International Relations and Diplomacy, January 2019, Vol. 7, No. 01, 14-36
doi: 10.17265/2328-2134/2019.01.003

Conflict and Emergency Management in a Post-Liberal World

Peter Simon Sapaty
Institute of Mathematical Machines and Systems, Kiev, Ukraine

The world is steadily moving to the post-liberal order with the urgent need of novel organizational and security approaches, also new levels of international cooperation, in order to support its stability and prosperity. The developed high-level Spatial Grasp Technology (SGT) and its Spatial Grasp Language (SGL) are briefed which may be particularly useful for solving numerous conflicts and crises problems emerging in different areas during this transitional period, in both local and global scale. SGT employs unlimited spatial scenario mobility and parallel holistic matching of distributed systems, with numerous communicating SGL interpreters potentially installed worldwide. Basic network creation and management operations are described in SGL which may operate on top of existing communication systems or serve individually as high level network protocols in case of non-local crises and disasters. Different operations on social networks are presented in SGL including finding strongest and weakest components with resultant changing of network topologies, also determining distances between different communities for preventing and predicting social conflicts. Fully distributed analysis, and tracing and simulation of multiple mobile objects in distributed spaces with complex routes are shown in SGL related to cruise missiles, defence objects and debris in outer space, as well as massively moving refugees through international borders. 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: post-liberal world, international cooperation, crises and emergency management, high-level networking technology, social dynamics and conflicts, distributed tracing of mobile objects, global spatial solutions

Introduction

For several generations, the world has been governed by what today is usually called as “the global liberal order” (Harari, 2018). Though having many faults and problems, it has proved superior to all alternatives. The liberal world of the early 21st century has become more prosperous, healthy, and peaceful than ever before. Nevertheless, people all over the world are now losing faith in the liberal order. Nationalist and religious views are back in vogue. Governments are increasingly restricting the flow of ideas, goods, money, and people. Walls are popping up everywhere, both on the ground and in cyberspace. Immigration is out; tariffs are in. In order to survive and flourish in the 21st century, humankind needs effective global cooperation and greater trust between countries; such trust should be global. We need to create a global safety-net to protect humans against different shocks. Viable blueprint for such cooperation is offered by liberalism. Nevertheless, governments all over the world are undermining the foundations of the liberal order, and the world may be turning into a network of fortresses. Humankind now faces the triple crisis of nuclear war, climate change, and technological

Peter Simon Sapaty, PhD, Chief Research Scientist, Institute of Mathematical Machines and Systems, National Academy of Sciences, Kiev, Ukraine.

disruption. In the 21st century, we face global problems that even large nations cannot solve by themselves. This “multiplex world” (Acharya, 2017) carries both risks and opportunities for managing international stability, and the world should accept the new realities and search for new ways to ensure peace and stability. There is growing number of investigations, new ideas, and other publications in this emerging area (Pabst, 2017; Duncombe, 2018; Ikenberry, 2018).

During this transitional period from liberal to post-liberal organization, we are witnessing a rapidly growing world dynamics with such words as disaster, crisis, and emergency being frequently used in everyday life and in different points throughout the globe, with detailed clarification and comparison of such terminology found in (Al-Dahash, Thayaparan, & Kulatunga, 2016). Emergency response and crisis management are already vital activities and essential part of infrastructures in different organizations (Dawkins, 2017). Crisis and security management are also considered in a global scale like dealing with disasters that could break global communications and the Internet (Baraniuk, 2015), associated with missile defence (Office of the Secretary of Defense, 2019), and even moving to outer space (Blount, 2012). Global terrorism (The Institute for Economics and Peace [IEP], 2016), cyber attacks (Melnick, 2018), natural disasters with their social and political impact (Gabrielsen & Lacasse, 2016; Albrecht, 2017), religious conflicts (Armstrong, 2016), as well as many others with relation to international security (Sapaty, 2018) 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 (Sapaty, 1993; 1999; 2005; 2017; 2018) which is suitable for runtime dealing with different crisis and emergency situations and especially in the post-liberal period, also showing practical examples of its application in different world areas.

The rest of this paper is organized as follows: Section 2 describes the high-level Spatial Grasp Technology (SGT) together with 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 work on their own even if traditional communications, Internet including, not operating during non-local conflicts and disasters. These include network creation from scratch, finding and collecting any path between nodes, forming any spanning and shortest path trees from a node to all other nodes, and also creating routing tables in all nodes allowing for shortest paths communications between any nodes of the network. Section 4 shows the use of SGT for analysing distributed social networks by finding strong and weak components in them with related topology changes to assist in conflict situations. It also shows how to outline different communities in a distributed social network, find their topographical centres and evaluate physical distances between them for preventing possible social conflicts, while doing this repeatedly with simulation of spatial mobility of individuals in time. Section 5 demonstrates how to organize in SGL the discovery and tracing of complexly moving targets like cruise missiles by distributed sensor networks, also shows possibility of high-level simulation and tracing of movement of multiple objects in space helping to avoid collisions with them for the vehicles launched, with the help of scattered space observation sensors which can be integrated worldwide under SGT. The chapter also offers how to organize distributed simulation and assistance of massive flows of refugees through international borders which may be caused by conflicts, disasters, or climate change. Section 6 concludes the paper.

Spatial Grasp Technology (SGT)

General Features

Within SGT, a high-level scenario for any tasks 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 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 Figure 1.

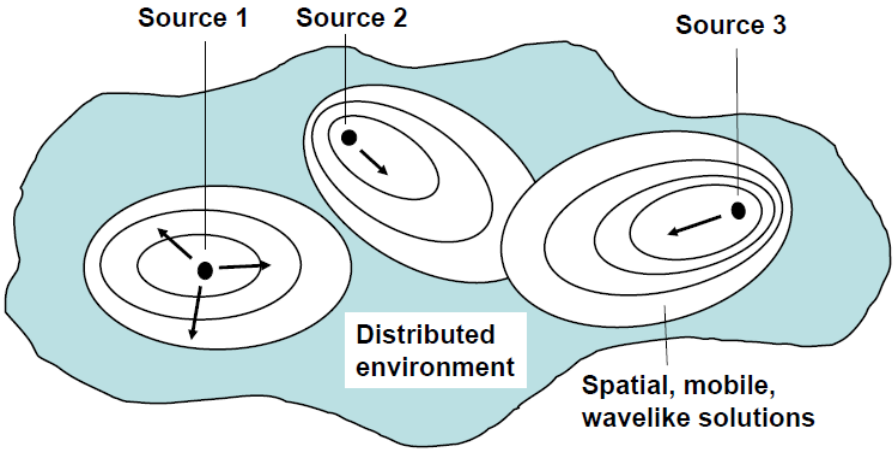

Figure 1. Spatial pattern growth & coverage & matching.

The self-spreading and 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.

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 Figure 2, capable of representing any parallel and distributed algorithms operating on, over, or in spatially scattered data or other distributed systems.

