SENSOR NETWORK CONTROL BASED ON RANDOMIZED AND MULTI-AGENT APPROACHES

Anna Sergeenko
Saint Petersburg State University
(Faculty of Mathematics and Mechanics,
Science Educational Center
"Mathematical robotics and artificial intelligence"),
and IPME RAS
Russia
a.sergeenko@spbu.ru

Oleg Granichin
Saint Petersburg State University
(Faculty of Mathematics and Mechanics,
Science Educational Center
"Mathematical robotics and artificial intelligence")
and IPME RAS
Russia
o.granichin@spbu.ru

Article history:
Received 09.08.2022, Accepted 27.09.2022

Abstract

In this paper, a development of randomized and multi-agent algorithms is presented. The examples and their advantages are discussed. Different combined algorithms, which are applicable for the multi-sensor multi-target tracking problem are shown. These algorithms belong to the class of methods used in derivative-free optimization and has proven efficacy in the problems including significant non-statistical uncertainties. The new algorithm, which is an Accelerated consensus-based SPSA algorithm is validated through the simulation.The main feature of that algorithm, combining the SPSA techniques, iterative averaging ("Local Voting Protocol") and Nesterov Acceleration Method, is the ability to solve distributed optimization problems in the presence of signals with fully uncertain distribution; the only assumption is the signal's limitation.

Key words

Randomized algorithms, multi-agent algorithms, SPSA, Local Voting Protocol, Tracking, Nesterov Acceleration

1 Introduction

An exact solution to any problem if the problem is formulated fully and accurate, but connections and relationships in the real world so complex and diverse that it is almost impossible to describe many phenomena in a strictly formalized way. A typical approach in theory is the choice of a mathematical model close to real processes and the inclusion of various perturbations (noise) in it, related, on the one hand, to the roughness of the mathematical models and, on the other hand, characterizing uncontrolled external perturbations on an object or system. Moreover, real systems are rarely exhaustively described by limited mathematical models.

In this paper, two types of algorithms, which are applicable to the problem described above, are discussed: randomized and multi-agent algorithms. It is important to note that there is no strict difference between randomized and multi-agent algorithms. Many algorithms have properties of both types, such as Ant colony algorithm, DNA-computing, Gossip protocol and others, which are discussed further.

The development of randomized and multiagent algorithms is presented further.

1.1 Randomized Algorithms

The development and accessibility of computer technology also had an impact on the classical sections of mathematical statistics, encouraging development and giving priority to recurrent schemes evaluation. The basis of a fairly new approach to solving problems of estimation and optimization in bad conditions (for example, with a degenerate sequence of observations) is the use of trial disturbances. If, when solving a problem through the input channels of a system or algorithm, it is possible to include in consideration some new perturbation with experimenter-specified or good known statistical properties, then it can be used to "enrich" information in the observation channel. Sometimes the role of a test perturbation can be played by a present in system measurable random process. In control systems, trials can be added via the control channel, in other cases, a randomized design of the experiment may play the role of a trial exposure. In the study of the updated system with a trial perturbation, which is sometimes a problem written in a different form of the old one, even using traditional methods, it is often possible to obtain encouraging results about the convergence and applicability of new algorithms. One of their remarkable characteristics is the consistency of assessments under "almost arbitrary" perturbations. Algorithms in which one or more steps are based on a random selection of a rule are called randomized algorithms.

1.1.1 Manhattan Project and Monte Carlo Method Randomized algorithms start with the Manhattan Project. The Manhattan Project is a secret US research project that began during World War II on August 13, 1942 to develop nuclear weapons [Smyth, 1945]. Before that, since 1939, research was conducted in the "Uranium Committee" (S-1 Uranium Committee). Scientists from the United States of America, Great Britain, Germany and Canada took part in the project. The project created three atomic bombs: the plutonium "Thing" (Gadget) (exploded during the first nuclear test), the uranium "Kid" (Little Boy) (dropped on Hiroshima on August 6, 1945) and the plutonium "Fat Man" (Fat Man) (dropped on Nagasaki on August 9, 1945).

It was during this project that a breakthrough occurred in the development of computers. Before World War II, computers were people who performed calculations. In the 18th century, paper calculations provided the first accurate predictions of comet returns and eclipses. Until the 1940s, there was a small selection of office equipment, adding machines, card sorters, and simple mechanical calculators available to assist human computers. The military needed ballistic tables, tide tables, and many kinds of large-scale mathematical tables. At Los Alamos, scientists and engineers needed more precision than they could get with slide rules, so they called on groups of people, mostly women, to do the math work.

