Seeding Programming

Vladimir Mochalov

Abstract The work is aimed at formalizing the implementation of the steps of the new method "seeding programming" focused on solving some optimization problems. Michelangelo told that there is a statue in every stone and all that is needed is to be able to remove all unnecessary and to take the statue to light. Based on Michelangelo's statement in the proposed method, we search for such a sequence of elements to remove from the original space ("stone"), which will lead to the formation of a set of remaining undeleted elements with the desired objective function. Initial elements of the search space either can be specified or they can be searched using special covering algorithms. To search for the sequence of elements to remove from the search space, we suggest to use search agents that form and use shared global memory.

Keywords Optimization method · Knowledge-based multi-agent system
Synthesis of solutions

1 Introduction

Michelangelo di Lodovico Buonarroti Simoni told that there is a statue in every stone and all that is needed is to be able to remove all unnecessary and to take the statue to light (Figs. 1 and 2). Statues differ from each other in appearance. Looking at a statue, we see the final result of the sculptor's work and speaking in technical language, we see the final objective function of the process of removing unnecessary elements ("small parts of the stone") from the original space ("stone"). Knowing the objective function, we can start to search for such extra elements from the initial space in order to obtain the desired result in the remainder.

V. Mochalov (✉)
Institute of Cosmophysical Research and Radio Wave Propagation FEB RAS,
Mirnaya str. 7, Kamchatka Region, Elizovskiy, Paratunka 684034, Russia
e-mail: sensorlife@mail.ru

R. Cárdenas et al. (eds.), *Proceedings of the 2nd International Conference on BioGeoSciences*, https://doi.org/10.1007/978-3-030-04233-2_5

Fig. 1 Michelangelo took the marble stone, removed all unnecessary, and got the statue of David

Fig. 2 Marble statues of the sculptor Bonazza located in the city of Peterhof

We draw the following analogies: “stone”—initial space from which we will remove the extra elements; “statue” is the best of the found solutions satisfying the given objective function; “sculptor” is a search agent that on the basis of knowledge and experience removes unnecessary elements from the initial space in order to obtain in the remainder a solution that satisfies the specified objective function. So, based on Michelangelo’s statement in the proposed method “Seeding programming,” we search for such a sequence of elements to remove from the original space (“stone”), which will lead to the formation of a set of remaining undeleted elements with the desired objective function. Initial elements of the search space can either be specified, or they can be searched using special covering algorithms. To search for the sequence of elements to remove from the search space, we suggest to use search agents.

2 Stages of Solving Problems Using Seeding Programming

Stage 1: Create objectives tree for solving optimization problem, define optimization parameters, define functions of estimating various optimization parameters, and form general target function and stopping criterion.

A concept of the “objectives tree” was introduced by Churchman and Ackoff in 1957. An objectives tree is a structure, constructed on the hierarchy principle (distributed into levels, ranged) assembly of project objectives, in which the following ones are emphasized: the general objective (“tree root”) and the subgoals of the first, second, and consequent levels subject to it (“tree branches”). In Fig. 3, a generalized objectives tree is shown. In leaf nodes of the tree, simple tasks are formed. Often the simple tasks are the requirements on achieving the specified thresholds of optimization parameters.

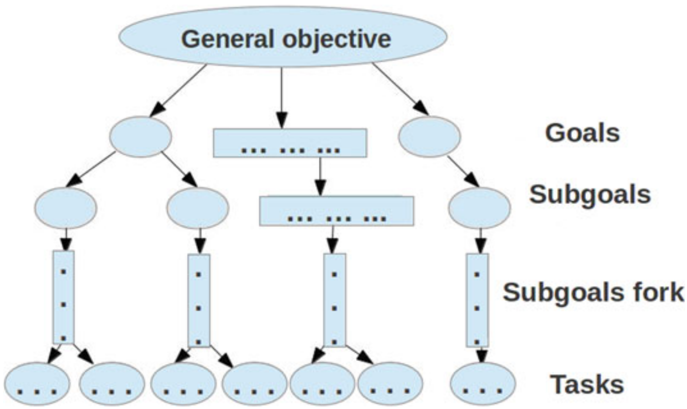

Fig. 3 Objectives tree

Stage 2: Select a suitable initial space ("stone"), removing elements from which we will look for the solution we need. It is necessary to mark that the smaller initial space contains less excess elements, so the less work required to remove them. At the same time, the initial space with a large number of elements potentially can be a greater number of better solutions compared to a space containing fewer elements. It is necessary to note the importance of suitable initial space selection. For example, if we choose the initial space that is too small, then we cannot find a solution that satisfies us, and vice versa, if we select too big initial space, then we will search for satisfying us solution for a very long time. We assume that elements of the initial space are either specified or their search is performed with special covering algorithms.

