Acta Scientific COMPUTER SCIENCES

Volume 3 Issue 9 September 2021

Review Article

Spatial Grasp Model for Management of Dynamic Distributed Systems

Peter Simon Sapaty*

Institute of Mathematical Machines and Systems National Academy of Sciences, Kiev, Ukraine

***Corresponding Author:** Peter Simon Sapaty, Institute of Mathematical Machines and Systems National Academy of Sciences, Kiev, Ukraine.

Received: July 07, 2021

Published: August 31, 2021

Abstract

More complex distributed and intelligent systems are being developed covering both terrestrial and celestial environments, which relate to economy, ecology, communications, security, and defense. Their efficient management, especially in dynamic and unpredictable situations, needs serious investigations and development, evenbreakthroughs, in scientific and technological areas. Their traditional representations as parts operating by certain algorithms and exchanging messages are becoming inadequate as such systems need much stronger integration in order to operate as holistic organisms pursuing global and often varying goals. The current paper is just oriented on acompletely different paradigm for organization and management of large dynamic and distributed systems, which extends and transforms the notion of algorithm for not only describing knowledge processing logic but also allowing it to directly exist, propagate, and operate as an integral whole in any distributed spaces, which may be constantly changing their volumes and structures. Having some organizational features related to powerful viruses, recent pandemics too, this ubiquitous Spatial Grasp (SG) model is presented in the paper on philosophical and implementation levels, with introduction of special spatio-charts for its exhibition and studies, which are extending traditional algorithmic flowcharts towards working directly in distributed spaces. Using this model for the creation ofresultant Spatial Grasp Technology and its basic Spatial Grasp Language, already described in detail in numerous publications, is also briefed. Elementary examples of dealing with distributed networks, collective human-robotic behavior, and removal of space debris by constellation of cleaning satellites explain SG advantages over traditional system organizations.

Keywords: Algorithm; Flowchart; Distributed Systems; Spatial Grasp; Spatio-chart; Holistic Solutions; Network Management; Collective Behavior; Space Debris

1. Introduction

More and more complex distributed and intelligent systems are being developed worldwide covering both terrestrial and celestial environments, as well as their integration, which relate to economy, ecology, communications, security, defense, and many other areas. Their efficient management, especially in dynamic, unpredictable, and crises-prone situations, needs serious investigations and development, often breakthroughs, in political, scientific, technological, and industrial areas. Their traditional representations as parts or agents operating by certain individual algorithms and exchanging messages with other parts are becoming inefficient and inadequate, as such systems need much stronger integration in order to operate as super-summative holistic organisms pursuing global philosophy and rapidly changing goals. The current paper is just oriented on a completely different paradigm for organization and management of large dynamic and distributed systems, which extends and transforms the notion of algorithm for not only describing knowledge processing logic but also allowing it to directly

exist, propagate, and operate as an integral whole in any distributed spaces. Having some organizational, self-spreading, self-replicating and self- recovering features in some sense resembling powerful world viruses, recent pandemics too, this ubiquitous Spatial Grasp (SG) model is presented in the paper on philosophical, methodological, and implementation levels. The rest of the paper is organized as follows.

Section 2 describes basics of Spatial Grasp model and how it differs from conventional algorithm, also introduces new type of a chart called spatio-chart as a further development of traditional flowcharts for describing and analyzing scenarios operating directly in distributed spaces. It shows how collections of actions can be described in SG and exhibited by spatio-charts, including the use of control rules supervising repetition, sequencing and branching in the spatial scenarios, with such organizations capable of being unlimitedly hierarchical and recursive.

Section 3 briefs the main elements of Spatial Grasp Technology (SGT) and its high-level recursive Spatial Grasp Language (SGL) based on SG philosophy, with already existing numerous publications, books including, on this approach, its implementation and numerous applications. This includes different types of distributed worlds GGT operates with, types of spatial variables of SGL, some of which may be stationary while others mobile, main types of SGL rules which can be nested, different control states provided by SGL scenarios propagation, and general organization of the distributed and networked SGL interpreter.

Section 4 provides examples of solving different distributed problems under SG model which confirm its advantages in comparison with traditional parallel and distributed system organizations. These include network management with finding and collecting a path between different nodes with printing it at the final or starting node, organizing collective behavior of a mixed human-robotic team, and also using large constellation of debris-cleaning satellites operating together under the global goal. In all these examples the solutions are provided by spatial SGL scenarios self-evolving and matching distributed dynamic spaces, which even cannot be classified as systems in advance of these solutions, and should be absolutely needed by any other approaches.