Before the advent of modern digital computers, complex analog computers were used to perform calculations. Although the word "computer" has come to mean several things, analog and digital computers perform the same basic task: calculating and manipulating numbers using logical rules. Analog computers have been around for hundreds of years and include such simple devices as the slide rule. Analog computers were vital to the work at Los Alamos. The Los Alamos project also used old IBM punched-card computers. When the machines were first brought to the lab, scientists were skeptical. A race was organized between IBM machines and manually operated computers. Although the two kept pace at first, after about a day's work, the manual operators began to tire while the punch-card machines continued to operate.

First, Enrico Fermi in Italy in the 1930s, and then John von Neumann and Stanislaw Ulam in the 1940s at Los Alamos, suggested that the connection between stochastic processes and differential equations "in reverse" could be used. They proposed to use the stochastic approach for approximating multidimensional integrals in the transport equations that arose in connection with the problem of neutron motion in an isotropic medium.

The idea was developed by Ulam, who, while playing solitaire while recovering from an illness, wondered what was the probability that solitaire would work out. Instead of using the usual combinatorics considerations for such problems, Ulam suggested that one could simply run the experiment numerous times and, by counting the number of successful outcomes, estimate the probability. But due to the need to conduct a large number of the same type of experimental actions, the method was not widely used.

With the advent of the first electronic computer ENIAC, which could generate pseudo-random numbers at high speed and use them in mathematical models, renewed interest in stochastic methods. Stanislav Ulam discussed his ideas with John von Neumann, who ended up using ENIAC for Ulam's proposed statistical selection method for various neutron transport problems [Nicholas Metropolis and Stanislaw Ulam, 1949]. Due to the need to turn off ENIAC for a significant time at the end of 1946, to continue research on the movement of neutrons, Enrico Fermi even developed a specialized analog computer, which was given the name FERMIAC (by analogy with ENIAC, but with an indication of Fermi's authorship), which was also on a mechanical level, the Monte Carlo method is implemented.

After the beginning of the use of computers, there was a big breakthrough, and the Monte Carlo method was used in many problems, for which the stochastic approach turned out to be more effective than other mathematical methods. However, the use of such a technique also had its limitations due to the need for a very large number of calculations to obtain results with high accuracy.

The year of birth of the term "Monte Carlo method" is considered to be 1949, when Metropolis and Ulam's paper "The Monte Carlo Method" [Nicholas Metropolis and Stanislaw Ulam, 1949] was published. The name of the method comes from the name of the commune in the Principality of Monaco, widely known for its numerous casinos, since roulette is one of the most widely known random number generators. Stanislav Ulam writes in his autobiography The Adventures of a Mathematician that the name was suggested by Nicholas Metropolis in honor of his uncle, who was a gambler.

In the 1950s, the method was used for calculations in the development of the hydrogen bomb. The main merits in the development of the method at this time belong to the employees of the US Air Force Laboratories and the RAND Corporation.

Currently, the main efforts of researchers are aimed at creating efficient Monte Carlo algorithms for various physical, chemical and social processes for parallel computing systems.

Sergei Mikhailovich Ermakov [Ermakov S.M., 1971] has been studying the Monte Calo method at the Faculty of Mathematics and Mechanics of St. Petersburg State University since 1956. One of the main areas of his scientific interests is the theory of the Monte Carlo method. Work on the evaluation of multidimensional integrals, the study of pseudo-random number generators, the probabilistic solution of linear and nonlinear integral equations contributed to the transformation of this computational method from a set of semi-empirical techniques into a rather strictly defined branch of mathematics.

The Monte Carlo method has been developed not only in optimization theory, but also in quantum theory. There is a quantum Monte Carlo method [Moral and Doucet, 2004], which is widely used to study complex molecules and solids. This name combines several different methods. The first of these is the variational Monte Carlo method, which is essentially the numerical integration of multidimensional integrals that arise when solving the Schrödinger equation. To solve a problem involving 1000 electrons, 3000-dimensional integrals are required, and in solving such problems, the Monte Carlo method has a huge performance advantage over other numerical integration methods. Another variation of the Monte Carlo method is the diffusion Monte Carlo method.