One of the interesting approaches to search the initial space is a knowledge-based multi-agent method for finding the sequence of adding elements in order to form a suitable initial space that contains the solution we need (ideally the initial space from which nothing needs to be removed). In this case, each agent adds elements to the start space on the basis of it rules of moving and general knowledge of the problem until it forms a set of elements which contains required solution after we can run seeding programming method to remove unnecessary elements (Fig. 4). For example, in paper [1] joint use of strategies for sequentially add elements to search space and sequentially remove elements from initial space are considered.

Stage 3: Create a shared global memory of agents (SGMA) for storing the agents' knowledge and experience about travelled routes (e.g., in [1] as SGMA is proposed to use shared global memory of the stored pheromone). Create an empty set of best

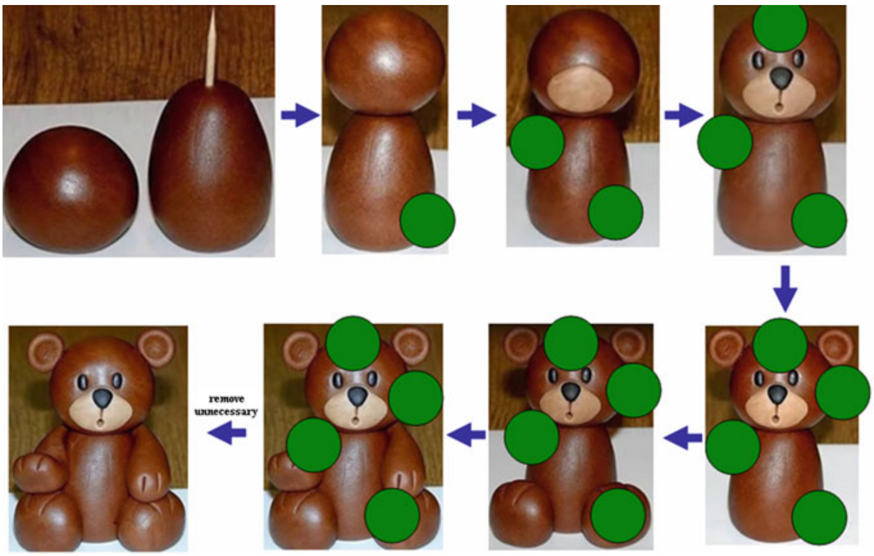

Fig. 4 Illustration of sequentially addition of elements to the start space

solutions Ω_{BEST}. Determine the maximum number of solutions b_{K} that will be stored in Ω_{BEST}.
Stage 4: Define agents that will be used to search unnecessary elements in initial space. Set for each agent parameters of its operation, objective function, and lower bound estimate K_{MIN} of the objective function.
Stage 5: Perform the following steps for each agent:

5.1. Using the movement rules of the agent and the SGMA, form the route M_{T} (array of elements) of agent moving on initial space elements,
5.2. Flip M_{T} array left to right,
5.3. In a loop for each element $T_{\mathrm{O}} \in M_{\mathrm{T}}$ perform these steps: (a) temporarily exclude T_{O} from M_{T}; (b) compute the confidence factor K_{D} of meeting the requirements of the objective function; (c) if in absence of T_{O}, the condition $K_{\mathrm{D}} < K_{\mathrm{MIN}}$ is satisfied then put T_{O} back to M_{T} into its place,
5.4. Using remaining elements in M_{T}, compute the confidence factor K_{DALL} of meeting the requirements of the overall objective function. If $K_{\mathrm{DALL}} > 0$, then using some information about the remaining elements in M_{T} update SGMA.

Stage 6. If the stopping criterion is not met and if it is necessary, then update SGMA (e.g., in paper [1] shared global memory of the stored pheromone is updated using the following rule: $\Delta\tau_{ij}(t+1) = (1-p) * \tau_{ij}(t) + \Delta\tau_{ij}(t)$ [2]), reduce initial solution search space, and move to the stage 4. Otherwise return the best solution from Ω_{BEST}.

3 Seeding Programming Implementation for Synthesis of a Given Category Nodes Placement into Geospace Question–Answering Sensor Network Structure

New space technologies (nanosatellites, CubeSats, SmallSats, etc.), private space companies and the projects for launching thousands of small satellites to organize space networks with different purposes give principally new opportunities to monitor geospheres. We should note the increasing number of separate monitoring systems applying the data obtained from the geospace. The term "geospace" is understood as the region of space that goes from the solar photosphere to the atmosphere of Earth. It includes the solar photosphere, chromosphere and corona, the solar wind, Earth's magnetosheath, magnetosphere, thermosphere, ionosphere, and atmosphere.

