BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Journal of Computer Science Research
https://ojs.bilpublishing.com/index.php/jcs

ARTICLE

Spatial Management of Distributed Social Systems

Peter Simon Sapaty*

Institute of Mathematical Machines and Systems, National Academy of Sciences, Glushkova Ave 42, 03187, Kiev Ukraine

ARTICLE INFO

Article history
Received: 29 June 2020
Accepted: 16 July 2020
Published Online: 30 July 2020

Keywords:
Social systems
Social networks
Parallel and distributed computing
Spatial Grasp Technology
Spatial Grasp Language
Holistic solutions

ABSTRACT

The paper describes the use of invented, developed, and tested in different countries of the high-level spatial grasp model and technology capable of solving important problems in large social systems, which may be represented as dynamic, self-evolving and distributed social networks. The approach allows us to find important solutions on a holistic level by spatial navigation and parallel pattern matching of social networks with active self-propagating scenarios represented in a special recursive language. This approach effectively hides inside the distributed and networked language implementation traditional system management routines, often providing hundreds of times shorter and simpler high-level solution code. The paper highlights the demands to efficient simulation of social systems, briefs the technology used, and provides some programming examples for solutions of practical problems.

1. Introduction

Social systems and social networks expressing them may be large and complex [1-4]. Consisting of numerous nodes and links between them and constantly changing their volume and structure, they may cover separate countries and the whole world. Proper dealing with such systems is crucial for the world's security and prosperity. This needs their detailed simulation and integration with live control and management within united concepts of virtual, physical and executive worlds. The current paper, based on previous publications [5-11], provides advanced approach for analysis of large social systems, which may have effective implementation on different platforms, also with the use of existing media systems and channels. The rest of this paper is organized as follows. In section 2, main demands to the simulation and support of distributed social systems are listed. Section 3 briefs the developed Spatial Grasp Technology allowing us to effectively deal with very large social networks, which may have worldwide distribution, including its high-level recursive Spatial Grasp Language (SGL) and organization of its networked interpreter. Section 4 provides examples of using SGL for describing very practical social problems, and Section 5 concludes the paper.

2. Demands to Social Systems Simulation and Support

Traditional centralized access, copying, and visualization

**Corresponding Author:*
Peter Simon Sapaty,
Institute of Mathematical Machines and Systems, National Academy of Sciences, Glushkova Ave 42, 03187, Kiev Ukraine;
Email: peter.sapaty@gmail.com

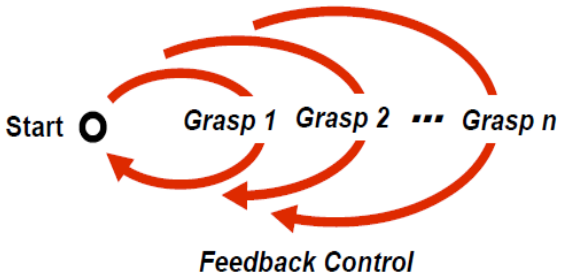
 DOI: https://doi.org/10.30564/jcsr.v2i3.2077

of social networks may not satisfy the needs as requiring unacceptable amount of time, storage, and computing resources, and the obtained network snapshots of these highly dynamic worlds are rapidly becoming outdated. The really suitable solutions may be achieved by massive and parallel and runtime dealing with social networks directly in points where the original information occurs. Within this context, we will be using the patented high-level Spatial Grasp Technology (SGT) already tested on different networked applications and described in Wiley, Springer and Emerald books [5-7, 9, 10], which allows us to find holistic solutions in large social systems by treating the whole distributed world as an integral spatial brain.

