UDC 623.764

P.S. SAPATY[*]

DISTRIBUTED MANAGEMENT IN CRISIS AND EMERGENCY SITUATION

[*]Institute of Mathematical Machines and Systems National Academy of Sciences of Ukraine, Kyiv, Ukraine

Анотація. *Метою даної статті є аналіз і демонстрація можливого застосування розробленої високорівневої Технології просторового захоплення (ТПЗ), а також її базової Мови просторового захоплення (МПЗ) для вирішення різних локальних і глобальних завдань, пов'язаних із кризовими та надзвичайними ситуаціями в динамічних розподілених середовищах. Коротко викладені основні поняття ТПЗ, рекурсивної організації МПЗ і її мережевого інтерпретатора, безліч взаємодіючих копій якого можуть встановлюватися по всьому світу і інтегруватися з іншими системами або працювати автономно у критичних ситуаціях. На МПЗ описані основні операції щодо створення і управління розподіленою мережею, які можуть працювати поверх існуючих систем зв'язку, таких, як інтернет, або самостійно виконувати функції мережевих протоколів високого рівня у разі нелокальних криз і катастроф. Досліджені і продемонстровані в МПЗ дві прикладні області з можливістю виникнення в них кризових ситуацій. Перша стосується повністю розподіленого аналізу і відстежування множинних мобільних об'єктів у розподілених просторах зі складними і заплутаними маршрутами, що можуть бути пов'язані з крилатими ракетами, об'єктами оборони та сміттям у космічному просторі, а також масовою міграцією людей через міжнародні кордони. Інша область належить до розподілених соціальних мереж, в яких різні співтовариства з різними культурами, традиціями або релігіями можуть бути просторово близькі одне до одного, співпадати або перетинатися. На МПЗ показано, як моделювати просторову динаміку таких угруповань, регулярно знаходити топографічні центри різних співтовариств і оцінювати відстань між ними, що може бути корисним для прогнозування та запобігання різних соціальних конфліктів. Ця технологія в її попередніх варіантах застосовувалась і апробовувалась у різних країнах, а її остання версія може бути легко встановлена за угодою на будь-яких платформах.*

Ключові слова: *кризи і управління надзвичайними ситуаціями, мережеві технології високого рівня, Мова просторового захоплення, розподілене відстежування мобільних об'єктів, соціальна динаміка і конфлікти.*

Аннотация. *Целью данной статьи являются анализ и демонстрация возможного применения разработанной высокоуровневой Технологии пространственного захвата (ТПЗ), а также ее базового Языка пространственного захвата (ЯПЗ) для решения различных локальных и глобальных задач, связанных с кризисными и чрезвычайными ситуациями в динамических распределенных средах. Кратко изложены основные понятия ТПЗ, рекурсивной организации ЯПЗ и его сетевого интерпретатора, множество взаимодействующих копий которого может устанавливаться по всему миру и интегрироваться с другими системами или же работать автономно в критических ситуациях. В ЯПЗ описаны базовые операции по созданию и управлению распределенной сетью, которые могут работать поверх существующих систем связи, таких, как интернет, или же самостоятельно выполнять функции сетевых протоколов высокого уровня в случае нелокальных кризисов и катастроф. Исследованы и продемонстрированы в ЯПЗ две прикладные области с возможностью возникновения в них кризисных ситуаций. Первая касается полностью распределенного анализа и отслеживания множественных мобильных объектов в распределенных пространствах со сложными и запутанными маршрутами, которые могут быть связаны с крылатыми ракетами, объектами обороны и мусором в космическом пространстве, а также массовой миграцией людей через международные границы. Другая область относится к распределенным социальным сетям, в которых разные сообщества с разными культурами, традициями или религиями могут быть пространственно близки друг к другу, совпадать или пересекаться. В ЯПЗ показано, как моделировать пространственную динамику таких группировок, регулярно находить топографические центры различных сообществ и оценивать расстояние между ними, что может быть полезным для*

ISSN 1028-9763. Математичні машини і системи, 2019, № 1

прогнозирования или предотвращения различных социальных конфликтов. Данная технология в ее предыдущих вариантах применялась и апробировалась в разных странах, а ее последняя версия может быть легко установлена по соглашению на любых платформах.
Ключевые слова: *кризисы и управление чрезвычайными ситуациями, сетевые технологии высокого уровня, Язык пространственного захвата, распределенное отслеживание мобильных объектов, социальная динамика и конфликты.*

Abstract. *The purpose of this paper is to analyse and demonstrate possible application of the developed high-level Spatial Grasp Technology, SGT, and its Spatial Grasp Language, SGL, for dealing with different local and global crisis and emergency situations in dynamic distributed environments. Main concepts of SGT, recursive organization of SGL, and its networked interpreter are briefed, where numerous communicating interpreter copies can be installed worldwide and integrated with other systems or operate autonomously in critical situations. Basic network creation and management operations are described in SGL which may operate on top of existing communication systems like internet or serve as high level network protocols on their own in case of nonlocal crises and disasters. Two crisis-prone application areas are investigated and demonstrated in SGL. The first one is dealing with fully distributed analysis and tracing of multiple mobile objects in distributed spaces with complex and tricky routes, which may relate to cruise missiles, defence objects and debris in outer space, or massively migrating individuals through international borders. The other considered area represents distributed social networks in which different communities, with different cultures, traditions or religions may spatially coincide. It is shown in SGL how simulate spatial dynamics of such societies, regularly find topographical centres of different communities and evaluate distances between them, which may be useful for prediction or prevention of different social conflicts. The proposed technology had trial implementations and applications in different countries, and its latest version can be readily installed by agreement on any platforms needed.*
Keywords: *crises and emergency management, high-level networking technology, Spatial Grasp Language, distributed tracing of mobile objects, social dynamics and conflicts.*