In this paper on the basis of the personal results obtained before in the area of sensor networks construction [1, 3–6], semantic analysis, and question–answering systems [7–9], we introduce a new notion of "geospace question–answering sensor networks" (GQASN) that means a distributed network which monitors ambient environment parameters applying the data from geospace and allowing nodes to answer defined types of natural language questions as well.

In the work, we use a model of the GQASN structure (Fig. 5), where on the functional level the following types of GQASN nodes can be defined: (1) functional nodes (F-nodes) that collect information in some neighborhood of their location; (2) transit nodes (T-nodes) that manage routing and retransmit the information collected by F-nodes to the information collection centers (ICC) to be utilized further; (3) ICCs that manage the GQASN and process information collected by the GQASN. In general case, there can be multiple ICCs in the GQASN, and the information that has arrived into each of them is available to one or multiple users for making decisions and performing certain actions. It means that information received by F-nodes should be retransmitted, with a required degree of reliability, to several ICCs by means of transit nodes allocated within the given object in a certain way. We think that ICC is capable of performing F-node and T-node functions. F-node can perform the T-node functions and information between nodes can be transmitted both via the wire and wireless networks.

Designing of the GQASN requires the solution of many complicated problems referring to different areas of research; they are: projecting of network nodes (measurement stations, sensors, etc.); construction of different physical–mathematical models of monitoring processes of ambient environment parameters applying the geospace data; model selection for information collection from the GQASN; development of methods and algorithms for the GQASN structure synthesis; estimate of measurement error and limitations, estimate of spatial and other limitations of network node placement; ensuring of the defined functional and

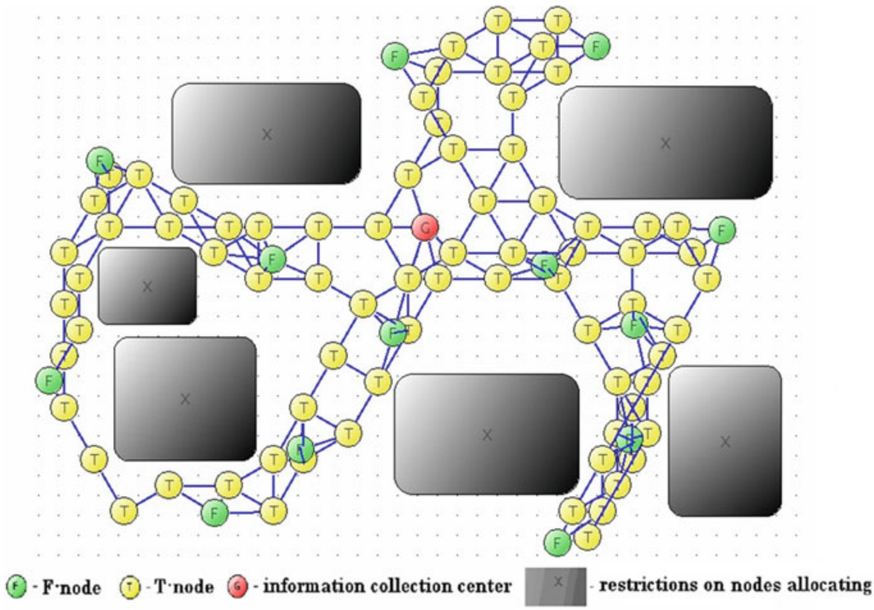

Fig. 5 Example of a distributed fault-tolerant GQASN structure

structural parameters of the synthesized GQASN; development of fitness functions of network nodes placement; development of algorithms for question–answering agents placement into the GQASN structure; development of self-organization algorithms for different GQASN levels (e.g., the function level performed by the nodes, routing level, level of tasks distribution between nodes and question–answering agents).

Two types of GQASN structures can be distinguished:

- distributed fixed network in which all the GQASN nodes are not moved after the initial placement;
- decentralized mobile self-organizing network compound of fixed nodes (distributed fixed network segment) and mobile units (mobile network segment) which can be moved in different directions and, as a consequence, form different network structures in dynamics by break and establish network connections with other nodes, removal and installation of new nodes into the network structure. Mobile units can be installed both on satellites and on mobile robots (drones, above-water and under-water vehicles). Some functions of the mobile units are: formation of a self-organizing GQASN structure; geographically distributed acquisition of data from the GQASN nodes; organization of the interaction of the GQASN mobile network segment with a fixed one. When applying the mobile robots, the following is possible: accurate nodes placement, distribution nodes over territory; moving, removing, reprogramming of nodes; charging and replacement of the GQASN nodes power sources; planning of cooperative behavior of mobile robots in the process of general aim solution and the GQASN nodes replacement based on the aims and current measurements of the whole GQASN.