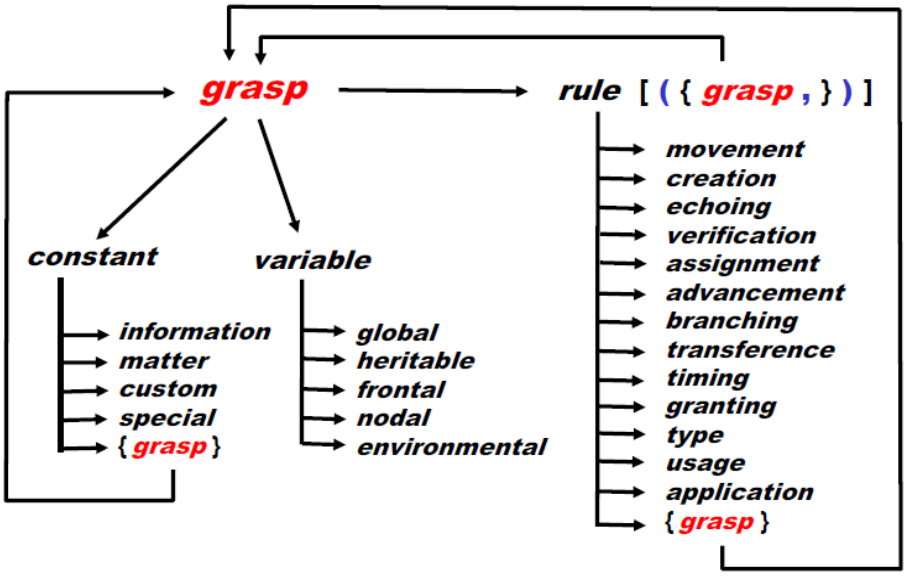

Figure 2. SGL recursive syntax.

An SGL scenario develops as parallel transition between sets of progress points (or props), with self-modified, self-repeating, and self-replicating scenario code freely moving in distributed spaces while losing its utilized parts if not needed any more. Starting from a prop, an action may remain in it or result in new props which may be multiple and remote. 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 props 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 a variety of 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; 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 and remoteness.

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 and 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 worlds when navigating them, also certain externally controllable 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.

Elementary Programming Examples

We are showing below only elementary examples of programming in SGL, where many others, including scenarios of different complexity and civil and defence orientation, can be found in the existing books on this paradigm (Sapaty, 1993; 1999; 2005; 2017; 2018) as well as other numerous publications for the last four decades. Many of them also explain methodology and culture of fully distributed programming in SGL and its previous versions, like WAVE (Sapaty, 1999; 2005).

- Assignment of the sum of values 15, 22, and 14.7 to the variable Result.

  assign (Result, add (15, 22, 14.7))

The variable Result will be created if not existing yet, and will be associated together with the new value with world position where the scenario started. A shortened version in traditional style:

Result = 15 + 22 + 14.7

- Moving physically from the current location independently and simultaneously to new locations (x2, y7) and (x4, y9).

  branch (move (location(x2, y7), move (location(x4, y9)))

  A shortened version will be as follows:

  move (x2, y7), move (x4, y9)

- Creating a virtual node John:

  create (node (John))

Starting from the current world location, a new isolated virtual node with given name John will be created, with the resultant control moving into it. If only node name mentioned, the shortened version will be as:

create (John)

- Extending the virtual network (already having node John) with new link-node pair like “John is father of Bob”.

```
advance (hop (node (John)),
        create (link (+ father of), node (Bob)))
```

The scenario first directly hops into node John and from it creates the mentioned link-node pair, with succession in virtual space provided by rule advance. Simplified version:

hop (John); create (+ father of, Bob)

- Ordering soldier Nick to use robot Fighter to fire by coordinates (x, y) with confirmation of the success or failure.

```
hop (Nick);
report_if ((hop (Fighter); fire (x, y)), success, failure)
```

As shown above, SGL directly and naturally operates with physical, virtual, executive, and just mathematical objects and their combinations, which allows us to use the same single language for various operations and at different levels in distributed system management, thus without traditional extra code and time consuming seams of multilingual organizations.

SGL Networked Interpreter

An SGL interpreter (Sapaty, 2017; 2018) consists of a number of specialized modules handling and sharing specific data structures, as in Figure 3.

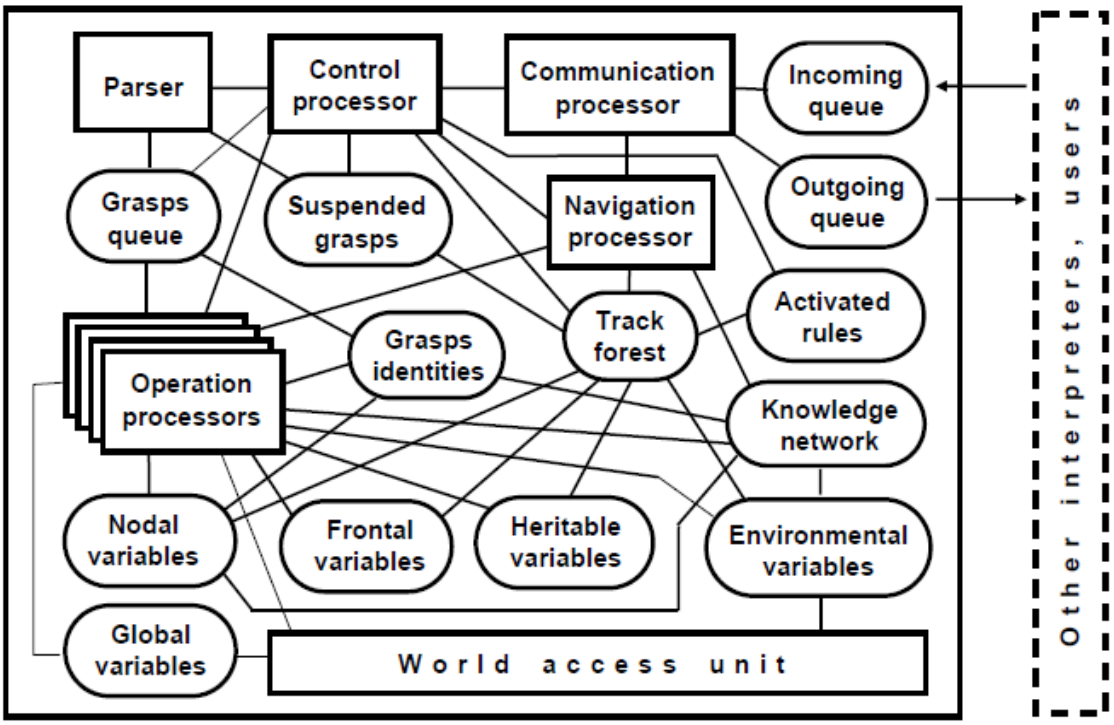

Figure 3. SGL interpreter organization and main components.