1. Introduction

With world dynamics growing rapidly, such words as disaster, crisis, and emergency are frequently used in everyday life and in different points throughout the Globe, with detailed clarification and comparison of such terminology found in existing publications [1]. Emergency response and crisis management are already vital activities and essential part of infrastructures in different organizations [2]. Crisis and security management are also considered in a global scale like dealing with disasters that could break global communications and the internet [3], associated with missile defence [4], and even moving to outer space [5]. Global terrorism [6], cyber attacks [7], natural disasters with their social and political impact [8, 9], religious conflicts [10], as well as many others with relation to international security [11, 12], are the areas were disasters, crises and emergency are common, with urgent need of their effective prevention, alleviation, and management. In this paper, we are briefing the developed high-level networking control, processing, and management technology [13–17] which is suitable for runtime dealing with different crisis and emergency situations, also showing practical examples of its application in the area mentioned.

The rest of this paper is organized as follows. Section 2 describes the high-level Spatial Grasp Technology, SGT, together with its basic Spatial Grasp Language, SGL, also organization and main features of the networked SGL Interpreter. Section 3 shows expression and solution in SGL of the basic and most vital network communication and management mechanisms, which can fully operate on their own even if traditional communications, internet including, not operating. These include network creation from scratch, finding and collecting any path between nodes, forming any spanning tree as well as shortest path tree from a node to all other nodes with accumulating shortest paths to all nodes in the starting node, also setting routing tables in particular nodes for this tree, and finally, creating routing tables in all nodes allowing shortest paths communications between any nodes of the network. Section 4 demonstrates how to organize in SGL

the discovery and tracing of complexly moving targets, which may be cruise missiles or any other objects in terrestrial or celestial environments, to be accomplished in a fully distributed and parallel mode. Section 5 shows how to outline different communities in a distributed social network, find their topographical centres and evaluate physical distance between them, which may be useful for preventing possible social conflicts, while doing this repeatedly with simulation of spatial mobility of individuals in time. Section 6 concludes the paper.

2. Spatial Grasp Technology (SGT)

2.1. General Features

Within SGT, a high-level scenario for any task to be performed in a distributed world is represented as an active self-evolving pattern rather than traditional program, sequential or parallel. This pattern, written in a high-level Spatial Grasp Language, SGL, and expressing top semantics of the problem to be solved, can start from any world point, being as the source of pattern's activity. It then spatially propagates, replicates, modifies, covers and matches the distributed world in a parallel wavelike mode, as shown in Fig. 1.

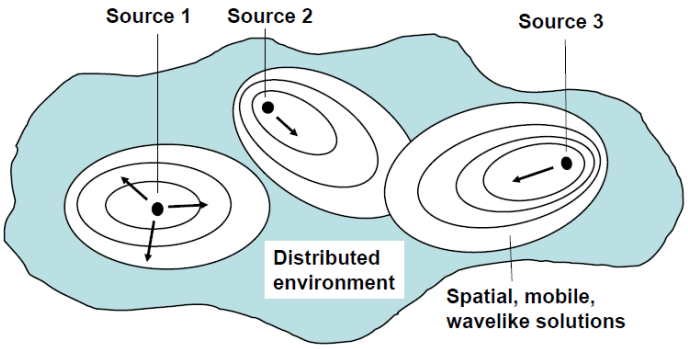

Figure 1 –Spatial pattern growth & coverage & matching

The self-spreading & matching patterns can create knowledge infrastructures arbitrarily distributed between system components (like humans, robots, sensors, etc.). These infrastructures, which may be left active, can effectively express distributed databases, command and control, situation awareness, autonomous decisions, as well as any other existing or hypothetical computational and control models.

2.2. Spatial Grasp Language (SGL)

SGL allows us to directly move through, observe, and provide any actions and decisions in fully distributed environments (whether physical, virtual, executive, or combined). It has universal recursive structure, shown in Fig. 2, capable of representing any parallel and distributed algorithms operating on, over, or in spatially scattered data or other distributed systems.

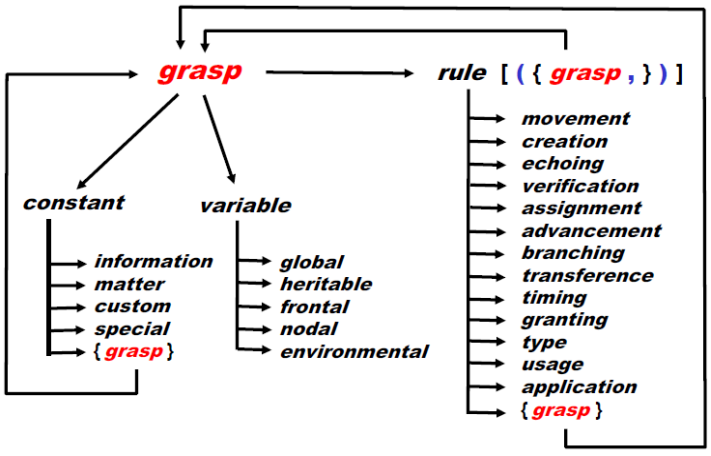

Figure 2 – SGL recursive syntax

An SGL scenario develops as parallel transition between sets of progress points (or props), with self-modified and self-replicating scenario code freely moving in distributed spaces. Starting from a prop, an action may result in new props (which may be multiple). Each prop has a resulting value, which may be arbitrarily complex, and resulting state (one of: thru, done, fail, and abort). Different actions may evolve independently or interdependently from the same prop, splitting and parallelizing in space. Actions may also spatially

succeed each other, with new ones applied sequentially or in parallel from the props reached by previous actions.