Section 5 concludes the paper summarizing obtained results and mentioning the new activities planned in this area, including new SGT implementation, new patenting of the SG model, and also new book on using SG model and the resultant technology for management of integrated terrestrial and celestial systems, which is currently in preparation.

2. Spatial grasp versus traditional algorithm

2.1 Algorithm and flowchart

Algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of specific problems or to perform a computation [1-3]. Algorithms are always unambiguous and are used as specifications for performing calculations, data processing, automated reasoning, and other tasks. In contrast, aheuristic is a technique used in problem solving that uses practical methods and/or various estimates in order to produce solutions that may not be optimal but are sufficient given the circumstances [4].

A flowchart is a type of diagram that represents a workflow or process [5,6]. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. An example of a simple flowchart is in figure 1 (where a processing step is usually depicted as a rectangular box and a decision as a diamond).

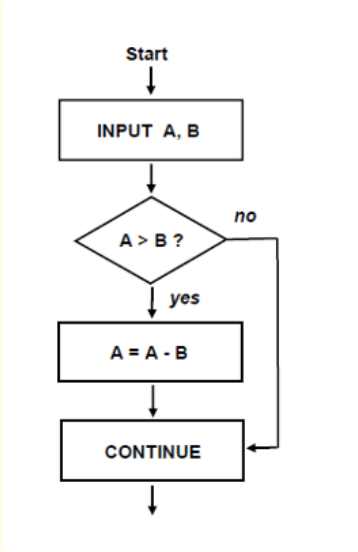

Figure 1: Example of a flowchart.

2.2 The simplest spatial grasp explanation

SG [7-13] operates by spatial scenarios self-spreading in physical and virtual worlds while creating, matching, transforming and managing them. The interpreted scenario text is not staying in any fixed point or points in space but rather spreads itself while carrying further its remainder and often omitting the utilized parts.

Single operation

Imagine you are staying in some point of space (which may be a paper sheet, computer memory, or any terrestrial or celestial environment) and just writing:

44.55

You will get this value in this point which may stay there indefinitely and without any name. Another example:

5 + 6.

This operation will produce the value 11 which will also stay in this point without a name too. One more:

R = 5 + 6.

The result 11 will be assigned to a variable R and will stay there under this name. It may be subsequently accessed by the name R if to come into this point again. Other example, but now related to physical space: move(x55, y88).

From the starting point in space, you will move to another point having certain x-y coordinates, and will stay there. If you want to create a node named John in the virtual space, with staying indefinitely in this node, just write:

create('John').

If the node John already exists, you may directly hop into it from the starting point, as follows.

hop('John').

A single action may produces a multiple result, for example, by hopping simultaneously to virtual nodes John,Peter, and Alex, if they already exist, as:

hop('John', 'Peter, 'Alex').

Or moving in parallel to a number of physical world locations from a starting point, as follows:

move((x55, y88), (x5, y12), (x105, y92)).

Generalizing the above mentioned and other possible examples with a single action, let the latter be named as g and applied in some Start point, we can receive the result in some region of space (consisting of computational,virtual, physical or combined points), symbolically named (but in capital) as G too, which may include the Start position, as shown in figure 2.

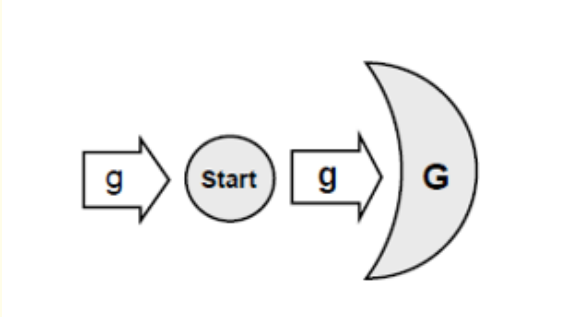

Figure 2: The simplest spatio-chart.

We will be using such space processing and navigation charts in the subsequent examples too, calling such form of exhibition as spatio-chart (or just spatiochart) versus traditional flowchart.

Multiple operations

Let us consider now a sequence of possible operations in space, with using semicolon as a delimiter between them. Assigning to a variable R and then changing its value, with finally staying in the same position in space, can be as follows:

R = 15; R = R + 10.