The interpreter copies 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, 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 emails, including different types of robots, too. They can also be concealed if dedicated to operations in hostile environments, allowing the latter to be analyzed and impacted in a stealth manner.

Distributed Network Management Basics in SGT

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 Figure 4, with nodes named a, b, c, d, e, f, g, and h originally distributed in space as shown by Figure 4(a).

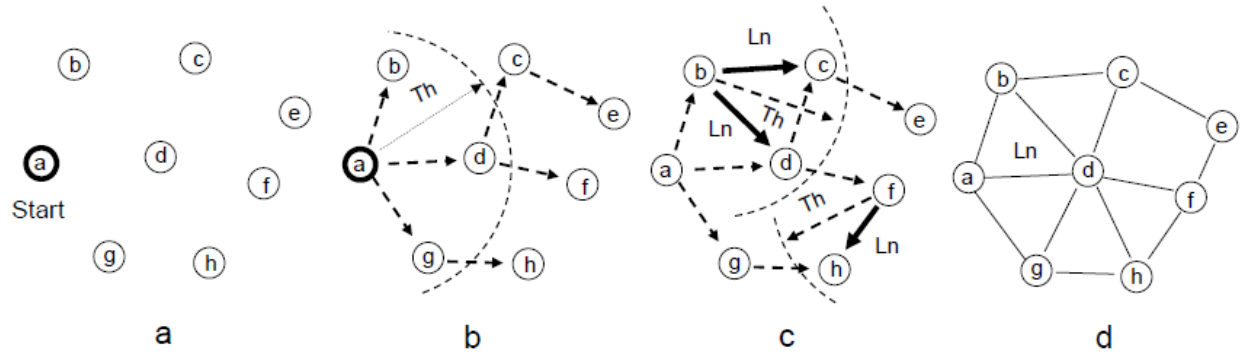

Figure 4. Distributed network creation.

Let us start in some node, let it be "a" as in Figure 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 Figure 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 Figure 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 Figure 4(d) with all established links-channels of the type "Ln", as was required.

Finding and Collecting Any Path Between Nodes

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

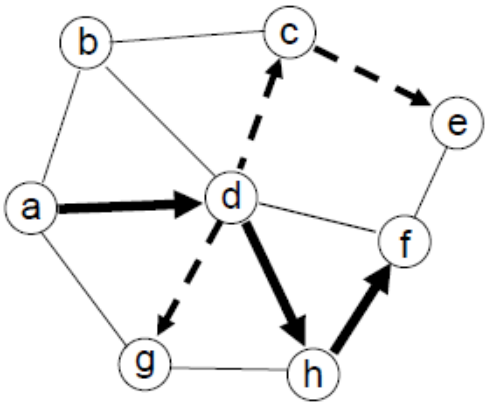

Figure 5. Reaching a node from another node.

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 Figure 5) is printed in node "f" (like a, d, h, and 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))

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 a 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 Figure 6:

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

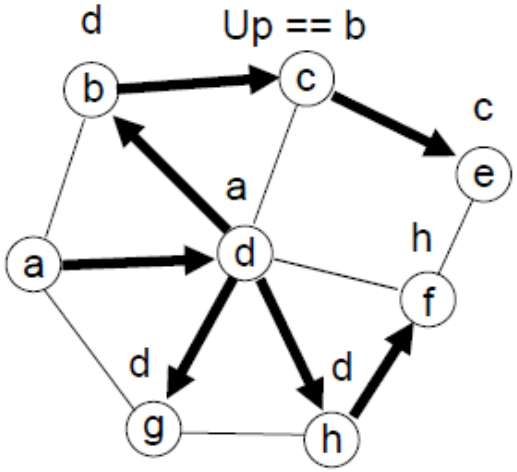

Figure 6. Creating any Spanning Tree from a node to all other nodes.

Such a tree can directly guide movement from any nodes, 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 as (e, c, b, d, and 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 as (a, d, b, c, and e):

hop (e); Path = NAME;

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

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 Figure 7 (which, in general, may be one of a number of such SPTs).

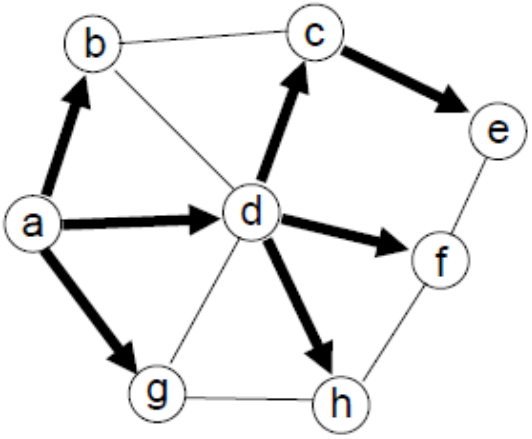

Figure 7. Creating shortest path tree from a node to all other nodes.

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)

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 Figure 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))

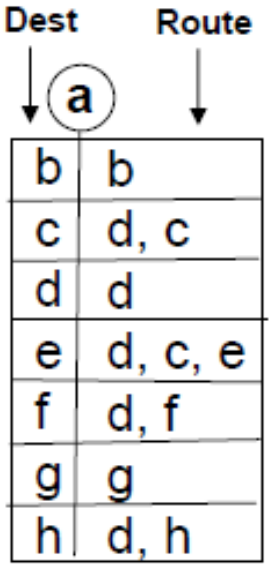

Figure 8. Registering in the SPT root node the shortest paths to all other nodes.

Using collected shortest paths in the SPT root node, any other nodes can be conveniently reached, 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 Figure 9 where vector Dest keeps names of all destination nodes in relation to the current node (similar to Figure 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).

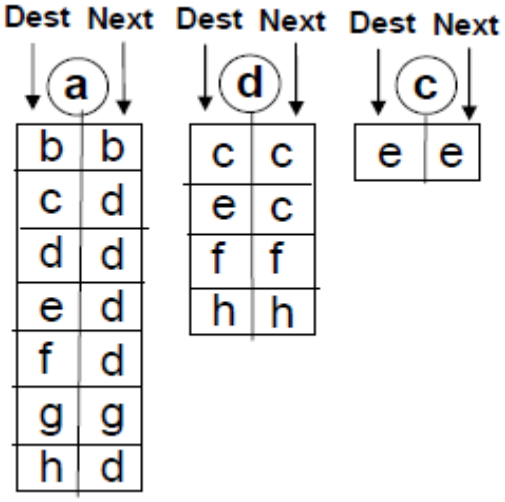

Figure 9. Routing tables in nodes for SPT in Figure 7.

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)])))

It 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.

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 Figure 10.

