$. The other components of $\mathscr{A}'$ can be defined in many ways, e.g. $Q' = \{q_t, q_u, q_v, q_x, q_y, q_z\}$, $C = \{1\text{r}, 1\text{b}, \text{stop}\}$, $q_0' = q_u$. The transition and output mappings are defined only partially; the other "values" are irrelevant: $\delta'(q_t, y) = q_t$, $\lambda'(q_t, y) = \text{stop}$, $\delta'(q_u, v) = q_v$, $\lambda'(q_u, v) = 1\text{b}$, $\delta'(q_u, x) = q_x$, $\lambda'(q_u, x) = 1\text{r}$, $\delta'(q_u, y) = q_y$, $\lambda'(q_u, y) = 1\text{b}$, $\delta'(q_u, z) = q_u$, $\lambda'(q_u, z) = 1\text{b}$, $\delta'(q_v, x) = q_x$, $\lambda'(q_v, x) = 1\text{r}$, $\delta'(q_x, y) = q_t$, $\lambda'(q_x, y) = \text{stop}$, $\delta'(q_y, z) = q_z$, $\lambda'(q_y, z) = 1\text{b}$, $\delta'(q_z, x) = q_x$, $\lambda'(q_z, x) = 1\text{r}$. The digraph representing the partially defined controller is in Fig. 4. We shall show how the agent works if the starting position (= initial state of $\mathscr{A}$) is $q_0 = xz$:

1° $\mathscr{A}'$, starting in $q_u$, receives input $z$, remains in $q_u$ and emits the command 1b. $\mathscr{A}$ receives it and transits to $zy$.

2° $\mathscr{A}'$ receives input $y$, transits to $q_y$ and emits 1b. $\mathscr{A}$ receives it and transits to $yz$.

3° $\mathscr{A}'$ receives input $z$, transits to $q_z$ and emits 1b. $\mathscr{A}$ receives it and transits to $zx$.
4° $\mathscr{A}'$ receives input $x$, transits to $q_x$ and emits 1r. $\mathscr{A}$ receives it and transits to $xy$.
5° $\mathscr{A}'$ receives input $y$, transits to $q_t$ and emits stop. $\mathscr{A}$ receives it and stops in $xy$. $\mathscr{A}'$ receives input $y$ remains in $q_t$ and emits stop etc. The agent stops in the position $xy$ "for ever".

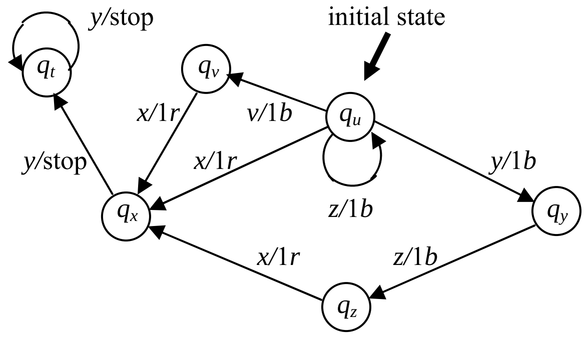

Figure 4
Finite transducer $\mathscr{A}'$

**5.2.5 Example – continuation of 5.2.3.** We have seen that the digraph in Fig. 2 possesses a synchronizing sequence if it is considered as a system Da/BL/N, i.e. when the agent's positions are arrow bound and each vertex has a cyclic ordering of outgoing arrows. Now we can ask whether there exists a colouring of arrows such that the coloured digraph becomes synchronized even in the system Dv/B/C. Yes, it exists (see Fig. 5); the synchronizing colouring exists with vertex bound positions without any cyclic ordering in vertices. Of course, the set of commands $C = \{b, r\}$, "*b*" means blue, in Fig. 5 dashed lines, "*r*" means red, in Fig. 5 solid. The synchronizing sequence is ***s*** = *r*, *b*, *b*, *r*. We must not forget that if the position is for example z, the command is *b* and from there no outgoing blue arrow exists, the agent remains in the same position. Hence his behaviour is identical as if there was a blue loop as in Fig. 6, where the loops were added and the digraph has constant outdegree 2 in each vertex. Colouring of such digraphs is the purpose of "road colouring" as described e.g. in [1] or [19].

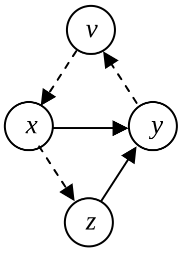

Figure 5
Synchronizing colouring

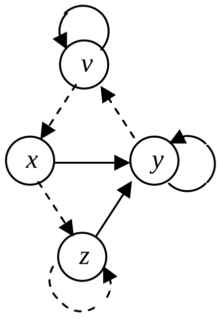

Figure 6
Constant outdegree

It is easy to see that the colouring from Fig. 3 is also synchronizing if applied to Fig. 2. The synchronizing sequence is very simple – *b*, *b*, *b*.