Hopping to virtual node John and then creating a new node Peter with relation to it from John as of his

father, with final staying in node Peter:

hop('John'); create(link('father'), node('Peter')).

Hopping to virtual node Peter and then creating a friendrelation to the already existing node John, with final staying in node John:

hop('Peter'); linkup('friend', node('John')).

Moving to a physical location by its absolute coordinates and then twice shifting to other locations by given coordinate changes, with final staying in the node reached by the second shift:

move(x55, y88); shift(x11, y22); shift(x9, y45).

The mentioned and any other examples with sequences of actions giapplied from a Startposition, with the next action gi + 1 originating in all or some space positions reached by the previous action gi, can be represented just as:

g1; g2; g3.

Their collective operation is also shown as a spatio-chart in figure 3 with regions reached by actions ginamed as Gi (which may generally include positions covered by the previous actions, the Startincluding). This also takes into account that the rest of the sequence has to propagate in space for delivering descriptions of further actions, and the already used operations are removed from the sequence.

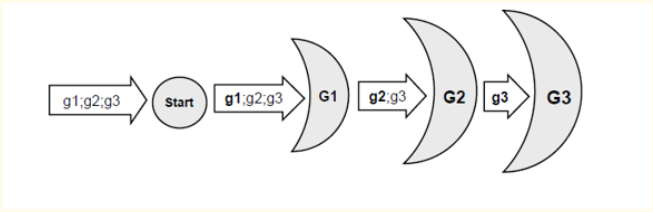

Figure 3: Spatio-chart for a sequence of actions in space.

In figure 4, detailing the sequence of operations of figure 3, it is shown that the movement from regions Gi to Gi+1 canbe made in parallel from different points of Gi (from the same points potentially too, as gi may themselves represent parallel operations), so the operational sequences in reality can be replicated at any stage of their development.

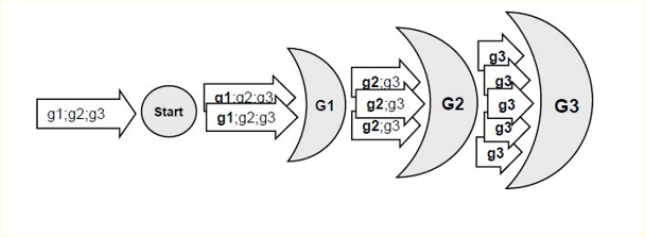

Figure 4: Possible code replications during parallel space navigation.

2.3 Using control rules

For sequencing

In a more advanced organization of the sequence of operations we may use different control rules embracing them, which can provide additional, often nonlocal, functionality and more advanced processing and coverage of distributed spaces, as follows for rule r1and the operational sequence considered before:

r1(g1; g2; g3).

The rule will be activated in the position where the whole sequence is applied, like Start as before. It then may influence the whole sequence of embraced operations with receiving a feedback from its entire development (if such feedback needed by the rule's functionality), as shown in figure 5.

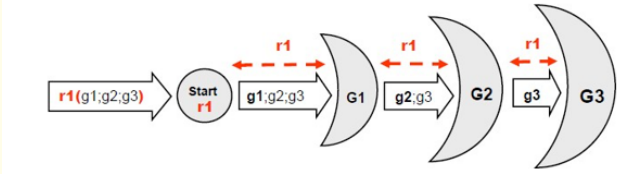

Figure 5: Spatio-chart with a control rtule.

The rule, for example, may represent such functionality as print, create, repeat and many other cases of nonlocal management and control. Let us consider a few examples in more detail.

r1➔ print.

By this rule, the results obtained by the mentioned sequence of operations (and not only by its last operation g3, but also by g1 and g2 classified as final results too) can be returned to the Start position and printed there, with final staying in the Start.

r1 ➔ create.

This rule can supply the space propagating operations, especially those describing movement in virtual spaces, with global power of creating these spaces or their individual elements if they are absent during this movement and therefore do not allow it to proceed further. This means that the same written sequence of actions gi can work in both space navigation and space creation modes, depending on circumstances.

r1 ➔repeat.

Under this rule, the sequence of operations gi at first is processed as usual, step by step, until the rest of it becomes empty, but after this, it starts to work from the very beginning again, as shown in figure 6. The repeat rule always leaves the already processed operations in their sequence (which are not removed as without it), with the whole sequence repeatedly propagating and working in space until this is possible.