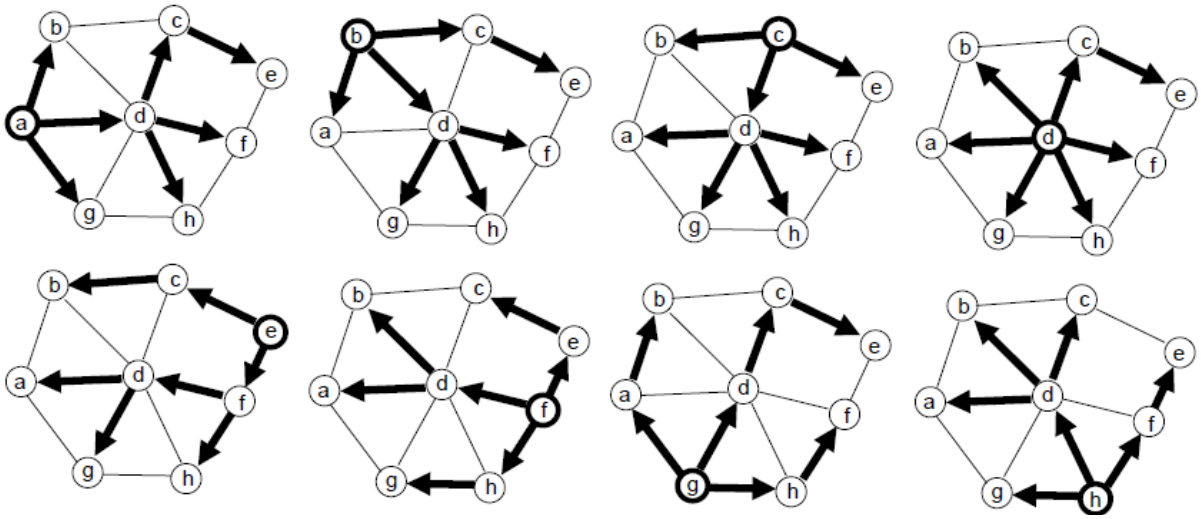

Figure 10. SPTs from all nodes of the network to all other nodes.

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

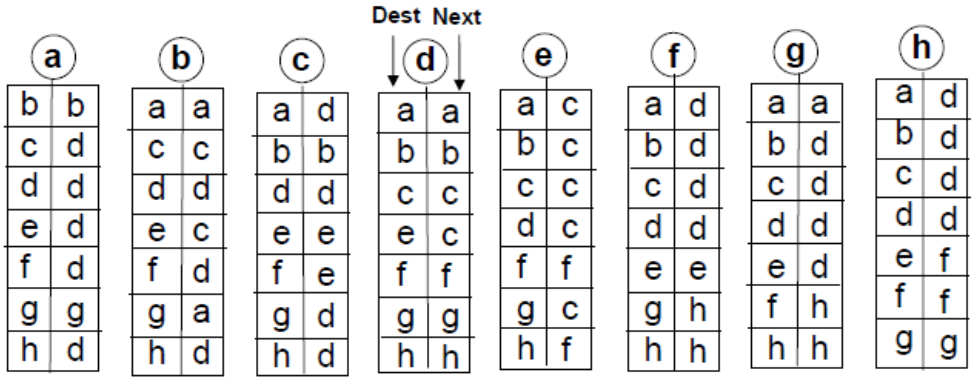

Figure 11. Routing tables providing shortest paths from all nodes to all other nodes of the network.

An example of following SP from any nodes say “g”, to any other nodes, like “e”, via the RTs of Figure 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 Figure 11 in nodes g, d, and c.

Distributed Operations on Social Networks

A great variety of operations on social networks under SGT can be found investigated and published in (Sapaty, 2018). We will be showing below only a few, which may especially relate to dealing with conflicts and crises throughout the new world organizational orders.

Finding Strongest Sub-networks, or Cliques

We present here a universal solution in SGL for finding all cliques in a network (i.e., maximum possible full subgraphs), assuming their number of nodes should not be lower than some threshold. Cliques, as traditionally considered strongest parts of the networks, are of great interest in the graph theory, and they may

also have practical importance for analysing social systems and conflicts in them, whether in national or international scale. The following scenario is finding all cliques in parallel in the network shown in Figure 12, with the number of nodes in them not less than three (with the number two, we just have a couple of nodes connected with a link).

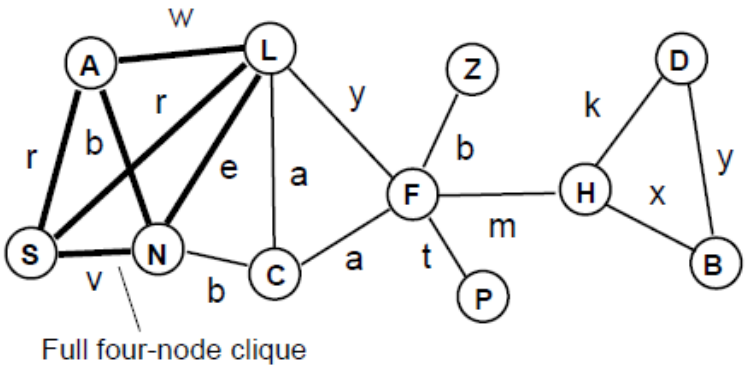

Figure 12. Finding strongest parts, or cliques, in the network.

```
hop_nodes (all); frontal (Clique) = NAME;
repeat (hop_links (all); not_belong (NAME, Clique);
        yes (and_parallel (hop (links (any), nodes (Clique)));
        if (PREDECESSOR > NAME, append (Clique, NAME), blind));
count (Clique) >= 3; output (Clique)
```

This scenario, starting in all nodes and following their links to other nodes in parallel, is collecting node names in new individual hops which have links with all previously collected nodes unless such nodes cannot be found, declaring the collected set of node names in the frontal variable Clique as a new clique. As the same clique can be collected in such a way when started from all nodes of the same clique, the duplicates are blocked by allowing inclusion of any new node into the clique's list only if the value of its name is lower than of the previous one. Also in the end, by calculating the number of elements in the collected clique's list, the clique is issued only if this number satisfies the given threshold. Many independent branches each trying to collect its own clique will be developing in the distributed network space in parallel. All cliques will be found in the network by the above scenario (only one highlighted in Figure 12 having four nodes, while all others being triangles). The following cliques will be issued in the nodes where their collection terminated:

(A, L, N, S), (C, L, N), (C, F, L), (B, D, H)

Discovering Weakest or Articulation Points

There are two articulation points F and H for the network considered before, which are highlighted in Figure 13.

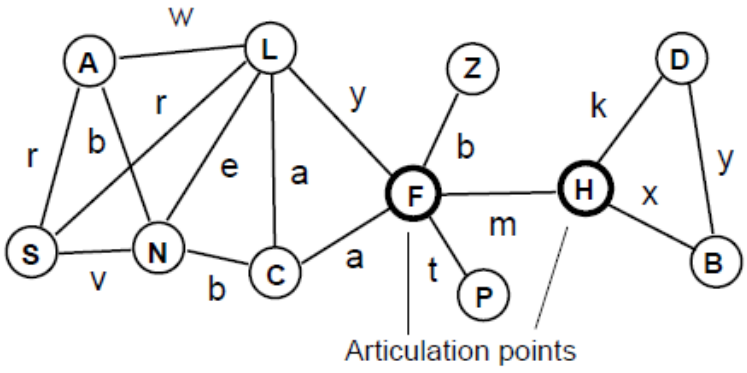

Figure 13. Weakest or articulation points of the network.