1.1.2 Random Search Method Many practical problems in mathematical form are presented in the form of problems about finding the root of some function or about finding a minimum point. In this case, it is often almost impossible to find an analytical solution. One of the alternative approaches is the random search method, in which a sequence of solution estimates is built iteratively, choosing a new estimate at each iteration based on the shift of the old one in a randomly chosen direction. This method is a direct development of the well-known trial and error method, when a solution is found by chance and, if successful, is accepted, and if it fails, it is rejected in order to immediately turn again to change as a source of opportunity. Such random behavior is reasonably based on the belief that randomness contains all possibilities, including the desired solution in all its variants. The random search method with optimal design makes it possible to determine the extremum of a function of a large number of variables with a relatively small amount of computer time.

The advantage of this method is that, in addition to the need for the existence of a single local extremum in the area under consideration, it does not impose significant requirements either on the type of the set of parameters by which the optimal value is found, or on the type of dependencies connecting the selected parameters with the optimizing criterion and restrictions. It allows you to find all local minima of a function of 10–20 variables with a complex relief. It is also useful when studying a function with a single minimum. This method has two advantages. First, it is suitable for any objective function, whether it is unimodal or not. Secondly, the probability of success in attempts does not depend on the dimension of the space under consideration. Although this method does not directly find the optimal solution, it creates suitable prerequisites for further application of other search methods. Therefore, it is often used in combination with one or more other types of methods.

The disadvantage of the method is that it is necessary to specify in advance the area in which random points are selected. If we set a region that is too wide, then it is more difficult to study it in detail, and if we choose a region that is too narrow, then many local minima may be outside of it.

A detailed description of the random variable method can be found in Rastrigin's book "Statistical Search Methods" [Rastrigin L. A., 376]

1.1.3 Genetic Optimization Algorithm Another example of a randomized algorithm is the genetic optimization algorithm. The first work on the simulation of evolution was carried out in 1954 by Nils Barricelli on a computer installed at the Institute for Advanced Study at Princeton University [Barricelli, 1954]. His work did not attract widespread public attention. Since 1957, the Australian geneticist Alex Fraser has published a series of papers simulating artificial selection among organisms with multiple controls for measurable characteristics. This groundbreaking allowed the computer simulation of evolutionary processes and the methods described in books by Fraser and Barnell and Crosby to become a more common activity among biologists from the 1960s. Fraser's simulations included all the essential elements of modern genetic algorithms. In addition to this, Hans-Joachim Bremermann published a series of papers in the 1960s that also took the approach of using a decision population subject to recombination, mutation, and selection in optimization problems. Bremermann's research also included elements of modern genetic algorithms.

Although Barricelli, in his 1963 paper [Barricelli, 1963], simulated the ability of a machine to play a simple game, artificial evolution became a well-established optimization technique after the work of Ingo Rechenberg and Hans-Paul Schwefel in the 1960s and early 1970s of the twentieth century – the group Rechenberg was able to solve complex engineering problems according to evolutionary strategies. Another approach was the evolutionary programming technique of Lawrence J. Vogel, which was proposed to create artificial intelligence.

With the growth of research interest, the computing power of desktop computers has also grown significantly, which made it possible to use new computer technology in practice. In the late 80s, General Electric began selling the world's first genetic algorithm product. They became a set of industrial computing tools. In 1989, another company, Axcelis, Inc. released Evolver, the world's first commercial genetic algorithm product for desktop computers [Aldawoodi, 2008].

The task is formalized in such a way that its solution can be encoded as a vector ("genotype") of genes, where each gene can be a bit, a number, or some other object. In classical implementations of the genetic algorithm, it is assumed that the genotype has a fixed length. However, there are variations of the genetic algorithm that are free from this limitation. In some, usually random, way, many genotypes of the initial population are created. They are evaluated using a "fitness function" whereby each genotype is associated with a specific value ("fitness") that determines how well the phenotype it describes performs the task.

From the resulting set of solutions ("generations"), taking into account the value of "fitness", solutions are selected (usually the best individuals have a higher probability of being chosen), to which "genetic operators" are applied (in most cases, "crossover" - crossover and "mutation" - mutation), resulting in new solutions. For them, the fitness value is also calculated, and then the best solutions are selected ("selection") for the next generation.

This set of actions is repeated iteratively, in this way an "evolutionary process" is modeled, lasting several life cycles (generations), until the algorithm's stopping criterion is met. This criterion could be:

1. finding a global or suboptimal solution,
2. exhaustion of the number of generations released for evolution,
3. exhaustion of time allotted for evolution.