Possible ways of problem statement for the GQASN structure synthesis

1. Synthesis of ICCs allocation. In this task, we know the spatial restrictions for allocating the ICCs. Also, we know the allocation of pre-installed ICCs. It is necessary to allocate ICCs in such way that the designed GQASN structure would have the “desired properties” assigned by a designer. During the synthesis of ICCs allocation, it is possible to optimize (by their removal or moving) some pre-installed ICCs noted by the designer.
2. Synthesis of F-nodes allocation. We know the spatial restrictions for allocating the F-nodes. Also, we know the allocation of ICCs and allocation of pre-installed F-nodes. It is necessary to allocate new F-nodes in such way that the designed GQASN structure would have the “desired properties” assigned by a designer. During the synthesis of F-nodes allocation, it is possible to optimize (by their removal or moving) some pre-installed F-nodes noted by the designer.
3. Synthesis of T-nodes allocation. We know the description of the GQASN allocation object, spatial restrictions for allocating the T-nodes. Also, we know the allocation of F-nodes, ICCs, and pre-installed T-nodes. It is necessary to allocate T-nodes in such way that the designed GQASN structure would have the “desired properties” assigned by a designer. During the synthesis of T-nodes

allocation, it is possible to optimize (by their removal or moving) some pre-installed T-nodes noted by the designer.

4. Complex sequential synthesis of ICCs, F-nodes, and T-nodes allocation. This statement suggests a sequential allocation of ICCs first (statement 1), then F-nodes (statement 2), and then T-nodes (statement 3).

It should be noted that the search space for synthesized solutions of concrete GQASN structure is very large and there are complex constraints in the objective function, and many of the solvable problems are NP-complete and to search for exact and approximate solutions of these problems, various algorithms of artificial intelligence [1, 10–13], linear and integer programming [14–16] are currently used and distributed calculations are performed.

Figure 6 illustrates a functional scheme of a given category nodes placement into the GQASN structure. This scheme can be used as the basis for synthesis of ICC, F-nodes, and T-nodes placement.

The question–answer agent (QA-agent) performs the function of generating the answer from natural language questions by collection, aggregation, and accumulation information from some F-nodes that are serviced by this QA-agent. After the accumulation of sufficient information from the group of F-nodes, the QA-agent generates an answer. The QA-agents can interact with each other to be able to answer the given types of questions under the established limitations. Physically, the QA-agent is a software/hardware add-on that can upgrade any type of GQASN nodes. The QA-agents can differ from each other by technical capabilities (due to various hardware and software add-ons) and functionality capabilities (the ability to answer different types of questions, performed functions, etc.).

In general, formulated in the natural language question Q enters to the input of one of the QA-agents that perform the functions of task coordinator for other QA-agents. After it the question Q enters to the input of the semantic analyzer module, which create ontological-semantic graph $G(Q)$. The graph $G(Q)$ enters to the input of the module for selection of QA-agents, which are best suited for generating the answer to question Q. The information about selected QA-agents is placed into a set FQA= {FQA_1, FQA_2, FQA_3, …, FQA_k}. The set FQA enters to the input of generating requests for QA-agents module. As a result, for each QA-agent $\mathrm{FQA}_i \in \mathrm{FQA}$, this module forms a request q_i. Each q_i request is transmitted to the QA-agent FQA_i, which first tries to find the necessary information in the local database and if it is not found FQA_i select serviced F-nodes from which the necessary information should be collected. The information of selected at this stage F-nodes is placed into the set $F = \{F_1, F_2, F_3, \ldots, F_n\}$. Further, the task formation module generates a task t_j for each F-node $F_j \in F$. Each F-node $F_j \in F$ that receives the task t_j executes it (using information stored on this F-node or receive it from the environment with the help of a sensor installed on the F-node) and sends the response r_j back to the QA-agent which generated the task t_j. On the basis of all the responses $r_1, r_2, r_3, \ldots, r_n$ obtained from F-nodes $F_1, F_2, F_3, \ldots, F_n$ QA-agent FQA_i generates a_i answer and sends it back to the task assignment