Each of them, when removed, splits the network into disjoint parts. Below is parallel and fully distributed solution for finding all articulation points in the whole network, by which each network node, having first selected one neighbour randomly, tries to navigate and mark the whole network from it while excluding itself from this process. After termination of the latter, if the node discovers still unmarked neighbours, it declares itself as articulation point and outputs its name, as follows.

```
hop_nodes (all); IDENTITY = NAME;
hopfirst_node (current);
stay (hopfirst_random(links(all));
      repeat (hopfirst_link (all)));
if (hopfirst_links (all), output (NAME))
```

Articulation points F and H will be issued by such nodes themselves.

Changing Network Structures

Many possible operations, depending on circumstances, can be carried out on social networks to change their structures in conflicts or emergency situations (see Sapaty [2018] for such procedures presented and explained in detail), with a couple of examples following.

Removing links between Clique's nodes. Let some links between clique's nodes (say, named as x or y) considered as suspicious or dangerous, and let the whole clique to be considered dangerous because of them, too. In this situation, we may be strongly weakening it by removing all links between all its nodes, as follows (in another scenario, all clique's nodes may be considered for removal, too, thus deleting the clique completely).

```
hop_nodes (all); frontal (Clique) = NAME;
repeat (hop_links(all); not_belong (NAME, Clique);
          yes (and_parallel (hop (links(any), nodes (Clique)));
          if (PREDECESSOR > NAME, append (Clique, NAME), blind));
count (Clique) >= 3;
stay, hop (links(any), nodes (Clique));
hop (links(any), nodes (Clique));
if (belong (LINK, (x, y)),
   (hop (links(any), nodes (Clique)); LINK = nil))
```

The resultant network after this operation is shown in Figure 14, with all links between all nodes of cliques (C, F, L) and (B, D, H) deleted.

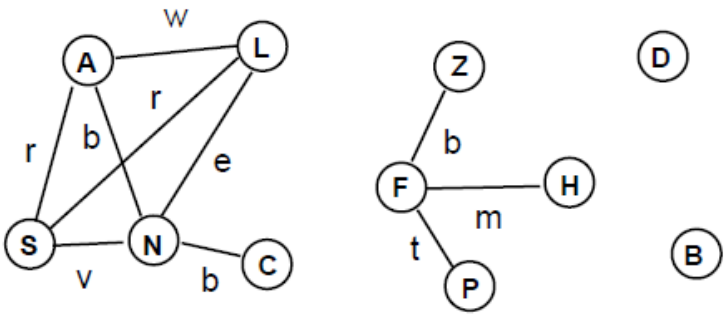

Figure 14. Removing all links between nodes of certain cliques.

Removing all articulation points. The following scenario will be removing articulation points after full completion of parallel finding of all such nodes.

```
align (
   hop_nodes (all); IDENTITY = NAME;
   hopfirst_node (current);
   stay (hopfirst_random (links(all));
         repeat (hopfirst_link(all))));
if (hopfirst_links (all), NAME = nil)
```

The resultant network will be as shown in Figure 15.

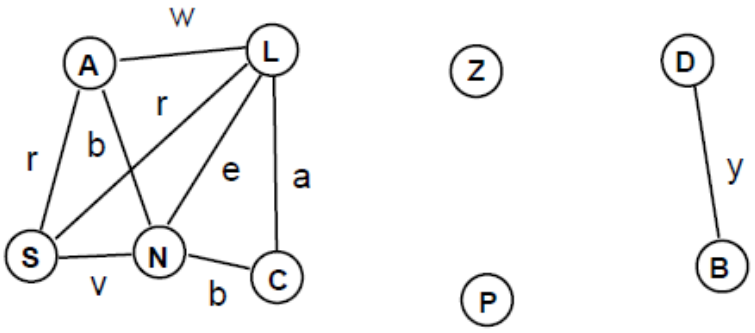

Figure 15. Removing all articulation nodes in the network.

Other social network topology changes may be oriented on their strengthening rather than weakening, as considered in (Sapaty, 2018).

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 Figure 16, different communities (which may be spread worldwide) are expressed by different types of links between their members (like c1 and c2).

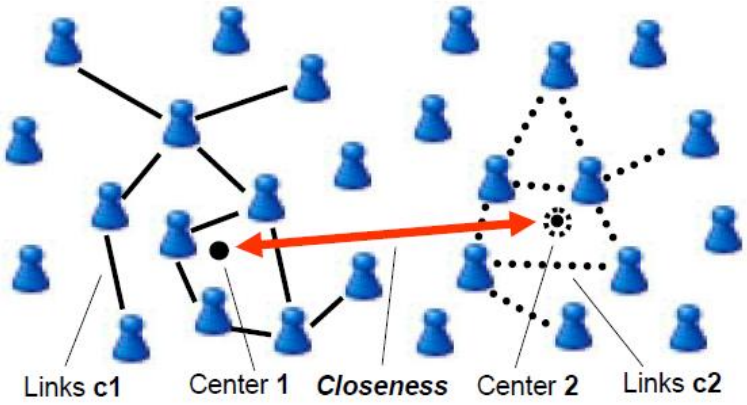

Figure 16. Finding centres of different communities

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

nodal (Center 1, Center 2, Closeness);

Center 1 = average (hop_nodes(all); yes (hop_links (c1)); WHERE);

Center 2 = average (hop_nodes (all); yes (hop_links(c2)); WHERE);

Closeness = distance (Center 1, Center 2);

output (Closeness)

The found value in Closeness may indicate existence of some unwanted trends in a multicultural society or world as a whole, say, if below a certain threshold (another example with different Closeness value is shown in Figure 17), which may need introduction and application of certain educational, organizational, or security measures.

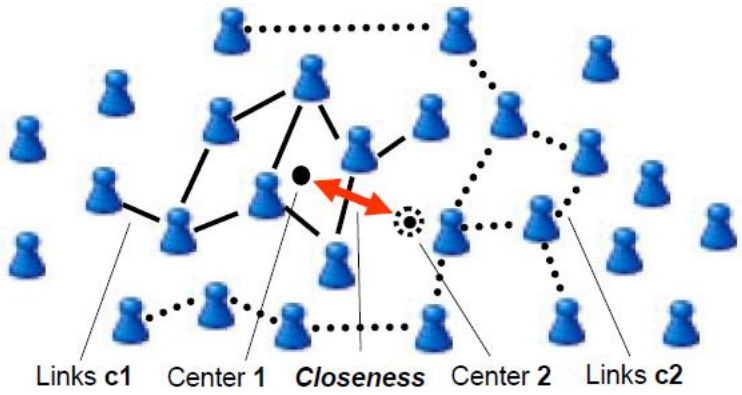

Figure 17. Communities may become too close to each other.