1.1.4 Gossip Protocol

The gossip protocol also applies to randomized algorithms. The origin of this protocol is related to the epidemic replication algorithms described by Alan Demers, Dan Green, Carl Houser, Irish Wes, John Larson, Schenker Scott, Sturgis Howard, Swinhart Denm, and Terry Doug in their 1987 study Epidemic Algorithms for Replicated Database Maintenance. [Demers et al., 1987]. Since the appearance of this study, the spread of the epidemic has generated a lot of interest in computing. In fact, the first practical use of protocols like Gossip can be seen in the network routing systems that were the "preamble" of the Internet.

Suppose we want to find the object that most closely matches some search pattern in a network of unknown size, but where the nodes of the network are connected to each other and where each node runs a small agent program that implements the gossip protocol.

1. To start a search, the user had to ask the local agent to gossip about the search string.
2. Periodically, at some rate, each agent randomly selects another agent and gossips with him.
3. The first time a search string is received, each agent checks its local machine against documents.
4. Agents also gossip about the best match. Thus, if A gossips with B, after the interaction, A will learn about the best matches known to B, and vice versa. The best matches will "spread" through the network.

The gossip protocol has the following advantages:

1. the system can easily scale up to millions of processes,
2. restarting or downtime of any node on the network will not affect the operation of the gossip protocol,
3. any node can join or leave at any time without affecting the overall quality of service of the system,
4. The Gossip protocol provides exponentially fast information propagation, so when new information needs to be propagated, a message can be quickly sent to the global node and all nodes can have the latest data in a limited amount of time.

the system can easily scale up to millions of processes, a restart or downtime of any node on the network will not affect the operation of the gossip protocol, any node can join or leave at any time without affecting the overall quality of service of the system, the gossip protocol provides exponentially fast information dissemination, so when new information needs to be distributed, a message can be quickly sent to a global node and all nodes can have the latest data in a limited amount of time.

1.1.5 Ant Colony Algorithm

The original idea comes from observing ants as they find the shortest path from colony to food source [Dorigo et al., 1996].

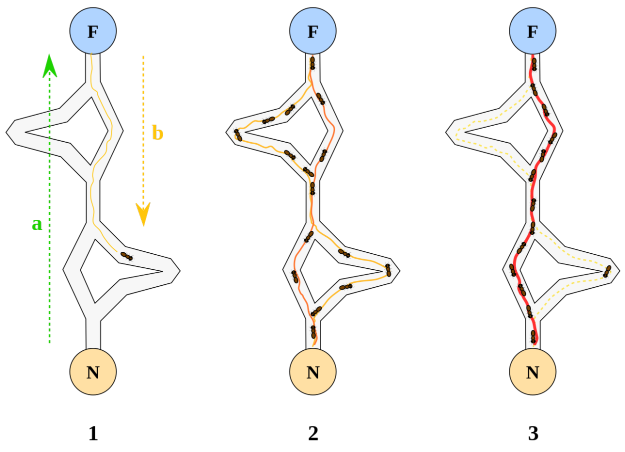

Figure 1. Ant algorithm

The first ant finds a food source (F) by any means (a) and then returns to the nest (N), leaving behind a trail of pheromones (b) 1. Then the ants choose one of the four possible paths, then strengthen it and make it attractive. Ants take the shortest route, as pheromones from longer paths evaporate faster. Among experiments on choosing between two paths of unequal length leading from a colony to a food source, biologists have noticed that ants tend to take the shortest route [Dorigo et al., 1996]. The model for this behavior is as follows:

The ant passes randomly from the colony. If it finds a food source, it returns to the nest, leaving a pheromone trail behind it. These pheromones attract other nearby ants that are more likely to take that route. Once back in the nest, they will reinforce the pheromone trail. If there are 2 routes, then more ants will have time to pass along the shorter one than along the long one. The short route will become more attractive. Long paths will eventually disappear due to the evaporation of pheromones. Ants use the environment as a means of communication. They exchange information indirectly, through pheromones, in the course of their "work". The exchange of information is local in nature: only those ants that are in the immediate vicinity of the pheromone trails can learn about them. Such a system is called stigmergy and is true for many social animals (it has been studied for the case of building pillars in termite nests). Such a mechanism for solving the problem is very complex and is a good example of the self-organization of the system. Such a system is based on positive (other ants strengthen the pheromone trail) and negative (evaporation of the pheromone trail) feedback. Theoretically, if the number of pheromones remains the same over time across all routes, then it will be impossible to choose a path. However, due to feedback, small fluctuations will force one of the paths and the system will stabilize towards the shortest path.