## 6 Target Reach in Systems Ne/BLTe/N and Ne/BLTv/N

As we have mentioned in 4.2.2, the case of oriented and coloured Dv/B/C systems have been deeply explored by many authors.

Now we turn our focus to the yet "untouched" types Ne/BLTe/N and Ne/BLTv/N. On one hand, they need not be coloured and oriented, but on the other hand, they must fulfil stronger constraints of local planarity and marked targets.

We shall present two theorems. The first one proves the existence of the desired sequence of commands leading to the target. The second one shows how to minimize the length of such a sequence.

## 6.1 Existence of Target Reaching Sequence of Commands

**6.1.1 Theorem:** Let $G = (V, E)$ be a finite connected locally planar graph with at least two vertices and with a marked edge $(u, v)$. Let $DG = (V, AG)$ be the equivalent digraph with a marked arrow $uv$. Let $C = \{1, \ldots, d_{\max}\}$ be the set of commands, where $d_{\max} = \max\{d(w): w \in V\}$. Let us have $\delta(xy, c) = yz$ where $yz$ is the $c^{\text{th}}$ outgoing arrow from the vertex $y$ for each $c \in C$ and $q = xy \in AG$. Then there exists a sequence of commands $\boldsymbol{s} = c_1, \ldots, c_n$ such that for each $q_0 = xy \in Q = AG$ the sequence (see 3.3) $\delta^*(xy, \boldsymbol{s})$ starts in $uv$ or ends in $uv$ or passes through $uv$.

**Proof:** First, we introduce the denotation $\boldsymbol{sr} = c_1, \ldots, c_n, c_{n+1}, \ldots, c_{n+m}$ for $\boldsymbol{s} = c_1, \ldots, c_n$, $\boldsymbol{r} = c_{n+1}, \ldots, c_{n+m}$ as the concatenation of sequences $\boldsymbol{s}$ and $\boldsymbol{r}$.

Since $G$ is connected, the digraph $DG$ is strongly connected and therefore there exists a path in $DG$ from $xy$ to $uv$ for each $xy \in AG$ and therefore there exists a sequence of commands $\boldsymbol{s} = c_1, \ldots, c_n$ such that $\delta(xy, \boldsymbol{s}) = uv$.

1° Let us denote $Q_1 = AG - \{uv\}$. If $Q_1 = \emptyset$ we have finished, even for the empty sequence $\Lambda$ we have $q \in AG \Rightarrow \delta(q, \Lambda) = uv$. If $Q_1 \neq \emptyset$ then we choose an element $q_1 = x_1y_1 \in Q_1$. Let $\boldsymbol{s}_1$ be the sequence of commands with the property $\delta(x_1y_1, \boldsymbol{s}_1) = uv$.

2° Let us denote $Q_2 = AG - \{uv, x_1y_1\}$. If $Q_2 = \emptyset$ we have finished since $\delta^*(q, \boldsymbol{s}_1)$ starts or ends in $uv$ for all $q \in AG$. If $Q_2 \neq \emptyset$ then we choose an element $q_2 = x_2y_2 \in Q_2$. Let $\boldsymbol{s}_2$ be the sequence of commands with the property $\delta(\delta(x_2y_2, \boldsymbol{s}_1), \boldsymbol{s}_2) = \delta(x_2y_2, \boldsymbol{s}_1\boldsymbol{s}_2) = uv$.

3° Let us denote $Q_3 = AG - \{uv, x_1y_1, x_2y_2\}$. If $Q_3 = \emptyset$ we have finished since $\delta^*(q, \boldsymbol{s}_1\boldsymbol{s}_2)$ starts or ends in $uv$ or passes through $uv$ for all $q \in AG$. If $Q_3 \neq \emptyset$ then we choose an element $q_3 = x_3y_3 \in Q_2$. Let $\boldsymbol{s}_3$ be the sequence of commands with the property $\delta(\delta(\delta(x_3y_3, \boldsymbol{s}_1), \boldsymbol{s}_2), \boldsymbol{s}_3) = \delta(x_3y_3, \boldsymbol{s}_1\boldsymbol{s}_2\boldsymbol{s}_3) = uv$.

… etc. By induction we find a sequence $\boldsymbol{s} = \boldsymbol{s}_1\boldsymbol{s}_2\ldots\boldsymbol{s}_m$ having the desired property.

**6.1.2** If we have a marked target vertex $v$ instead of target arrow $uv$, because of connectivity of $G$ there certainly exists an arrow $uv \in AG$ and we can use the sequence $\boldsymbol{s}$ from the theorem 6.2.1.