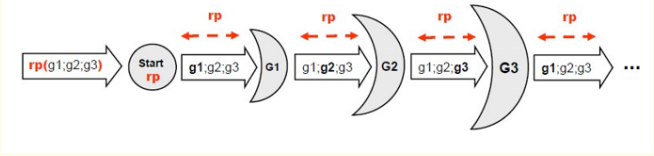

Figure 6: Repeated navigation of space.

The operational sequence can be embraced by any number of control rules, which can be nested, as follows for rules r1and r2 and also shown in figure 7 (with r1activated in the Start position, and r2 in the positions in space G1(which may be more than one) reached by operation g1.

r1(g1; r2(g2; g3)).

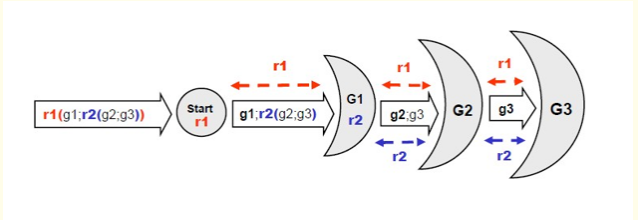

Figure 7: Nested control rules.

A few examples of combination of such nested rules as follows:

r1 ➔ create r2 ➔ repeat

r1 ➔ print r2 ➔ create

r1 ➔ repeat r2 ➔ repeat

r1 ➔ repeat r2 ➔ print

For the last case, printing the results obtained by actions g2 and g3 will be organized in all positions of the region G1 reached by g1 (which, moreover, can be repeated by rule r1 at the higher level) and not in Start as for r1in the second case).

For branching

Other rules may allow branching in space, with different branches (separated by comma) developing from the same positions in space (as follows and also shown in figure 8, where r1

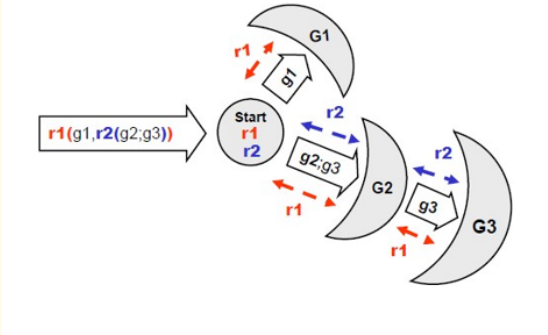

Figure 8: Branching under control rule.

is used to control two branches, and r2 to manage the sequence of operations belonging to the second branch).

r1(g1, r2(g2; g3)).

Possible meaning of r1 and r2 may be as follows, with both rules activated in the Start position.

r1 ➔ or r2 ➔ create.

Rule r1, activating two branches in any order or in parallel from the same Start position, selects the first one in time replying with a positive termination result, considering its space locations and data obtained there as the final result, while ignoring all achievements of another branch. Rule r2 covers the sequence of operations g2 and g3 ofthe second branch, supplying them with creative power during space navigation (will be completed and accepted if classified as the resultant branch by r1).

r1 ➔ if r2 ➔ print.

Rule r1 first launches branch g1, and only if and after it terminates with a positive result, activates the second branch with g2and g3 embraced by rule r2, which organizes printing their final results in the Start position with staying there, the latter considered as holding the final result. If g1 results with failure, the final stay will be in the Start too but without any new result.

r1 ➔ and r2 ➔ repeat.

Rule r1 activates the two branches in any order or in parallel, and only after both branches reply with their final success, r1 can confirm the whole success of this mission (with r2 organizing repetitive development of the sequence of two embraced operations). The successful positions reached in space by both branches will be considered as holding the final results of the scenario, with subsequent saying in them indefinitely. If any branch replies with failure, the development of the second branch will be terminated as soon as possible, as not needed any more. After r1 fails, there will be no position to stay in space further, with Startjust abandoned.

We can also consider the development of operations g2 and g3 not in a sequence but in branching mode too, as follows, also depicted in figure 9, with both rules activated in the Start position, where r1 is embracing r2 and all operations r2 coordinates, i.e. g2 and g3.