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 the following extended SGL scenario.

nodal (Center 1, Center 2, Closeness, Delay 1 = …, Delay 2 = …, Delay 3 =…,

Threshold = …, Maxshift1 = dX1_dY1, Maxshift2 = dX2_dY2);

branch(

```
  (hop_nodes (all); yes (hop_links (c1));
  repeat (WHERE + random (Maxshift 1); sleep (Delay 1))),
  (hop_nodes (all); yes (hop_links (c2));
  repeat (WHERE + random (Maxshift 2); sleep (Delay 2))),
repeat (
   Center1 = average (hop_nodes(all); yes (hop_links (c1)); WHERE);
   Center 2 = average (hop_nodes (all); yes (hop_links (c2)); WHERE);
   Closeness = distance (Center 1, Center 2);
   if (Closeness <= Threshold, output (Danger, Closeness));
   sleep (Delay 3)))
```

Many other tendencies and problems in distributed social systems, also in worldwide business and industrial systems (often called industrial ecosystems) can be effectively simulated and analyzed with the help of SGT (Sapaty, 2018).

Spatial Objects Discovery and Tracking

SGT can effectively simulate, trace, and control in parallel of numerous moving objects (whether technical or human) in distributed terrestrial or celestial environments. Some examples are presented and discussed in this section.

Tracing Moving Objects by Distributed Sensor Networks

Distributed sensor networks operating under SGT can catch and follow moving objects throughout the whole region despite limitations of individual sensors, as in Figure 18. The figure shows some territory covered with a heterogeneous network of communicating radar stations, each having SGL interpreter installed, with presumably hostile objects like cruise missiles moving through the area.

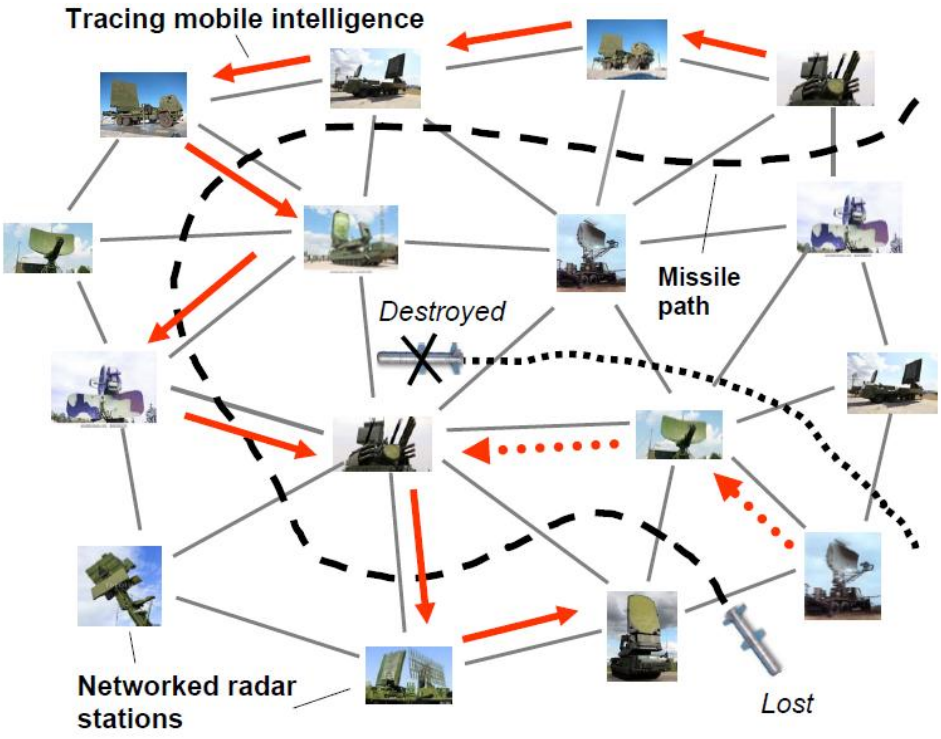

Figure 18. Distributed objects tracking by a sensor network.

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

Depending on the collected history, such object may be decided to be destroyed; it may also happen to be finally lost after safely passing through the whole radar-controlled area. The SGL scenario below will be following the moving object wherever it goes, despite its possible complex route like of a cruise missile. The scenario can operate with multiple moving objects appearing at any time, where sensors regularly search for new targets, and each new target is assigned an individual tracking intelligence propagating in distributed networked space in parallel with other similar intelligences.

```
hop (all_nodes);
frontal (Object, History, Threshold = …);
whirl (
   Object = search (aerial, new));
   visibility (Object) >= Threshold;
   free_repeat (
      loop (
         visibility (Object) >= Threshold);
         accumulate (History, Object);
         if (negative (History), blind_destroy (Object)));
```

```
max_destination (
   hop (neighbors_all); visibility (Object));
if (visibility (Object) < Threshold,
   blind_output (Object, 'lost', History))))
```

The offered organization of tracing and impacting of multiple moving objects in distributed environments by networked sensors with embedded SGL interpreters and virus-like mobile intelligence operating without any (often vulnerable) central resources can also be effectively used in many other areas, with some examples shown below.

Simulation of Space Objects Movement

The previous solution was on the level of networked sensor nodes through which spatial control was explicitly propagating and following the object's movement in physical space. In tracing movement of physical objects under SGT, we may also come to a higher level of abstraction, associating with each physical object its unique virtual copy with appropriate name, which is imaginably moving in physical space similarly to the physical object. This organization is shown in Figure 19 for possible space objects orbiting the Earth. Of course, such virtual copies should be electronically supported and exchanged by the communicating intelligent sensors (which may have terrestrial or celestial origin), but their actual movement in space (and between the sensors) can be organized implicitly, on implementation level, when virtual objects update their space coordinates by frequently "seeing" their physical origins. All this can be effectively expressed and supported by the same SGL language, allowing us to create and manage distributed systems on a variety of levels and their mixtures.