## 6.2 Minimization of Target Reaching Sequence of Commands

**6.2.1 Theorem:** Let $G = (V, E)$, $DG = (V, AG)$, $uv$ and $C$ be defined as in 6.1.1. Let $D\delta C = (AG, AG\delta)$ be the **transition digraph** corresponding to the digraph $DG$ and the set of commands $C$ (see 2.9) and let $D\delta CRuv = (2^{AG-\{uv\}}, AR)$ be the **reduced total digraph** derived from $D\delta C$ and $uv$ (see 2.8). Let $\boldsymbol{p}$ be the shortest path from the vertex $AG-\{uv\}$ to the vertex $\emptyset$ in the graph $D\delta CRuv$ and let $\boldsymbol{s} = c_1, \ldots, c_n$ be the sequence of colours of the arrows generating the path $\boldsymbol{p}$. Then $\boldsymbol{s}$ is the shortest sequence of commands such that for each $q_0 = xy \in Q = AG$ the sequence $\delta^*(xy, \boldsymbol{s})$ starts in $uv$ or ends in $uv$ or passes through $uv$.

**Proof:** 1) Let $\boldsymbol{s} = c_1, \dots, c_n$ be such a sequence of commands that for each $q_0 = xy \in Q = AG$ the sequence $\delta^*(xy, \boldsymbol{s})$ starts in $uv$ or ends in $uv$ or passes through $uv$. Since

$$\delta^*(q_0, c_1, \dots, c_n) = q_0, \delta(q_0, c_1), \delta(\delta(q_0, c_1), c_2), \dots, \delta(\dots(\delta(\delta(q_0, c_1), c_2), \dots, c_n)$$

the sequence $\delta^*(xy, \boldsymbol{s})$ represents a path containing the vertex $uv$ in the coloured transition digraph $D\delta C$.

Let the arrows of a path $\boldsymbol{r}$ starting in the vertex $AG-\{\mathrm{uv}\}$ in the reduced total digraph $D\delta CRuv$ have the colours $c_1, \dots, c_n$. Then the next vertices of the path are

$$\delta( AG-\{\mathrm{uv}\}, c_1) = \{\delta(xy, c_1)\text{: } xy \in AG-\{\mathrm{uv}\}\}$$

$$\delta( AG-\{\mathrm{uv}\}, c_1, c_2) = \{\delta(xy, c_1)\text{: } xy \in AG-\{\mathrm{uv}\}\}$$

… etc.

and at least one of them (and all after it, if exist) must be $\varnothing$, because for each $xy \in AG-\{\mathrm{uv}\}$ the sequence$\delta^*(xy, c_1, \dots, c_n)$ contains $uv$. Hence $\boldsymbol{r}$ is a path from the vertex $AG-\{\mathrm{uv}\}$ to the vertex $\varnothing$in the digraph $D\delta CRuv$.

2) Conversely, if $\boldsymbol{r}$ is a path from the vertex $AG-\{\mathrm{uv}\}$ to the vertex $\varnothing$in the graph $D\delta CRuv$ and $c_1, \dots, c_n$ are the colours of its arrows, then the sequence $\delta^*(xy, c_1, \dots, c_n)$ must contain $uv$.

This concludes the proof; the shortest path $\boldsymbol{p}$ represents the shortest sequence of colours = commands $\boldsymbol{s} = c_1, \dots, c_n$ of the desired properties.

**6.2.2 Example.** Let us return to the example 5.2.1 (the graph from Fig. 1), to the second part, when the marked target position is $vx$. The target finding sequence of commands is 1, 1, 1, 2, 1, 1, 1, 2, 1, 1 (the stop at the end is not important). In the digraph $D\delta CRuv$ we have

$AG-\{\mathrm{vx}\} = \{vy, xv, xy, xz, yv, yx, yz, zx, zy\}$ $1\rightarrow$ $\{vy, xv, xy, yv, yx, yz, zx, zy\}$ $1\rightarrow$

$1\rightarrow$ $\{vy, xv, xy, yv, yx, yz, zx\}$ $1\rightarrow$ $\{vy, xv, xy, yx, yz, zx\}$ $2\rightarrow$ $\{xv, xz, yv, yz, zy\}$ $1\rightarrow$

$1\rightarrow\{vy, yv, zx, zy\}$ $1\rightarrow\{xy, yv, yx\}$ $1\rightarrow\{xv,yz\}$ $1\rightarrow\{zy\}$ $1\rightarrow\{yv\}$ $1\rightarrow \varnothing$

**Real world interpretation.** The supervisor of the network can use theorem 6.2.1 for two purposes:

- ascertaining that the network posses a synchronizing sequence
- finding the shortest one.