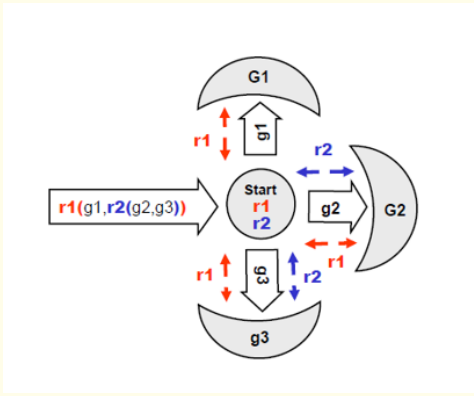

Figure 9: Nested branching.

r1(g1, r2(g2, g3)).

Possible examples of combinations of these rules, hopefully clear enough without further explanation.

r1 ➔ or r2 ➔ or

r1 ➔ if r2 ➔ and

r1 ➔ and r2 ➔ or

r1 ➔ and r2 ➔ and

2.4 Recursive scenarios

In figures 2 to 8 we had in mind that operations g1, g2, g3 could be of any complexity. This means that each one itself could be substituted by the whole scenarios shown in all these figures, for which again each operation could be substituted by the whole scenario again, and so on, with such recursion potentially to any depth. If, for example, to substitute operation g2 of the previous scenario r1(g1, r2(g2, g3)) shown in figure 8 with the scenario (g4 r3(g5, g6)) enclosed in parentheses as a whole unit, we will receive a more detailed organization with the resultant combined scenario as follows, where its spatial development is shown in figure 10:

r1(g1, r2((g4; r3(g5; g6)), g3)).

We have shown only very few examples for structuring of spatial scenarios and rules coordinating their collections, with using semicolon to separate succeeding each other operations, and comma for branches. We may also have comma as the only separator by substituting the sequences of components separated by semicolon with another rule advance covering them, with comma being se-

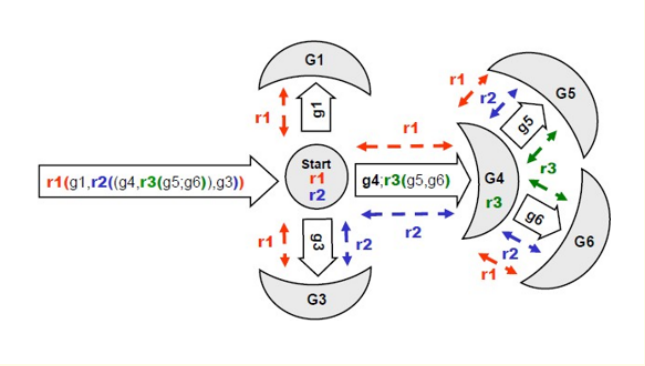

Figure 10: Spatio-chart recursion.

parator too, as follows, thus resulting in uniformity of all possible spatial scenarios.

g1; g2; g3 ➔ advance(g1, g2, g3).

This may sometimes be less convenient to write, but the rule advancein general provides additional and very useful options for sequencing of operations, as described in the full SGL.

In general, the Spatial Grasp model is much more advanced, diverse, and complex, with capability of returning the obtained results and control states whatever remote and multiple they might appear. It also allows to make any decisions for the further space navigation, creates dynamic operational infrastructures capable of solving any distributed problems, also effectively mimicking or implementing any other models and approaches (Petri nets and neural nets including), and so on.

3. Spatial grasp technology (SGT) basics

3.1 The spatial grasp language

The mentioned above and many other SG model's capabilities can be expressed by the recursive high level Spatial Grasp Language (SGL) in which all spatial scenarios are represented, with its top level syntax following. (The overall SGL scenario is called grasp, 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 are 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***

Figure a

3.2 The worlds SGT operates with

SGT allows us to directly operate with the following world representations: Physical World (PW), considered as continuous and infinite, where each point can be identified and accessed by physical coordinates; Virtual World (VW), which is discrete and consists of nodes and semantic links between them; and Executive world (EW) consisting of active "doers" with communication possibilities between them. Different kinds of combination of these worlds can also be possible within the same formalism, as follows: Virtual-Physical World (VPW) where individually named VW nodes can associate with coordinates of certain PW points or any its regions; Virtual- Execution World (VEW), where doer nodes may have special names assigned to them and semantic relations in between, similarly to pure VW nodes; Execution-Physical World (EPW) can have doer nodes associated with certain PW coordinates; and Virtual-Execution-Physical World (VEPW) combining all features of the previous cases.