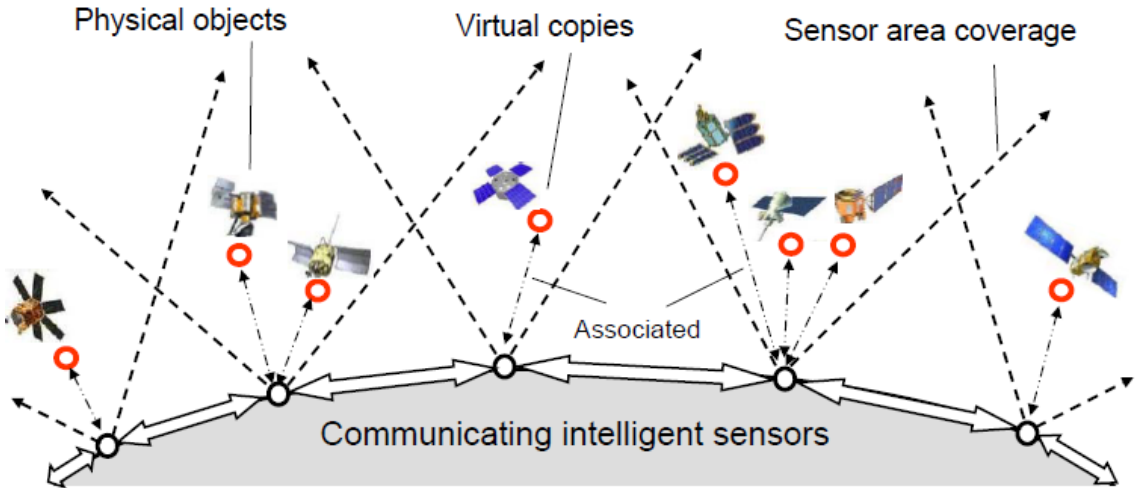

Figure 19. Simulating moving physical objects in space by their propagating virtual copies.

Creating a virtual node copy of the physical object just discovered throughout the world with its identity given, and organizing the node's continuous simulated movement in space by matching the physical object's movement, with collection and regular update of the individual object's propagation history (also making corrections in stationary databases related to the sensors passed, and in world's global space database, if needed) can be expressed in SGL as follows:

```
frontal (ObjectIdentity = …);
max_destination (
   hop (all_sensor_nodes); visibility (ObjectIdentity));
```

```
visibility (ObjectIdentity) >= Threshold;
-------------------------------------------------
create_node (ObjectIdentity); nodal (History);
loop (
   WHERE = check (coordinates, ObjectIdentity);
   History = extend (History, WHERE, TIME);
   update (DATABASE (ObjectIdentity), History);
   sleep (Delay))
```

Space Navigation With Collision Avoidance

Such created simulation objects, imaginably moving in physical space, too, can be entered and analyzed as virtual nodes by other SGL scenarios like the ones launched from manned or unmanned orbital vehicles regularly investigating the space ahead of their planned way for a possible collision avoidance in the area around their next waypoints, with resultant correction of these waypoints in case of danger, as shown in Figure 20. The speeds of the simulated objects, available from their regularly updating History, and of the vehicle itself (given in its Parameters) should be taken into account, too, which may be high for orbital objects (The area to be investigated may be covered by more than a single sensor, as shown in the figure, with entering the simulated virtual objects of interest in all of them). The SGL scenario managing the vehicle's physical movement in space by the waypoints given and with possibility of collision avoidance may be as follows:

```
frontal (Waypoints = …; Parameters = …);
nodal (Area, Next);
Repeat (
   nonempty (Waypoints);
   Next = withdraw (Waypoints); Area = area (Next, Parameters);
   Next = correct (Next, analyse_collect (hop_nodes (Area); History));
   Move (Next))
```

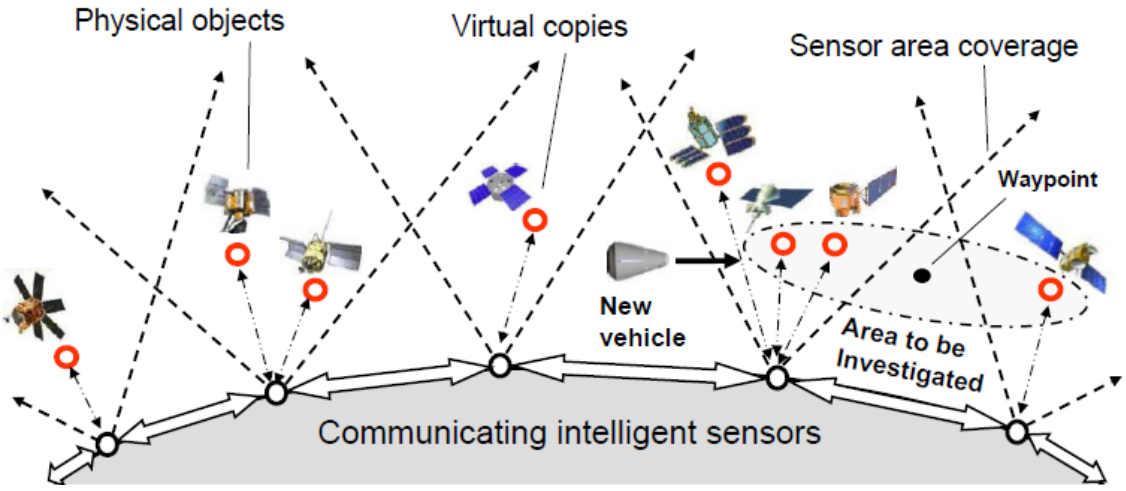

Figure 20. Investigating proper area for collision avoidance for the vehicle's next waypoints.

Managing Refugees Flow

This is another example describing movement of numerous objects (here human refugees) in a distributed environment represented in Figure 21 by a set of adjacent countries or regions. Assuming there is flow of people from certain locations into some desired destinations through the already (possibly, unofficially even illegally) established and used channels shown by oriented links.

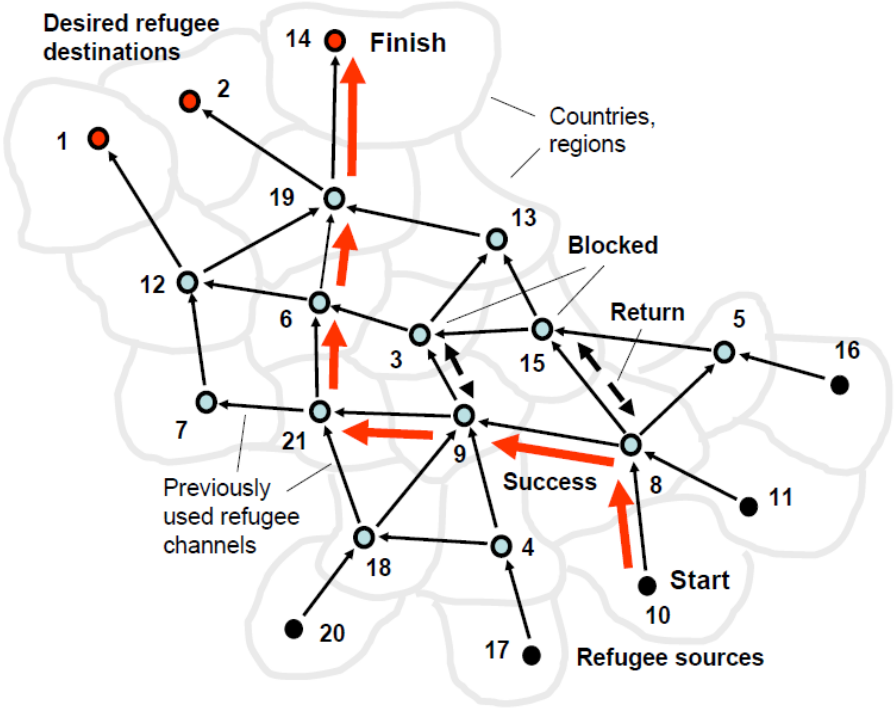

Figure 21. Simulation and management of movement of refugees to the desired destinations.