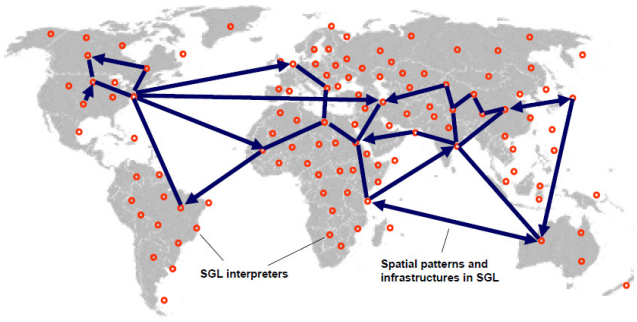
3. Spatial Grasp Technology Basics

3.1 General SGT Idea

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 sequential or parallel program. This pattern, expressing direct world vision, perception and top semantics of the problem to be solved, is written in a high-level Spatial Grasp Language (SGL). Starting from any world point (which may be multiple and arbitrarily distributed) it *spatially self-propagates, self-replicates, self-modifies, self-covers and self-matches* the distributed world in parallel wavelike mode. If needed, it also echoes back the reached control states and data discovered or obtained (which may happen to be arbitrarily remote, say, half-world away) for making decisions at higher levels and further space navigation from the reached positions, which may include the starting and any previous ones (see Figure 1,a). The self-spreading & self-matching SGL patterns-scenarios can dynamically create and leave any *knowledge infrastructures* arbitrarily distributed between system components which may cover any regions, the whole world including, as in Figure 1,b.



a



b

Figure 1. Controlled navigation of distributed spaces with creation of distributed infrastructures

Arbitrary number of spatial processes in SGL can start any time and in any places, cooperating or competing with each other, and these spatial processes can be organized on certain agreements (as in the past for the previous technology version, WAVE, installed at different universities of Germany, UK, US, and Canada [9-10]), or represent specific stealth solutions for particular purposes, depending on applications. The created infrastructures, which may remain active and capable of evolving further at any time (including self-launching new spatial SGL scenarios) can effectively support or express distributed databases, advanced command and control, situation awareness, autonomous and collective decisions. They can express or mimic any existing or hypothetical computational and/or control models, effectively integrate distributed simulation and real control with runtime changing watershed in between, and even provide a sort of self-consciousness for highly intelligent and arbitrarily distributed systems.

3.2 Spatial Grasp Language

General SGL organization is as follows (with full syntax just on a single page, see [5-7]), where syntactic categories are shown in italics, vertical bar separates alternatives, parts in braces indicate zero or more repetitions with a delimiter at the right, if multiple, and constructs in brackets may be optional:

grasp → *constant* | *variable* | [*rule*] [({ *grasp,*})]

constant → *information* | *matter* | *custom* | *special* | *grasp*

variable → *global* | *heritable* | *frontal* | *nodal* | *environmental*

rule → *type* | *usage* | *movement* | *creation* | *echoing* | *verification* | *assignment* | *advancement* |

branching | *transference* | *exchange* | *timing* | *qualifying* | *grasp*

An SGL scenario, called *grasp*, applied in some

DOI: https://doi.org/10.30564/jcsr.v2i3.2077

point (or points) of the distributed space, can just be a *constant*, a *variable*, and can also be a *rule* (expressing certain action, control, description or context) optionally accompanied with operands separated by comma (if multiple) and embraced in parentheses. These operands can be of any nature and complexity (including arbitrary scenarios themselves) and defined recursively as *grasp,* i.e. can be constants, variables or any rules with operands (i.e. as grasps again), and so on. Rules, starting in some world point, can organize navigation of the world sequentially, in parallel, or any combinations. They can result in staying in the same application point (which can also serve as starting point for further navigation) or can cause movement to other world points with obtained results to be left there, as in the rule's final points, form which to proceed further. Such results can also be returned to the rule's starting point, from which the rest of the scenario, if any, can develop. The rules, due to recursive language organization, can form arbitrary operational and control infrastructures covering any spaces and environments and expressing any sequential, parallel, hierarchical, centralized, localized, mixed, and up to fully decentralized and distributed algorithms. SGL may be considered as pursuing a quite different philosophy, methodology, and programming styles, serving as language and tool for *directly dealing with distributed dynamic spaces, both virtual and physical, and not being the language for programming computers and networks,* as usual, which is totally shifted to its automatic implementation.