3.3 SGL variables

Spatial variables, stationary or mobile, which can be used in fully distributed physical, virtual or executive environments, are effectively serving multiple cooperative processes under the unified control. These are: Global variables (most expensive), which can serve any SGL scenarios and be shared by them, also by their different branches; Heritable variables appearing within a scenario step and serving all subsequent, descendent steps; Frontal variables serving and accompanying the scenario evolution, being transferred between subsequent steps; Environmental variables allowing us to access, analyze, and possibly change different features of physical, virtual andexecutive words during their navigation; and finally, Nodal variables as a property of the world positions reached by scenarios and shared with other scenarios in same positions.

3.4 SGL rules

SGL rules, capable of representing any actions or decisions, belong to the following main categories: (a) hierarchical fusion and return of potentially remote data; (b) distributed control, sequential and/or parallel, in both breadth and depth of the scenario evolution; (c) a variety of special contexts detailing navigation in space, also clarifying character and peculiarities of the embraced operations and decisions; (d) type and sense of a value or its chosen usage for guiding automatic language interpretation; and (e) individual or massive creation, modification, or removalof nodes and connecting links in distributed knowledge networks, allowing us to effectively work with arbitrary knowledge structures. All rules are pursuing the same unified ideology and organizational scheme, as follows: (1) they start from a certain world position, being initially linked to it; (2) perform or control the needed operations in a distributed space, which may be branching, stepwise, parallel and arbitrarily complex, also local and remote; and (3) produce or supervise concluding results of the scenario embraced, expressed by control states and values in different points.

3.5 Control states

The following control states can appear after completion of different scenario steps. Indicating local progress or failure, they can be used for effective control of multiple distributed processes with proper decisions at different levels. These states are: thru – reflects full success of the current scenario branch with capability of further development; done – indicates success of the current scenario step with its planned termination; fail – indicates non- revocable failure of the current branch and no possibility of further development from the location reached; and fatal – reporting terminal failure with nonlocal effect, while triggering massive abortion of all currently evolving scenario processes and removal of associated temporary data with them. These control states, appearing in different branches of parallel and distributed scenario at bottom levels, can be used to obtain generalized control states at higher levels, up to the whole scenario, in order to make proper decisions for the further scenario evolution.

3.6 Networked SGL interpreter

Communicating Interpreters of SGL can be in arbitrary number of copies, say, up to millions and billions, which can be effectively integrated with any existing systems and communications, and their dynamic networks can represent powerful spatial engines capable of solving any problems in terrestrial and celestial environments. Such collective engines can simultaneously execute many cooperative or competitive tasks without any central resources or control, as symbolically depicted in figure 11 (SGL interpreters just named U, as universal computational and management nodes).

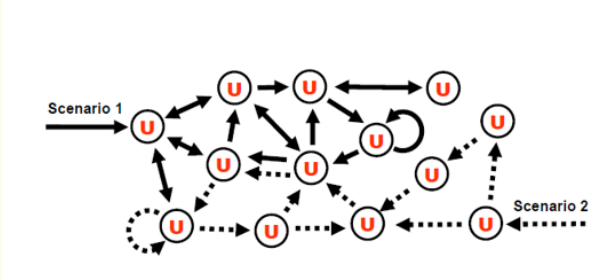

Figure 11: SGL interpretation networks as a global world computer.

As both backbone and nerve system of the distributed interpreter, its hierarchical spatial track system dynamically spans the worlds in which SGL scenarios evolve, providing automatic control of multiple distributed processes. Self-optimizing in parallel echo processes, this (generally forest-like) distributed structure provides hierarchical commandand control, also remote data and code access. It supports spatial variables and merges distributed control states for making proper decisions at different organizational levels. The track infrastructure is automatically distributed between different active components (humans, robots, computers, smartphones, satellites, etc.) during scenario spreading in distributed environments.

Detailed information on SGT, SGL and its networked interpreter, also solving numerous problems from very different classes under such approach, can be obtained from many existing publications, including [7-13], also justby spatial grasp in google.com.

4. Examples of spatial scenarios in SGL

We will show here some solutions of practical problems from different areas which are entirely based on the Spatial Grasp model described in this paper, with explanation of its advantages.

4.1 Network management

Finding any path between nodes a and f in a network of figure 12, as follows.