Elementary operations can directly use states and values of props reached by other actions whatever complex and remote they might be. Any prop can associate with a position in physical, virtual, executive or combined world. Staying with world points, it is possible to directly access and impact local world parameters in them. Overall organization and control of the breadth and depth space navigation and coverage is provided by SGL rules, which may be nested. These rules, for example, can be: elementary arithmetic, string, or logic operation; hop in a physical, virtual, execution, or combined space; hierarchical fusion and return of both local and remote data; distributed control, both sequential and parallel; a variety of special contexts for navigation in space, influencing embraced operations and decisions; type or sense of a value or its chosen usage guiding automatic interpretation; creation or removal of nodes and links in distributed knowledge networks. A rule can also be a compound one integrating other rules or defined as a result of operations of arbitrary complexity.

Working in fully distributed physical, virtual or executive environments, SGL has different types of variables, called spatial, effectively serving multiple cooperative processes: Heritable Variables – starting in a prop and serving all subsequent props which can share them in both read & write operations; Frontal Variables – transferred on wavefronts between consecutive props and replicated if multiple new props emerge; Environmental Variables accessing different elements of physical and virtual words when navigating them, also certain parameters of SGL interpreter; and Nodal Variables as a temporary property of world nodes, accessed and shared by all activities associated with and reaching these nodes. These types of variables, especially when used together, allow us to create flexible and robust spatial algorithms working in between components of distributed systems rather than in them. Such algorithms can replicate, spread and migrate in distributed environments (partially or as a whole), always preserving global integrity and control.

2.3. SGL Networked Interpreter

An SGL interpreter consists of a number of specialized modules handling and sharing specific data structures, as in Fig. 3.

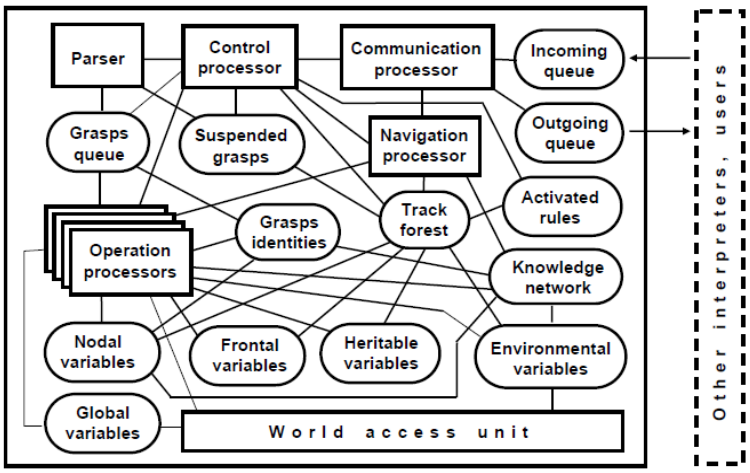

Figure 3 – SGL interpreter organization and main components

The interpreters can communicate with each other, and their distributed network can be mobile and open, changing the number of nodes and communication structure at runtime. The backbone and nerve system of the distributed interpreter is its dynamic spatial track system with its parts kept in the Track Forest memory of local interpreters. It is logically interlinked with similar parts in other interpreter copies thus providing altogether global control coverage. The distributed track structure enables for both hierarchical and horizontal control, also remote data and code access, with high integrity of emerging parallel and distributed solutions achieved without any centralized resources. Dynamically created track forests, spanning the systems in which SGL scenarios evolve, are also used for supporting spatial variables and echoing & merging control states and remote data, while self-optimizing in parallel echo processes. They also route further grasps

to the positions in physical, virtual, executive or combined spaces reached by the previous grasps, uniting them with frontal variables left there by preceding grasps.

The distributed SGL interpreter may have any number of communicating nodes, up to thousands to millions to billions, effectively converting the whole world into a universal spatial machine operating under spreading intelligent scenarios. Any number of such scenarios can operate simultaneously (cooperatively or competitively) while starting at any time and from same or different world points. The SGL interpreter copies may be integrated with (or implanted into) any existing systems, popular media and email including. They can also be concealed if to operate in hostile environments, allowing the latter to be analyzed and impacted in a stealth manner.

3. Distributed Network Management Basics in SGT

3.1. Exemplary Network Creation

We are starting here from scratch, having only elementary communications between different nodes limited by some threshold physical distance Th, and without any communication infrastructure between them. Only copies of SGL interpreter are installed in each node. Different stages of the possible network infrastructure creation are shown in Fig. 4, with nodes named a, b, c, d, e, f, g, and h originally distributed in space as shown in Fig. 4, a.