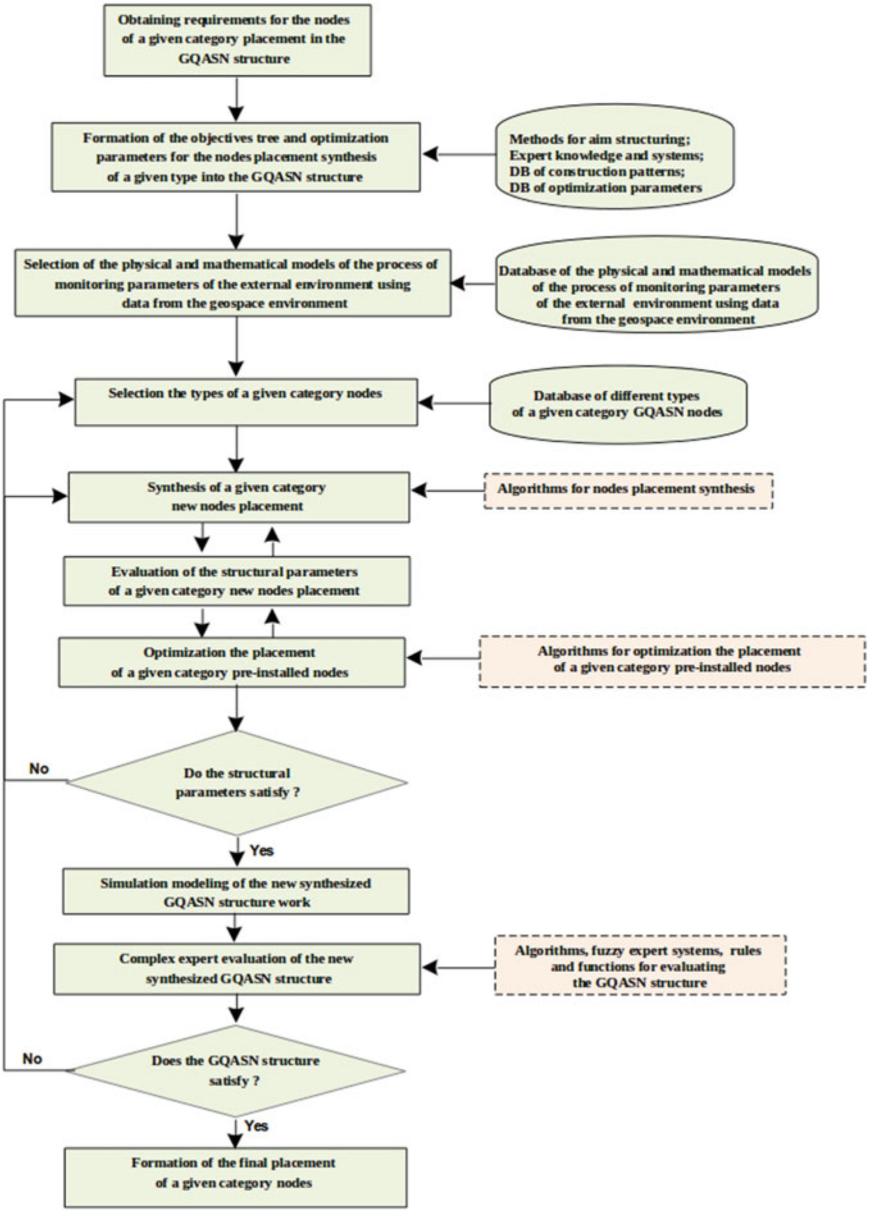

Fig. 6 Functional scheme of a given category nodes placement into GQASN structure

coordinator. Thus, the task assignment coordinator receives all answers from QA-agents and on the basis of them makes up a general answer *A*, which is transmitted to the user as an answer to the question *Q* (Fig. 7).