The following SGL scenario simulates movement of individuals originating from one of refugee sources (or Start) and finding next move through the oriented links if it brings the individual closest to the desired destination (or Finish) as in Figure 21. It may happen that upon arrival in a certain point an individual may find out that either everybody arriving there or she personally (with showing her identity) are not allowed to be there. In this case, the individual removes this point from the allowed visiting points in her records, returns physically to the previous point, and tries again to find among the remaining neighbors another location closest to the desired Finish, moves there, and so on. All this can be described by the following SGL scenario.

```
nodal (Start = 10);
frontal (Finish = 14, Identity = …, Next, Blocked);
move (Start);
repeat (
  Next =
   (min_destination (
      hop(links (+ all), nodes_notbelong (Blocked)); distance (WHERE, Finish)); NAME);
  if (Next == nil, blind_output ('stuck', Identity, NAME), move (Next));
  if (not_allowed (Identity), (Blocked && = NAME; move (BACK)));
```

if (NAME == Finish, blind_output ('arrived')))

This scenario can be used in numerous copies working in parallel with each other by simultaneously navigating the network of Figure 21, which may represent a real network of checkpoints at borders of different countries. This scenario can also be used for distributed simulation of the flows of numerous refugees that may appear at different times and in different locations, if to add to this model some time-bound plausible object (i.e., people) generators, also include physical distances and links propagation delays.

Conclusions

We have briefed the patented high-level networking ideology and 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 the recursive control code dynamically matching any systems. Effective expression in SGL of basic network creation and management routines, simulating and analysing structures and dynamics of social networks, also tracing, simulating, and investigating of complexly moving technical and human objects by distributed sensor networks showed the applicability of the approach offered for solving complex crisis and security problems in the emerging post-liberal world organization. SGT can really integrate the splitting world on a qualitatively new and higher level, with its controlled super-virus spatial nature technologically ignoring and penetrating through any borders and fences, while offering global solutions of problems that may originate at any world points, any moments of time, also cover large parts of the world, the whole Globe including. SGL scenarios for solving different networking problems are extremely compact which allows us to create and modify solutions for crisis and emergency problems at runtime, on the fly, also effectively use advanced collective robotics, as shown in other publications. The latest technology version can be readily installed by agreement on any platforms needed, even in traditional university environments, as was accomplished for previous SGT versions in different countries.

References

Acharya, A. (2017). After liberal hegemony: The advent of a multiplex world order, *Ethics & International Affairs, 31*(3), 271-285. Retrieved from https://www.researchgate.net/publication/319604543_After_Liberal_Hegemony_The_Advent_of_a_Multiplex_World_Order

Albrecht, F. (2017). *The social and political impact of natural disasters: Investigating attitudes and media coverage in the wake of disasters.* Retrieved from https://uu.diva-portal.org/smash/get/diva2:1090236/FULLTEXT01.pdf

Al-Dahash, H. F., Thayaparan, M., & Kulatunga, U. (2016). Understanding the terminologies: Disaster, crisis and emergency. In P. W. Chan and C. J. Neilson (Eds.), *Proceedings of the 32nd Annual ARCOM Conference* (Vol. 2, pp. 1191-1200). Manchester, UK: Association of Researchers in Construction Management. Retrieved from http://usir.salford.ac.uk/39351/

Armstrong, K. (2016). *The role of religion in today's conflict.* Retrieved from https://www.unaoc.org/repository/Armstrong_Religion_Conflict.pdf

Baraniuk, C. (2015). The disastrous events that would break the internet. *BBC Future*. Retrieved from http://www.bbc.com/future/story/20150310-how-to-break-the-internet

Blount, P. J. (2012). Targeting in outer space: Legal aspects of operational military actions in space. *Harvard Law School National Security Journal.* Retrieved from http://harvardnsj.org/2012/11/targeting-in-outer-space-legal-aspects-of-operational-military-actions-in-space/

Dawkins, P. W. (2017). *Emergency response and crisis management plan.* Retrieved from http://www.bennett.edu/wp-content/uploads/2016/06/Emergency_Response_Crisis_Management_Plan.pdf

Duncombe, T. D. (2018). After liberal world order. *International Affairs, 94*(1), 25-42. Retrieved from https://www.chathamhouse.org/sites/default/files/images/ia/INTA94_1_3_234_Duncombe_Dunne.pdf

Gabrielsen, R. H., & Lacasse, S. (Eds.). (2016). Natural disasters and societal safety. Retrieved from https://www.ntva.no/fellesrapport2015/naturaldisasters/assets/common/downloads/Natural%20Disasters%20and%20Societal%20Safety.pdf

Harari, Y. N. (2018). We need a post-liberal order now. *The Economist, Open Future*. Retrieved from https://www.economist.com/open-future/2018/09/26/we-need-a-post-liberal-order-now

Ikenberry, G. J. (2018). The end of liberal international order? *International Affairs, 94*(1), 7-23. Retrieved from https://scholar.princeton.edu/sites/default/files/gji3/files/inta94_1_2_241_ikenberry.pdf

Melnick, J. (2018). *Top 10 most common types of cyber attacks*. Retrieved from https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/

Office of the Secretary of Defense. (2019). *Missile defense review.* Retrieved from https://www.defense.gov/Portals/1/Interactive/2018/11-2019-Missile-Defense-Review/The%202019%20MDR_Executive%20Summary.pdf

Pabst, A. (2017). *A post-liberal world? Constructive alternatives to liberal globalisation amid the threat of neo-fascism.* Retrieved from https://doc-research.org/2017/05/post-liberal-world/

Sapaty, P. (1993). A distributed processing system. *European Patent No. 0389655*. Retrieved from https://data.epo.org/gpi/EP0389655A1-A-distributed-processing-system

Sapaty, P. (1999). *Mobile processing in distributed and open environments*. New York: John Wiley & Sons.

Sapaty, P. (2005). *Ruling distributed dynamic worlds*. New York: John Wiley & Sons.

Sapaty, P. (2017). *Managing distributed dynamic systems with spatial grasp technology*. New York: Springer.

Sapaty, P. (2018). *Holistic analysis and management of distributed social systems*. New York: Springer.

Sapaty, P. S. (2018). Holistic spatial management of international security. *Austin Journal of Robotics & Automation, 4*(1). Retrieved from http://austinpublishinggroup.com/robotics-automation/online-first.php

The Institute for Economics and Peace (IEP). (2016). *Global terrorism index 2016*. Retrieved from http://economicsandpeace.org/wp-content/uploads/2016/11/Global-Terrorism-Index-2016.2.pdf