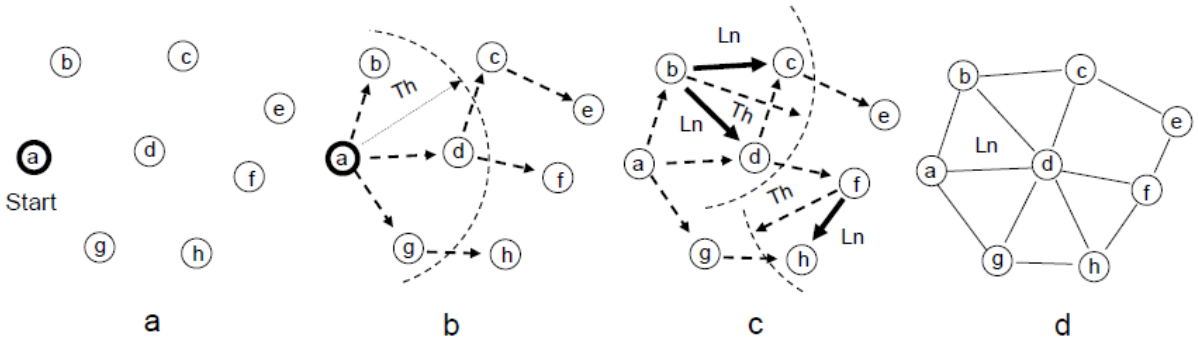

Figure 4 – Distributed network creation

Let us start in some node, let it be a as in Fig. 4, a, after writing in SGL:

```
hop(a)
```

Starting in node a and stepwise reaching all other nodes by hopping to neighboring nodes within the given maximum allowed distance Th between them, as shown in Fig 4, b (to avoid looping, the nodes are allowed to be entered first time only):

```
hop(a);
repeat(hop_first(nodes(all, within(Th))))
```

Starting in node a and reaching all other nodes as before, but together with creation of links-channels of type Ln between all neighboring nodes within Th distance, as shown in Fig. 4, c, can be done by:

```
hop(a);
repeat(hop_first(nodes(all, within(Th)));
       stay(hop(nodes(all, within(Th))); BACK > NAME;
            create(link(Ln), node(BACK))))
```

To avoid competing attempts of establishing same link between two nodes when starting from them both, we allow doing this only after comparing their names, allowing the node with higher value to dominate (could be organized vice versa too).

We will finally be having the full network shown in Fig. 4, d with all established links-channels of the type Ln, as was required.

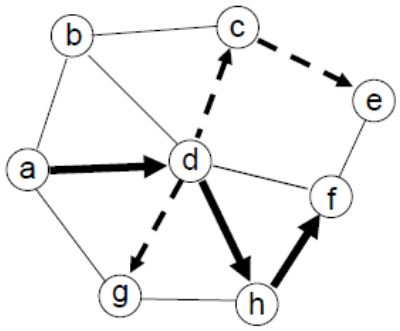

Figure 5 – Reaching a node from another node

3.2. Finding and Collecting Any Path between Nodes

Having created the network infrastructure, as above, we may, starting in some node like a, reach any other needed node like f, by the following SGL scenario navigating the network in a wavelike mode, i.e. stepwise and in parallel (see Fig. 5).

```
hop(a); repeat(hop_first(links(all));
                or((NAME == f; done), stay))
```

We may additionally decide to collect the passed path and organize its output at the destination node, as follows.

```
hop(a); frontal(Path = NAME);
repeat(hop_first(link(any)); Path &&= NAME;
       or((NAME == f; output(Path); done), stay))
```

Arbitrary path found between these two nodes (may not be optimal like the one in Fig. 5) is printed in node f like: (a, d, h, f). Such path can also be issued in the starting node a by the modified scenario:

```
hop(a); frontal(Path = Name);
output_repeat(hop_first(link(any)); Path &&= NAME;
              or(blind(NAME == f; Path), stay))
```

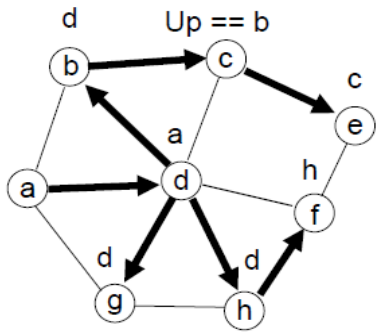

Figure 6 – Creating any Spanning Tree from a node to all other nodes

3.3. A Spanning Tree from a Node to All Other Nodes

In the previous example, we have found any path from a node to some particular node, which was then issued outside the network within the network asynchronous wavelike navigation. In similar network navigation, we may create a Spanning Tree (ST) starting in the same node and covering the whole network, thus explicitly showing possible paths to all other nodes. This can be easily done with registering this ST directly in the distributed network structure, by remembering node predecessor names in a special variable Up associated with each node, as follows, see also Fig. 6:

```
nodal(Up); hop(a);
repeat(hop_first(links(any)); Up = BACK)
```

Such a tree can directly guide movement from any node, like e, to the starting node a:

```
hop(e); repeat(hop(Up))
```

We may also, if needed, collect this passed path in both ways and issue it in node a.
a) If to register from node e to node a, will be having Path (e, c, b, d, a):

```
hop(e); Path = NAME;
```

```
repeat(hop(Up); Path &&= NAME); output(Path)
```

b) If to register from node a to node e, will be having Path(a, d, b, c, e):

```
hop(e); Path = NAME;
repeat(hop(Up); Path = NAME && Path); output(Path)
```

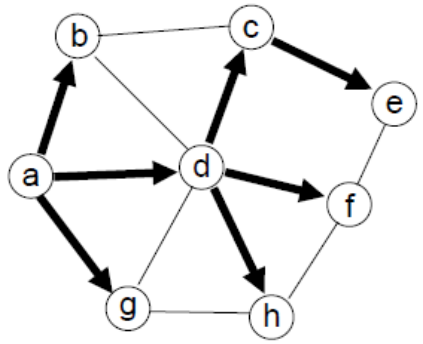

Figure 7 – Creating Shortest Path Tree from a node to all other nodes

3.4. Shortest Path Tree (SPT) from a node to all other nodes

With some modification, we may create and register instead of any Spanning Tree, a Shortest Path Tree (SPT) from a node to all other nodes, as shown in Fig. 7 (which, in general, may be one of a number of such SPTs).