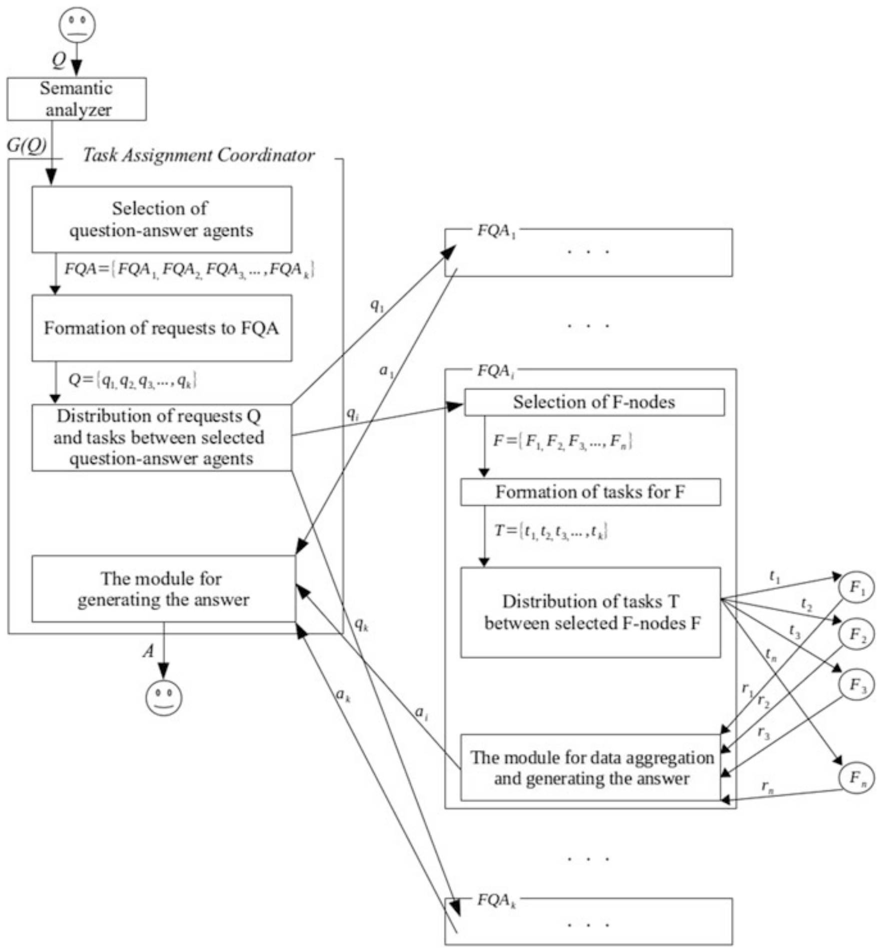

Fig. 7 Scheme for generating the answer using the GQASN question–answer agents

In recent years, the research area of *Natural Computing* is rapidly developing. It unites mathematical methods in which the principles of natural mechanisms of decision making are embedded [2]. Scientists have developed bio-inspired algorithms (BA) of modeling animals' behavior ([2, 10–13], etc.]) for solving various optimization problems that either do not have exact solution or the solutions' search space varies large and complex constraints of the objective function are presented, as well as NP-complete.

The described recommendations on applying BA and the proof in [17] that even the constrained variant of the problem of minimal coverage on plane is NP-complete allow us to conclude about the possibility to apply bio-inspired algorithms for the GSASN structure design.

Bio-inspired algorithms can be seen as multi-agent systems, each agent in which operates autonomously on very simple rules. The most frequently used bio-inspired agents (B-agents) include: ants, bees, termites, fireflies, birds, fish, bats, cats, and wolves.

The initial data of the algorithm for synthesis of a given category nodes $type_x$ placement are the following: allocation of pre-installed nodes of $type_x$ (set of nodes Ω_{pin}); allocation of installed nodes of type $\neq$ $type_x$; description of the object that the GQASN needs to be located at (its dimensions, scheme, spatial requirements for nodes allocation and etc.); characteristics of ready-to-use nodes of $type_x$; adopted self-organization and routing algorithms; information collection model; functional requirements; optimization parameters; fuzzy expert systems, etc.

The algorithm below is based on the adaptation of the multi-agent bio-inspired algorithm for wireless sensor network design proposed in paper [1].

Step 1 Create objectives tree for solving optimization problem. Define a set M_{ALL} of all optimization parameters; the functions for calculating the parameters of M_{ALL}; a subset of optimization parameters $M_1 \in M_{ALL.}$ Determine the membership functions of fuzzy sets that characterize the optimization parameters of the M_{ALL}; fuzzy expert system to derive the confidence factor to meet the functional requirements of the designer. Create an empty set of the best solutions Ω_{BEST}. Determine the maximum number of solutions b_K that will be stored in Ω_{BEST}.

Step 2 Create a set Ω_p of the possible placement points of type $type_x$ nodes (the set Ω_p can be formed with the help of: the algorithms for covering the object of placement with a mesh (based on an equilateral triangle or hexagon or square) or with circles with a given radius; covering algorithms in accordance with the choice and recommendations of the designer; other covering algorithms). Create an empty set Ω_T. Create node of type $type_x$ in each point of Ω_p and add this node to Ω_T set.