3.3 SGL Interpreter

The SGL interpreter [7-12] consists of a number of specialized functional processors working with and sharing specific data structures. SGL interpretation network generally serves multiple scenarios or their parallel branches simultaneously navigating the distributed world. Each interpreter can support and process multiple SGL scenario code which appears in its responsibility at different moments of time. Implanted into any distributed systems and integrated with them, the interpretation network (having potentially millions to billions of communicating interpreter copies) allows us to form dynamic and ubiquitous world computer (actually as "spatial brain") with unlimited power for simulation and management of the world itself. Different interpreter copies appear to be dynamically interlinked by *spatial hierarchical track system* which is the result of wavelike navigation of distributed environments in SGL, and this track system effectively supports the overall management and control of highly parallel and fully distributed solutions. This internal system, working in alternating top-down and bottom-up modes, also serves as automatically created and hidden from the user powerful spatial computational, decision-making and distributed knowledge transferring, supporting, and cleaning engine. This allows us to write global SGL scenarios expressing only top semantics of the tasks to be solved, which are often hundreds of times shorter and simpler than under any other approaches for solving similar problems. SGL interpreter can have both software and hardware efficient implementations, with new patent being prepared on it. It can also be deeply integrated with any other existing networking systems and engines, thus deeply penetrating into the distributed social tissue and becoming an inseparable and intelligent part of it.

4. Some Programming Examples

We will consider two very simple programming examples in SGL related to this paper, as follows.

(1) Finding distance between averaged centers of different communities

This example is shown in Figure 2, where different communities in a social network are defined by different type of semantic links between their nodes (like c1 and c2), and such communities may semantically and spatially intersect. After finding topographical centers of communities by the following SGL scenario, if communities are located too close to each other, an "alarm" is issued (say, in case they may be antagonistic to each other).

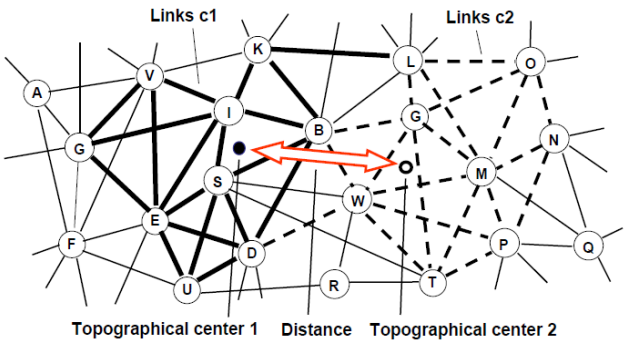


Figure 2. Finding topographical centers and distance between them in a distributed social network

```
nodal(Center1, Center2, Threshold = …);
Center1 = average(hop(all); if(hop_link(c1), WHERE));
Center2 = average(hop(all); if(hop_link(c2), WHERE));
If (distance(Center1, Center2) > Threshold, output("alarm")
```

The nodes of social network may not be stationary and can change their positions in time, with accounting this by extending the scenario above as follows, with regular finding migrating topological centers and issuing corresponding alarms.

 DOI: https://doi.org/10.30564/jcsr.v2i3.2077

nodal(Center1, Center2, Shift, Threshold = …);

parallel(

(hop(all); repeat(Shift = random(dX, dY); WHERE + Shift); sleep(delay1))),

repeat(Center1 = average(hop(all); if(hop_link(c1), WHERE))

Center2 = average(hop(all); if(hop_link(c2), WHERE));

if(distance(Center1, Center2) > Threshold, output("alarm")); sleep(delay2))

In a further extension of this scenario we may allow nodes to create new links with other nodes at runtime, also lose the existing ones.

(2) Human-robotic teaming

This is symbolically depicted in Figure 3, where communicating humans and robots (all treated as "units) are randomly swarming and supposedly eliminating the discovered unwanted objects (as "targets"), also informing close neighbors (humans or robots) about the targets seen, thus prompting collective actions.