The following scenario can accomplish this with registering the resultant SPT in the network structure similarly to the ST scenario before. The main difference will be with re-registering the SPT predecessor nodes in variables Up in nodes if better (shorter) solutions for shortest paths to these nodes appear available.

```
nodal(Dist, Up); hop(a); Dist = 0; frontal(Far);
repeat(hop(links(all)); Far += 1;
       or(Dist == nil, Dist > Far); Dist = Far; Up = BACK)
```

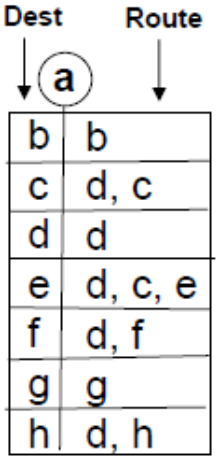

Figure 8 – Registering in the SPT root node the shortest paths to all other nodes

Registering in the starting node of the shortest paths to all other nodes on the basis of SPT found can be done by the synchronous two-vector structure at the starting node, as shown in Fig. 8 (keeping names of all other nodes in Dest, and corresponding paths to them in Route):

```
hop(a); nodal(Dest, Route); hop(all_other);
frontal(Path);
repeat(Path = NAME && Path; hop(Up));
seize(Dest &&= Path[last]; Route &&= unit(Path))
```

Using collected shortest paths in the SPT root node, any other node can be conveniently reached, like, for example, node e:

```
hop(a); frontal(Path) = Route[order(Dest, e)];
repeat(hop(link(any), node(withdraw(Path, 1))))
```

The path followed from node a to node e will be as: a→d→c→e.

For this SPT we can also create routing tables (RT) in all nodes having descendants (i.e. a, d and c) rather than collecting full shortest paths in a to all other nodes, as before, by the following scenario.

```
hop(a); nodal(Dest, Next); hop(all_other); frontal(Fin) = NAME;
repeat(hop(Up); seize(Dest &&= Fin; Next &&= BACK))
```

The result is depicted in Fig. 9 where vector Dest keeps names of all destination nodes in relation to the current node (similar to Fig. 8), and the synchronous to it vector Next provides just

names of next hop nodes towards reaching the needed destinations (rather than full paths to these destinations, as before).

Movement from the SPT root a to any other node (let it be e again) via the obtained set of routing tables in nodes can be done by the following scenario:

```
hop(a); repeat(hop(link(any), node(Next[order(Dest, e)])))
```

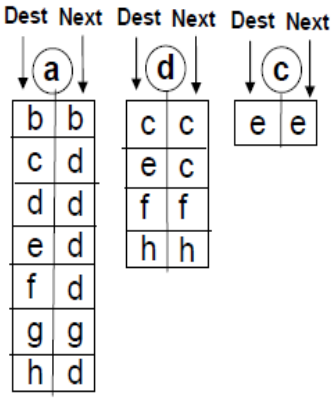

Figure 9 – Routing tables in nodes for SPT in Fig. 7

Will be having the same result as before when remembered full paths to other nodes in the starting node, i.e.: a→d→c→e.

3.5. Creating Routing Tables from All Nodes to All Other Nodes

The previous routing tables solution related only to the single SPT providing shortest paths from the root node a to all other nodes. We can also easily organize in SGL the finding of routing tables placed in each node and providing altogether shortest paths from any node to any other nodes, as follows.

```
nodal(Dest, Next);
hop(nodes(all)); color(nodal(Dist, Up));
sequence(
  (frontal(Far); Dist = 0;
   repeat(hop(links(all)); Far += 1;
          or(Dist == nil, Dist > Far); Dist = Far; Up = BACK)),
  (frontal(Fin = NAME);
   repeat(hop(links(all); Up == BACK;
          seize(Dest &&= Fin; Next &&= BACK))))
```

This scenario is based on parallel creation of SPTs from all nodes to all other nodes, with such SPTs shown in Fig. 10.

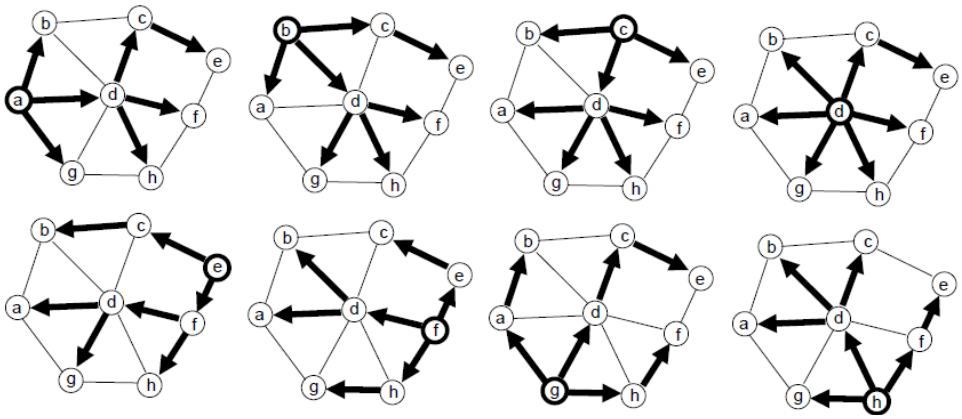

Figure 10 – SPTs from all nodes of the network to all other nodes

The obtained routing tables in all network nodes are shown in Fig. 11, with same meanings as in Fig. 9 of synchronized Dest and Next vectors, now present in all nodes.