Step 3 Create a shared global memory of the stored pheromone (SGMSP) to share some "knowledge" between B-agents. The pheromone is stored on the edges of a fully connected undirected weighted graph (FCUWG), the nodes of which are the ones of $type_x$. To store the edges of the graph in computer memory, it is required to create a two-dimensional array *feromoneNetwork* with $N(N - 1)/2$ memory cells of type *float*, where $N = |\Omega_T|$ is the number of nodes. All values of *feromoneNetwork* must be initialized as zeros.

Step 4 Define agents that will be used.

Step 5 Execute bio-inspired multi-agent algorithms.

5.1. Create a two-dimensional array *feromoneDif* to store changes in the pheromone using, for example, the following Java code:

```
float feromoneDif [ ] [ ] = new float[N-1] [ ];
for(int i = 0; i < N; i++)
```

feromoneDif [i] = new float[N - (i+1)];

All values of *feromoneDif* must be initialized as zeros (in the above code, the zero initialization is done automatically). Define the number of different bio-inspired agents *m*, the strategy for choosing the initial location of the agent, and other parameters needed for the agent to perform the work.

5.2. For each agent, perform the following steps (the code can be parallelized, i.e., to run in a separate thread for each agent):

5.2.1 Form, using the movement rules of the B-agent, SGMSP the route M_T (array) of agent moving on nodes Ω_T.

5.2.2. Create an empty extensible array of nodes M_{STR}, in which the nodes of the designed structure will be placed.

5.2.3. Select the design strategy:

(a) sequentially add nodes to the network -> Go to step 5.2.4;

(b) sequentially remove nodes from the network -> Add to M_{STR} all nodes from M_T array in the same sequence order. Go to step 5.2.11.

5.2.4. Create an empty set H_P, which will contain the caches of such internal parameters of the functions of computing estimates M_1, which will increase the speed of computing estimates M_1 for the next iteration. Create a variable i to store the index of the current node from the M_T array and initialize its value to 0 (i = 0). Set the node T_C ($T_C = M_T[0]$) as the current one.

5.2.5. Add to M_{STR} the node T_C. Form the network structure S_S of nodes M_{STR}.

5.2.6. Calculate using the caches H_P the values estimations of the optimization parameters of the set M_1 having structure S_S. Clear H_P. Save the caches of the internal parameters of the functions of computing estimates M_1 of the current iteration to the set H_P.

5.2.7. Using a fuzzy expert system, calculate the reliability coefficient K_{D1} of meeting the requirements of the designer for parameters of a set M_1 of structure S_S.

5.2.8. If $K_{D1} > p_1$, where p_1 is a set threshold, go to step 5.2.11.

5.2.9. If $i < |M_T|$, put $i = i + 1$ and accept the next node $T_C = M_T[i]$ as the current one. Repeat steps 5.2.5.–5.2.9. while i does not become equal to $|M_T|$.

5.2.10. Exit with notification of the failure from the function of agent design of the network structure.

5.2.11. Steps of eliminating optimization:

5.2.11.1. Select the strategy of eliminating optimization:

(a) step-by-step optimization with consideration of optimization parameters M_1. The fuzzy expert estimation of the structural parameters M_1 is used;

(b) step-by-step optimization with consideration of all optimization options M_{ALL}. The unit of simulation modeling and complex assessment of the network is used;

5.2.11.2. Revert the M_{STR} array,

5.2.11.3. In a loop, temporarily exclude each node $T_O \in M_{STR}$ from M_{STR}, then compute the confidence factor K_D of meeting the requirements of the eliminating optimization strategy parameters. If in absence of node T_O evaluation of network structure stops meeting the designer requirements, put T_O back to M_{STR} into its place.

5.2.11.4 Revert the M_{STR} array,

5.2.12. Perform simulation modeling of the network. The results of the modeling and structural-parametric estimates of the various parameters are the input to the complex expert system for evaluation of network structure. Calculate with the latter the confidence factor K_{DALL} of meeting all the designer requirements.

5.2.13. If $K_{DALL} > 0$ then, in accordance with one of the following strategies, increase the pheromone amount in the array *feromoneDif*:

(a) consequent update—increase the amount of pheromone on the edges of the agent sequential traveling on nodes of M_{STR} by the value equal to $\Delta\tau_{ij,k}(t) = Q_{agent} * K_{DALL}$, where Q_{agent} is the amount of pheromone secreted by the agent on one edge;

(b) full-mesh update—increase the amount of pheromone on all edges of the fully connected graph constructed on the basis of nodes of M_{STR} by the value equal to $\Delta\tau_{ij,k}(t) = Q_{agent} * K_{DALL}$.