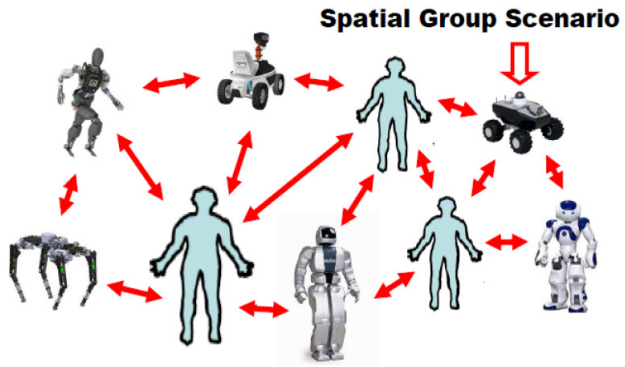


Figure 3. United human-robotic collectives

hop(all_units); repeat(

Shift = random(*dx_dy*);

if(empty(WHERE + Shift), shift(Shift));

append((own, direct_neighbors); Targets), seen(targets));

impact(targets); sleep(*delaytime*)))

This scenario can also have different extensions similar to the previous one, where human-robotic collectives may dynamically organize runtime hierarchies (with higher levels potentially occupied by robots too) improving their collective vision and cooperative fight with unwanted objects. Such human-robotic collectives can have holistic qualities and capabilities in SGL (see possible formalization of gestalt theory laws under SGT [6-7]), they can even have a sort of distributed consciousness for very complex and important applications, especially for crises management and defense.

5. Conclusion

The main advantage of the philosophy, methodology and technology developed is that it operates in both simulated and actual worlds, with feeling of direct presence and free movement in them. And all this can be expressed within the same formalism and very high level language enabling us to hide most of traditional systems management routines inside its fully distributed, parallel and intelligent implementation. This paradigm, known as WAVE in the past [9,10] has some relation to mobile agents (having appeared well before them), but it navigates and grasps distributed spaces holistically and globally, also leaving active spatial infrastructures which may cover the whole world. This is quite different from traditional agents-based and interoperability philosophies which consider the system as consisting from well defined autonomous parts which need to be interlinked and integrated by some additional means, which often does not work properly. The technology developed can also be symbolically considered as unlimitedly powerful world super-virus, which has enormous power not only to kill but also create, restructure, improve, and rule the world. One of its currently investigated applications is simulation of global pandemics and spatial methods of fighting them. Another considered application—global missile defence systems, both terrestrial and celestial, especially for withstanding very high speed dangerous objects which may have tricky routes. Concerning social networks, the tech offered can effectively combine distributed interactive simulation of large social systems with their effective management, with watershed between the two regulated at runtime within the symbiotic simulation-control SGL scenarios. Other investigated applications include advanced mosaic-type operations in distributed systems, simulation of such complex features as awareness and consciousness, also technological support of space conquest and advanced terrestrial and celestial missions. SGL can be quickly implemented even within standard university environments, similar to its previous versions in different countries under the author's supervision.

References

[1] Baraldi, C., Corsi, G.. Social Systems Theory. in: N. Luhmann. Springer Briefs in Education. Springer, Cham, 2017.

[2] Ghoshal, G., Mangioni, G., Menezes R. et al. Social System as Complex Networks. Social Network Analysis and Mining, 2014, 4: 238.

[3] Denny, M. Social Network Analysis, Institute for Social Science Research, University of Massachusetts

DOI: https://doi.org/10.30564/jcsr.v2i3.2077

Amherst, 2014.

[4] Mangal, V., Gadh, V. Systems Theory and Social Networking: Investigation of Systems Theory principles in Web 2.0 Social Network Systems. International Journal of Business and Commerce, 2013, 3.

[5] Sapaty, P. Complexity in International Security: A Holistic Spatial Approach, Emerald Publishing, 2019.

[6] Sapaty, P. Holistic Analysis and Management of Distributed Social Systems, Springer, 2018.

[7] Sapaty, P. Managing Distributed Dynamic Systems with Spatial Grasp Technology, Springer, 2017.

[8] Sapaty, P. Distributed Human Terrain Operations for Solving National and International Problems", International Relations and Diplomacy, 2014, 2(9).

[9] Sapaty, P. Ruling Distributed Dynamic Worlds. John Wiley & Sons, New York, 2005.

[10] Sapaty, P. Mobile Processing in Distributed and Open Environments, John Wiley & Sons, New York, 1999.

[11] Sapaty, P. A distributed processing system, European Patent No. 0389655, Publ. 10.11.93, European Patent Office, 1993.

 DOI: https://doi.org/10.30564/jcsr.v2i3.2077