An example of following SP from any node, say g, to any other node, like e, via the RTs of Fig. 11, will be by using the same scenario as shown before for a single SPT from node a:

```
hop(g); repeat(hop(link(any), node(Next[order(Dest, e)])))
```

This scenario will follow the path g→d→c→ e with using RTs of Fig. 11 in nodes g, d and c.

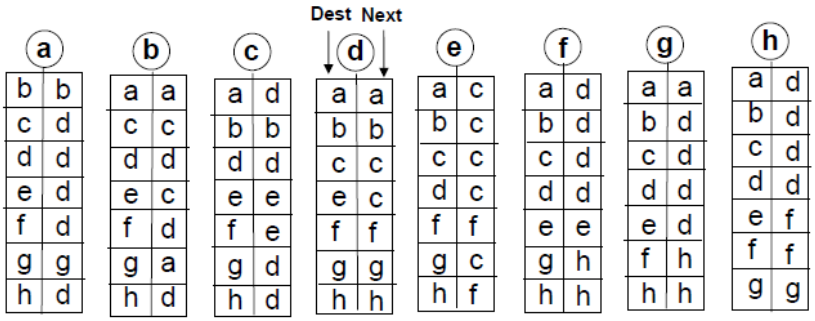

Figure 11 – Routing tables providing shortest paths from all nodes to all other nodes of the network

4. Spatial Objects Discovery and Tracking

Distributed sensor networks operating under SGT can catch and follow any moving objects throughout the whole region despite limitations of individual sensors, as in Fig. 12. The latter shows some area covered with a network of communicating radar stations, each having SGL interpreter installed, with presumably hostile objects like, say, cruise missiles moving through the area.

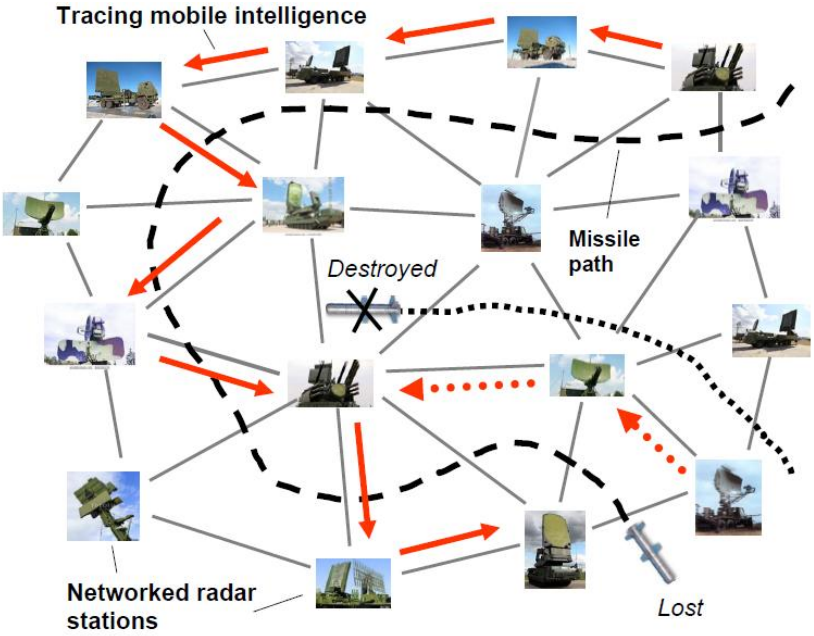

Figure 12 – Distributed objects tracking & destruction

The sensor-radar first seeing a new object (i.e. which is within the given visibility threshold) is becoming the start of distributed operation, after which the object is supposed to be seen by this radar for some time or may move and be visible by other sensors. Object's moving & behavior history can be collected & updated at each passed radar sensor by the SGT-produced mobile spatial intelligence individually assigned to this particular object and following its electronically via the radar network, just following its physical move in the real world.

Depending on the collected history, such object may be tried to be destroyed, it may also happen to be ultimately lost after safely passing through the radar-controlled area. The SGL scenario shown in Fig. 13 will be following the moving object wherever it may go, despite its possible tricky route, like that of a cruise missile. The scenario can operate with multiple moving ob-

jects appearing at any time, where each sensor regularly searches for new targets, and each new target is assigned individual tracking intelligence which can propagates in distributed virtual space in parallel with other intelligences, following the physical movement of targets.

```
1  hop(all_nodes);
2  frontal(Object, History, Threshold = …);
3  whirl(
4    Object = search(aerial, new));
     visibility(Object) >= Threshold;
5    free_repeat(
6      loop(
         visibility(Object) >= Threshold);
         accumulate(History, Object);
         if(negative(History), blind_destroy(Object)));
7      max_destination(
         hop(neighbors_all); visibility(Object));
8      if(visibility(Object) < Threshold,
          blind_output(Object, History, 'lost')))))
```

Figure 13 – SGL scenario for distributed tracing of mobile objects

Some additional explanation of different scenario stages numbered in Fig. 13 may be as follows.