5.2.14. If K_{DALL} is greater than the estimate of the worst solution from Ω_{BEST}, or ($|\Omega_{BEST}| < b_K$ and $K_{DALL} > 0$), then add into Ω_{BEST} the current solution. By the solution, we mean the couple (M_{STR}, K_{DALL}). If $|\Omega_{BEST}| \geq b_K$, then leave in Ω_{BEST} only b_K best solutions.

5.3. After all agents have performed step 5.2, update SGMSP (*feromoneNetwork* array) in accordance with the following well-known rule [2]: $\Delta\tau_{ij}(t + 1) = (1 - p) * \tau_{ij}(t) + \Delta\tau_{ij}(t)$, where $\Delta\tau_{ij}(t)$ is the amount of pheromone on edge (i, j) in the array of pheromone changes *feromoneDif*, and $p \in [0, 1]$ is the coefficient of pheromone evaporation. To enhance the intermediate best solutions, the amount of pheromone on the edges of the routes of the best solutions Ω_{BEST} should be increased (an example is using "elite" ants).

5.4 If the stopping criterion is not met, go to step 5.1.

Step 6 If it is necessary to continue the search, then create new set Ω_T and add to it type type$_x$ nodes located in points of possible nodes placement (e.g., to cover the object of nodes placement with a mesh more densely in comparison with the previous coverage) and move to Step 3. Otherwise return the best solution from $\Omega_{BEST.}$

4 Conclusion

Seeding programming is a new method for solving various optimization problems that either do not have exact solution or the solution search space is very large and complex constraints of the objective function are present, as well as NP-complete.

One possible implementation of the algorithm that implements the seeding programming method is given in this paper for synthesis of a given category nodes placement into the GQASN structure.

References

1. Mochalov VA (2015) Multi-agent bio-inspired algorithms for wireless sensor network design. In: Proceedings on IEEE 17th international conference on advanced communication technology, ICACT 2015, Phoenix Park, Korea, pp 34–42
2. Bonavear F, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, New York, 320 p
3. Mochalov VA (2015) Synthesis of the wireless sensor network structure in the presence of physical attacks. Lect Notes Comput Sci 9247:11–22
4. Mochalov VA, Pschenichnikov AP (2015) Functional scheme of the flying sensor networks architecture design. ICACT Trans Adv Commun Technol (TACT) 4(4):659–663. https://doi.org/10.1109/ICACT.2016.7423591
5. Mochalov VA (2016) Certificate of registration of computer software “Program for synthesis of monitoring networks by bio-inspired algorithms” No. 2016612039
6. Mochalov VA, Mochalova AV (2017) Algorithms for changing the structure of geospace self-organizing question-answering sensor networks. In: VIII international conference “solar-terrestrial relations and physics of earthquake precursors”, p 11. https://doi.org/10.1051/e3sconf/20172002009
7. Mochalova AV, Mochalov VA (2017) Mathematical model of an ontological-semantic analyzer using basic ontological-semantic patterns. In: Lecture notes in artificial intelligence, proceedings of 15th Mexican international conference on artificial intelligence, pp 53–66
8. Mochalova AV, Zacharov VP, Mochalov VA (2017) Ontology modification using ontological-semantic rules. In: 19th international conference on advanced communications technology (ICACT)—opening new era of smart society, pp 902–906
9. Kuznetsov VA, Mochalov VA, Mochalova AV (2016) International conference on advanced communication technology, ICACT, pp 651–658
10. Brabazon A, O’Neill M, McGarraghy S (2015) Natural computing algorithms. Springer, 554 p
11. Mandal JK, Mukhopadhyay S, Pal T (2016) Handbook of research on natural computing for optimization problems (Vols 2), Igi-Global, 1015 p
12. Fister I Jr, Xin-She Y, Fister I, Brest J, Fister D (2013) A brief review of nature-inspired algorithms for optimization. ELEKTROTEHNIˇSKI VESTNIK 80(3):1–7
13. Binitha S, Sathya S (2012) A survey of bio inspired optimization algorithms. Int J Soft Comput Eng (IJSCE) 2(2):137–151
14. Gounaris CE, Rajendran K, Kevrekidis IG, Floudas CA (2015) Designing networks: a mixed integer linear optimization approach, 56 p. https://arxiv.org/pdf/1502.00362.pdf
15. Taccari L (2015) Mixed-integer programming models and methods for bilevel fair network optimization and energy cogeneration planning. Ph.D. dissertation, 209 p

16. Fraccaroli E, Quaglia D Toolchain for optimal network synthesis. http://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid014072.pdf
17. Fowler RJ (1981) Optimal packing and covering in the plane are NP-complete. Inf Process Lett 12(3):133–137