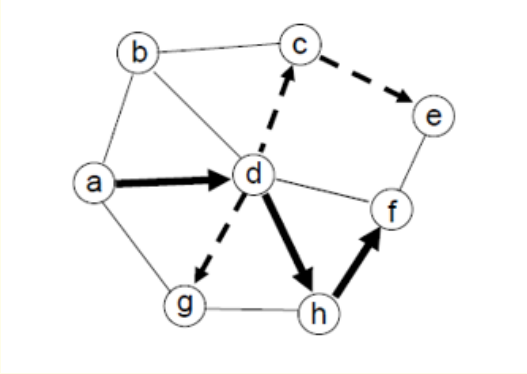

Figure 12: Finding a path between two network nodes.

```
hop(a); repeat(hopfirst(links(all));

if(NAME == f, done)).
```

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

```
hop(a); frontal(Path = NAME); repeat(hopfirst(-
link(any)); Path &&= NAME;

if(NAME == f, (output(Path); done))).
```

Arbitrary path found between these two nodes (may not be optimal like the one in Figure 12) is printed in node f like:

`(a, d, h, f)`. Such path can also be returned and issued in the starting node a as:

```
hop(a); frontal(Path = Name); output(repeat
(hopfirst(link(any)); Path &&= NAME;

if(NAME == f, done(Path)))).
```

SG solution analysis

The described solutions on a distributed network are entirely based on its spatial navigation by SGL self-evolving scenarios with finding a path between the nodes needed, also collecting and printing this path in its final or starting nodes. Such integral parallel and fully distributed solutions are much superior to traditional methods of representing distributed computations in the form of multiple parts or agents exchanging messages.

Many more on the network management under SGT can be found at [7,8,12,14].

4.2 Human-robotic collectives

We can easily organize any collectives from human, robotic or mixed units operating under spatial scenarios, as symbolically shown in figure 13.

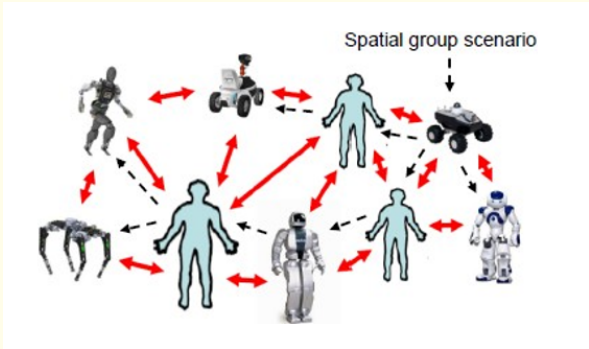

Figure 13: Unified human-robotic teams with collective scenario injected from any unit.

Randomized collective group movement, starting in any node, with minimal Range distance allowed between units when moving, can be organized as follows, where each unit by discovering some dangerous objects or situations issues a corresponding alarm message or sound.

```
hop(any_unit);
repeat(
  hop_first(all_neighbors);
  free(nodal(Limits = (dx(0,8), dy(-2,5)), Range = 100, Shift);
       repeat(Shift = random(Limits);
              if(empty(Shift, Range), WHERE += Shift);
              if(analyze(Seen), output(alarm));
              sleep(delay))))
```

SG solution analysis

This scenario starts from any human or robotic unit and then covers all reachable units by flooding them in parallel, regardless of their number which may be arbitrarily large and not known in advance. It immediately tasks each reached unit with the collective movement and observation functionality, without waiting for full completion of the scenario distribution. This holistic self-evolving spatial solution is also much superior to traditional parallel computations representing the system as a collection of agents exchanging messages.

Many more on collective behavior under SGT can be found at [7-10].

4.3 Space debris collection

Dealing with such complex problem as huge amount of space debris can be possible only by using large constellations of special cleaning (like de-orbiting) satellites working together, see figure 14. The following scenario is launched from some ground station G2 and enters any currently reachable satellite, after which floods the whole constellation by direct inter-satellite links in an attempt to find a suitable cleaner-satellite for the removal of junk initially given by its parameters detected by G2.