1. Starting in all sensor nodes in parallel.

2. Describing types and initial values of the spatial variables used, like for the visibility threshold.

3. Organizing continuous, endless, looping of the remaining stages 4 to 8 whatever their success or failure.

4. Trying to find a new areal object with sufficient visibility, continuing the remaining stages 5 to 8 if such an object discovered, otherwise terminating this branch.

5. Making the following sections 6 to 8 mobile and autonomous (which will also be repeated as much as needed) from the starting, main scenario body, thus allowing the stage 4 concentrate on the search of new objects again.

6. Looping in the current sensor-radar node if the object still has sufficient visibility, collecting its behavior history, and trying to destroy it depending on the history collected, if a decision is taken.

7. After losing object's visibility in the current node (thus appearing lower than the threshold given), contacting all neighboring sensor-nodes in parallel and moving to the neighboring sensor in which this object's visibility is highest.

8. Checking the visibility of the same object in the new node, and in case it is lower than the given threshold, declaring the failure of the object's observation along with its history collected, also terminating this scenario branch). Otherwise continue observation of the object from stage 6 with its sufficient visibility.

The offered organization of tracing and impact of multiple moving objects in distributed environments by networked sensors with embedded SGL interpreters and virus-like mobile spatial intelligence in SGL, which can collectively operate without any, often vulnerable, central resources, can also be effectively used in many other areas. These may include advanced operations in outer space related to missile defense or dealing with space debris, tracing international crimi-

nals, flow of different goods, materials and finances throughout the whole world, as well as control and management of human migration.

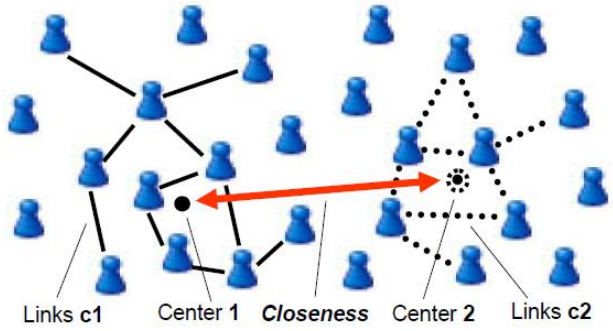

Figure 14 – Finding centres of different communities

5. Finding Centres of Different Communities and Analyzing Distances between Them

Of practical interest may be finding topographical centres of different social communities with assessment of physical distances between them, say, for preventing possible conflicts as these groupings may be pursuing quite different even hostile to each other cultures, religions, traditions, and principles. In Fig. 14, different communities are expressed by different types of links between their members (like c1 and c2).

The following scenario outlines such communities and finds their topographical centres with evaluating and outputting physical distance, or Closeness, between them.

```
nodal(Center1, Center2, Closeness);
Center1 = average(hop_nodes(all); yes(hop_links(c1)); WHERE);
Center2 = average(hop_nodes(all); yes(hop_links(c2)); WHERE);
Closeness = distance(Center1, Center2);
output(Closeness)
```

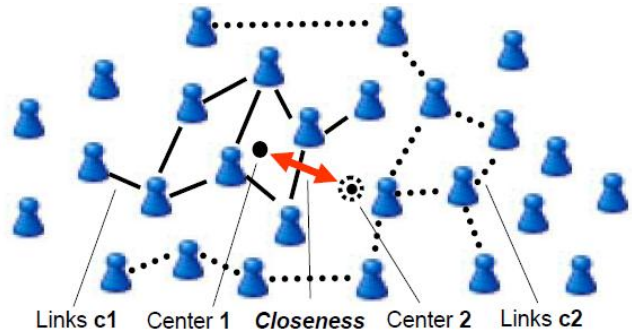

Figure 15 – Communities may become too close to each other

The found value in Closeness may indicate existence of some unwanted trends in a multicultural society, say, if below a certain threshold (another example, with different Closeness value shown in Fig. 15), which may need introduction and application of certain educational, organizational, or even security measures.

We can also model possible spatial mobility of members belonging to different groupings in time (which may differ for different communities), with regular finding heir topographical centres and measuring physical closeness between them, also providing warnings if this distance becomes suspicious. This can be implemented by extended SGL scenario shown in Fig. 16.

Additional explanation of different, numbered, scenario stages in Fig. 16 may be as follows.

1. Description of different types of used spatial variables with initial values of some of them.

2. Organizing three independent parallel branches consisting of: stage 3, stage 4, and stages 5 to 8.

3. Simulating continuous randomized physical movement of individuals belonging to the first community, with proper delay between iterations.

4. Simulating continuous randomized physical movement of individuals belonging to the second community, with another delay between repetitions.

5. Organizing endless repetitive operation of the third branch consisting of stages 6 to 8.

6. Finding topographical centers of all individuals belonging to the first and second communities at different, current moments of time.

7. Calculating and assessing physical distance between the centers found with issuing warning when their closeness becomes dangerous.