**6.2.3** Theorem 6.2.1 solves the problem of marked target finding in the case Ne/BLTe/N when the target is in "arrow" *uv*. As concerns the system Ne/BLTv/N with a marked target vertex *v*, it is enough to replace the set $AG - \{uv\}$ by the set $AG - \{xv: xv \in AG\}$ in the reduced total digraph.

## Conclusion, Further Perspectives

We have shown that the position identification of agents moving in networks possesses several important features:

- It represents a specific and interesting issue within the theory of agent based systems.
- It has interesting practical applications. Among these, mainly the transport ones are promising.
- It demonstrates interesting links between three branches of discrete systems – finite transducers, coloured graphs and transportation.

Moreover, the authors hope that this paper can stimulate further studies oriented mainly to the following directions:

**D1 Generalizations,** namely

- towards higher cleverness of agents,
- concerning random environment - e.g. the colouring of roads perhaps partially damaged, etc.,
- extending the number of agents moving in the same network.

**D2 New methods and algorithms** solving concrete versions of the main problem or of the auxiliary one.

## References

[1] Adler, R., Goodwin, I., Weiss, B.: Equivalence of Topological Markov Shifts, in Israel. J. Math 27 (1977), pp. 49-63

[2] Bonabeau, E.: Agent-based Modeling: Methods and Techniques for Simulating Human Systems. In Proc.National Academy of Sciences 99 (2001) No. 3, pp. 7280-7287

[3] Černý, J.: Poznámka k. homogénnym experimentom s konečnými automatmi, Mat. fyz. čas SAV 14 (1964), pp. 208-215 (in Slovak)

[4] Cicortas, A., Iordan, V.: A Multi-Agent Framework for Execution of Complex Applications. Acta Polytechnica Hungarica 3 (2006), No. 3, pp. 97-119

[5] Culik, K. II, Karhumäki, J., Kari, J. A.: Note on Synchronized Automata and Road Coloring Problem, in: Lecture Notes in Computer Science 2295 (2002), pp. 175-185

[6] Erol, K.: A Study of Agent-based Traffic Simulation", Final Report, FHWA, US DOT, 1998

[7] Frankovič, B., Dang,, Tung-T., Budinská, I.: Agents' Coalitions Based on a Dynamic Programming Approach. Acta Polytechnica Hungarica 4 (2007), No. 2, pp. 5-21

[8] Fukuda, T., Hosokai, H., Otsuka, M.: Path Planning and Control of Pipeline Inspection and Maintenance Robot, in IECON'88, Proc. 14th Annual Conf. (1988) pp. 38-43

[9] Ginsburg, S.: On the Length of the Smallest Uniform Experiment which Distinguishes the Terminal States of a Machine, in J. Assoc. Comput. Mach 5 (1958) pp. 266-280

[10] Göhring, W.: Minimal Initializing Word: A Contribution to Cerny's Conjecture, in Journal of Automata, Languages and Combinatorics 4 (1997) pp. 209-226

[11] Jungers, R. M., Blondel, V. D.: Observable Graphs, in arXiv:cs/0702091v1 [cs.MA] 16 Feb 2007

[12] Kelemen, J.: Agents from Functional-Computational Perspective, in Acta Polytechnica Hungarica 3 (2006) No. 4, pp. 37-54

[13] Kranakis, E., Santoro, N., Sawchuk, C. and Krizanc, D.: Mobile Agent Rendezvous in a Ring, Proceedings of the 23rd International Conference on Distributed Computing Systems, May 19-22, 2003, p. 592

[14] Mateescu, A. and Salomaa, A: Many-valued Truth Functions, Cerny's Conjecture and Road Coloring, in EATCS Bulletin 68, 1999, pp. 134-150

[15] North, M. J., Macal, C. M.: Managing Business Complexity: Discovering Strategic Solutions with Agent-based Modeling and Simulation, Oxford University Press: New York, NY, 2007 ISBN 0195172116

[16] Roman, A., Foryś, W.: Lower Bound for the Length of Synchronizing Words in Partially Synchronizing Automata, in Sofsem 2008 (W. Geffert et al. eds.), Springer Verlag Berlin, Heidelberg 2008, pp. 448-459

[17] Rystsov, I. K.: Reset Words for Commutative and Solvable Automata in Theoret. Comput. Sci. 172 (1997), pp. 273-279

[18] Sebestyénová, J.: Case-based Reasoning in Agent-based Decision Support System. Acta Polytechnica Hungarica 4 (2007), No. 1, pp. 127-138

[19] Trahtman, A. N.: The Road Coloring Problem, in arXiv:0709.0099 [es.DM] 21 Dec 2007

[20] Vaščák, J.: Navigation of Mobile Robots Using Potential Fields and Computational Intelligence Means. Acta Polytechnica Hungarica 4 (2007) No. 1, pp. 63-74