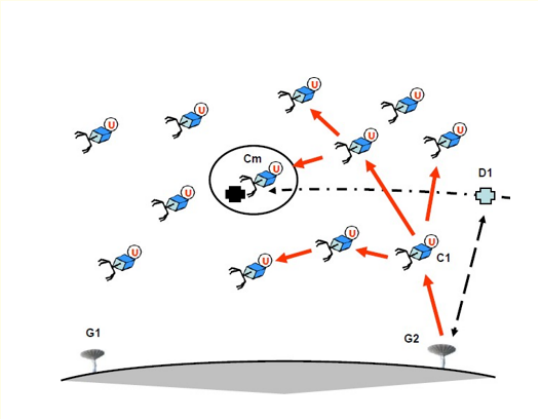

Figure 14: Debris removal by self-organized network of cleaning satellites.

```
hop(G2);
frontal(Details) = find_select(radar, junk, TIME);
hop_first(any_cleaner, radar);
repeat(
Snapshot = parameters(closest_junk, seen);
if(match(Details, Snapshot),
(deorbit(Snapshot); abort));
update(Details, TIME);
hop_first(all_cleaners, direct_links)).
```

SG solution analysis

This is highly dynamic and fully distributed solution, where satellites are moving with high speed and possibly in different orbits or even directions. The self-spreading holistic scenario can operate with any constellation topologieswhich may rapidly change in time, and also any number of currently available satellites, which may not be known in advance. This SG solution is also much superior to traditional representation of a distributed system in the form of collection of agents exchanging messages, where, moreover, such a system in this dynamic multiple satellite case cannot be defined in advance at all.

Many more on debris and also management of multiple satellite architectures can be found at [17-22].

5. Conclusion

This concludes the paper by summarizing the obtained results, which are confirming uniqueness and power of the model proposed, where instead of representing distributed systems as collection of parts or agents exchanging messages, we have their integral, holistic and semantic level solutions as self-evolving and self-matching spatial patterns, which can often simplify and shorten global management code up to a hundred times in comparison with other approaches and languages. We can also mention here the new activities planned in this area, including new SGT implementation, which can be done even within traditional university environments as for the previous technology versions, new patenting of the SG model as a further development of the previous patent [13], and also new book on using SG model and the resultant technology for management of integrated terrestrial and celestial systems. Such a book is currently in preparation under the contract with

Bibliography

1. Algorithm.

2. Definition of ALGORITHM. Merriam-Webster Online Dictionary.

3. Goodrich MT and Tamassia R. "Algorithm Design: Foundations, Analysis, and Internet Examples". John Wiley and Sons, Inc (2018).

4. Heuristic.

5. Flowchart.

6. A Lynch. Flow Chart Design - How to design a good flowchart, 04/22/2021.

7. Sapaty PS. "Symbiosis of Real and Simulated Worlds under Spatial Grasp Technology". Springer (2021): 251.

8. Sapaty PS. "Complexity in International Security: A Holistic Spatial Approach". Emerald Publishing (2019):160.

9. Sapaty PS. "Holistic Analysis and Management of Distributed Social Systems". Springer (2018): 234.

10. Sapaty PS. "Managing Distributed Dynamic Systems with Spatial Grasp Technology". Springer (2017): 284.

11. Sapaty PS. "Ruling Distributed Dynamic Worlds". New York: John Wiley and Sons (2005): 255.

12. Sapaty PS. "Mobile Processing in Distributed and Open Environments". New York: John Wiley and Sons (1999): 410.

13. Sapaty PS. "A distributed processing system". European Patent N 0389655, Publ. 10.11.93, European Patent Office 35.

14. PS Sapaty. "Global Network Management under Spatial Grasp Paradigm, Global Journal of Researches in Engineering". *Journal of General Engineering* 20.5 (2020):58-69.

15. PS Sapaty. "Spatial Management of Large Constellations of Small Satellites". *Mathematical Machines and Systems* 2(2021).

16. Sapaty PS. "Global Management of Space Debris Removal Under Spatial Grasp Technology". *Acta Scientific Computer Sciences* 3.7 (2021).

17. Space Debris.

18. Space Debris, NASA Headquarters Library.

19. Deorbit Systems, National Aeronautics and Space Administration, Nov 28 (2020).

20. Y Chen., et al. "Optimal mission planning of active space debris removal based on genetic algorithm". IOP Conf. Series: Materials Science and Engineering 715 (2020): 012025.

21. PS Sapaty. "Managing multiple satellite architectures by spatial grasp technology". *Mathematical Machines and Systems* 1 (2021): 3-16.

22. PS Sapaty. "Spatial Management of Large Constellations of Small Satellites". *Mathematical Machines and Systems* 2 (2021).

23. Sapaty PS. "Spatial Grasp as a Model for Space-based Control and Management Systems". *Mathematical Machines and Systems* 1 (2021): 135-138.

Volume 3 Issue 9 September 2021