8. Setting certain time delay between repetitive invocations of stages 6 and 7.

Many other tendencies and problems in distributed social systems can be effectively simulated and analyzed with the help of SGT [16].

```
1  nodal(Center1, Center2, Closeness, Delay1 = …, Delay2 = …, Delay3 =…,
         Threshold = …, Maxshift1 = dX1_dY1, Maxshift2 = dX2_dY2);
2  branch(
3    (hop_nodes(all); yes(hop_links(c1));
     repeat(WHERE + random(Maxshift1); sleep(Delay1))),
4    (hop_nodes(all); yes(hop_links(c2));
     repeat(WHERE + random(Maxshift2); sleep(Delay2))),
5    repeat(
6      Center1 = average(hop_nodes(all); yes(hop_links(c1)); WHERE);
       Center2 = average(hop_nodes(all); yes(hop_links(c2)); WHERE);
7      Closeness = distance(Center1, Center2);
       if(Closeness <= Threshold, output('Danger: ', Closeness));
8      sleep(Delay3)))
```

Figure 16 – SGL scenario for modelling dynamics and evaluating closeness of communities

6. Conclusions

We have briefed the invented and patented high-level networking technology which allows us to solve complex problems in large distributed environments in parallel and fully distributed mode, without vulnerable central resources, effectively using unlimited spatial mobility of recursive control code dynamically matching any systems. Expression in SGL of tracing and investigating of complexly moving objects by intelligent sensor networks, and simulating and analysing dynamics and closeness of different communities in social networks were demonstrated too. The future plans include investigating more areas for SGT solutions of emergency, crisis, and security problems, also further development of ideas and implementations expressed in [11, 12], with new book related to international security based on SGT currently in preparation. SGL scenarios for solving different networking problems appear to be extremely compact (can even be shorter if abbreviations for different rules and special constants used, as in previous versions of SGL, see [14, 15]), which allows us to create and modify solutions for crisis and emergency problems at runtime, on the fly. The latest technology version can be readily installed by agreement on any platforms needed.

REFERENCES

1. Al-Dahash H.F., Thayaparan M., Kulatunga U. Understanding the Terminologies: Disaster, Crisis and Emergency / Chan P.W., Neilson C. J. (Eds.) *Proc. of the 32nd Annual ARCOM Conference* (5–7 September 2016). Manchester, UK: Association of Researchers in Construction Management. Vol. 2. P. 1191–1200. URL: http://usir.salford.ac.uk/39351/.

2. Dawkins P.W. Emergency Response and Crisis Management Plan. *Bennett College*. August 2, 2017. 57 p. URL: http://www.bennett.edu/wp-content/uploads/2016/06/Emergency_Response_Crisis_ Management_Plan.pdf.

3. Baraniuk C. The Disastrous events that would break the internet. *BBC Future*. 2015. 11 March. 10 p. URL: http://www.bbc.com/future/story/20150310-how-to-break-the-internet.

4. Missile Defense Review. *Office of the Secretary of Defense*. 2019. 24 p. URL: https://www.defense.gov/Portals/1/Interactive/2018/11-2019-Missile-Defense-Review/The%202019%20MDR_Executive%20Summary.pdf.
5. Blount P.J. Targeting in Outer Space: Legal Aspects of Operational Military Actions in Space. *Harvard Law School National Security Journal*. 2012. 14 p. URL: http://harvardnsj.org/2012/11/targeting-in-outer-space-legal-aspects-of-operational-military-actions-in-space/.
6. Global Terrorism Index 2016. The *Institute for Economics and Peace (IEP)*. 105 p. URL: http://economicsandpeace.org/wp-content/uploads/2016/11/Global-Terrorism-Index-2016.2.pdf.
7. Melnick J. Top 10 Most Common Types of Cyber Attacks. May 15, 2018. 12 p. URL: https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/.
8. Natural Disasters and Societal Safety / eds. R.H. Gabrielsen, S. Lacasse. *Det Norske Videnskaps-Akademi*. 2016. 140 p. URL: https://www.ntva.no/fellesrapport2015/naturaldisasters/assets/common/downloads/Natural%20Disasters%20and%20Societal%20Safety.pdf.
9. Albrecht F. The Social and Political Impact of Natural Disasters – Investigating Attitudes and Media Coverage in the Wake of Disasters. *Acta Universitatis Upsaliensis UPPSALA*. 2017. 61 p. URL: https://uu.diva-portal.org/smash/get/diva2:1090236/FULLTEXT01.pdf.
10. Armstrong K. The Role of Religion in Today's Conflict. January 2016. 11 p. URL: https://www.unaoc.org/repository/Armstrong_Religion_Conflict.pdf.
11. Sapaty P.S. Holistic Spatial Management of International Security. *Austin Journal of Robotics & Automation*. 2018. Vol. 4, Issue 1. 8 p. URL: http://austinpublishinggroup.com/robotics-automation/online-first.php.
12. Sapaty P.S. Holistic Spatial Management of International Security. *Mathematical Machines and Systems*. 2018. N 4. P. 11–25.
13. Sapaty P. A Distributed Processing System. European Patent N 0389655; Publ. 10.11.93, European Patent Office; Munich, 1993.
14. Sapaty P. Mobile Processing in Distributed and Open Environments. New York: John Wiley & Sons, 1999. 410 p.
15. Sapaty P. Ruling Distributed Dynamic Worlds. New York: John Wiley & Sons, 2005. 255 p.
16. Sapaty P. Managing Distributed Dynamic Systems with Spatial Grasp Technology. Springer, 2017. 284 p.
17. Sapaty P. Holistic Analysis and Management of Distributed Social Systems. Springer, 2018. 234 p.

Стаття надійшла до редакції 08.02.2019