There are various variations of the ant colony algorithm:

1. elite ant system,
2. MMAS (Max-Min ant system),
3. ant ranking system (ASrank),
4. long-term orthogonal ant colony (COAC).

1.1.6 DNA-computing An example of another randomized algorithm is DNA computing. In 1994, Adleman showed [Adleman, 1994] how the solution to the problem of finding a Hamiltonian path in a graph can be reproduced using standard DNA operations already at that time: ligation, polymerase chain reaction, gel electrophoresis, biotin-streptavidin purification. All stages of DNA calculations are comparable to the steps in the enumeration method, although they are performed in a laboratory, and not on a computer: instead of the classical binary system, Adleman proposed using a four-letter code format corresponding to the four bases of DNA. In [Sergeenko et al., 2020], it was shown that the time required to perform DNA calculations for solving the problem of finding a Hamiltonian path in a graph is less than when solving the same problem using the branch and bound method, starting from a certain number of vertices and with sufficient sparseness of the connectivity matrix of the graph.

Another essential application of self-organization arises in DNA computing. Aiming to explain the concept of DNA computing, we consider a known combinatorial problem of determining a Hamiltonian path in a graph. As well known, the Hamiltonian Path problem consists of discovering a path in an undirected or directed graph precisely visiting each node once and starting with a given vertex and ending with another specified one. The most popular branch-and-bound method can hardly be used in many situations due to its exponentially growing complexity. The DNA computations based on the principles of self-organization inherent in nature [Adleman, 1994] are able to resolve such problems in linear time-consuming.

We remind that the DNA molecules usually form a double helix consisting of nucleotides containing a phosphate group, a sugar group, and a nitrogen base. There are four different nucleotides: adenine ("A"), thymine ("T"), guanine ("G"), cytosine ("C"). In the helix, the nucleotides join in pairs according to a fundamental *complementarity rule*: "A" is always opposite to "T", and "G" is always opposite to "C" [Watson and Crick, 1953]. DNA computing allows modeling in a biological laboratory dealing with double-stranded DNA, which are able to "split" into two separate strands (by heating) or to do reverse action (*ligation*) with the help of the particular enzyme *ligase*.

Let us consider a directed graph $G = (\mathcal{N}, E)$ with $|\mathcal{N}| = n$ nodes. Random DNA sequences consisting of 20 nucleotides represent nodes (vertexes). An edge is associated with DNA sequence obtained as a concatenation of the complement of the second half of the starting node and the complement of the first half of the finishing node.

For example, if the first node has the DNA code

$$\begin{aligned} &TCAGTACCAG \quad TACAGTCACA \\ complement: (&AGTCATGGTC \quad ATGTCAGTGT), \end{aligned}$$

where A is adenine, T is thymine, G is guanine, C is cytosine, the second node has the DNA code

$$\begin{aligned} &TAGGTATGCT \quad CAGATAAAGG \\ complement: (&ATCCATACGA \quad GTCTATTTCC), \end{aligned}$$

and there is an edge from the first node to the second, then that edge is coded by

$$ATGTCAGTGT \quad ATCCATACGA.$$

This procedure is repeated with every available edge. Note that the directions of DNA strands are ignored for simplicity.

Further, all described DNA representations are created in a lab and mixed in a single ligation reaction connecting the molecules according to the complementary rule. All available pathways are simultaneously in parallel created in one probe. An example of a short molecule is shown in Figure 2.

An analogy with multiagent technologies is undoubtedly evident in this process. By forming strands and units into massive DNA molecules, as if they were agents within a cascade process, nucleotides provide a

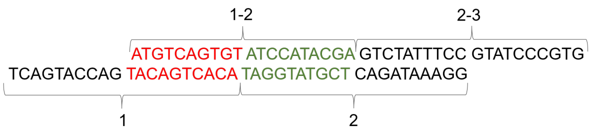
1-2
2-3
ATGTCAGTGT ATCCATACGA GTCTATTTCC GTATCCCGTG
TCAGTACCAG TACAGTCACA TAGGTATGCT CAGATAAAGG
1
2

Figure 2. An example of a molecule after ligation reaction

solution to the problem. The subsequent steps intend to check if the molecules encoding the Hamiltonian path are within the strand or not. The time consumption of DNA computing results from local interactions and clustering growing linearly as the number of nodes. Subsequently, in [Sergeenko et al., 2020], this characteristic is compared with the same one appearing in the branch-and-bound method. It is turned out that that DNA computing performs much better for the adjacency matrix sparsity parameters equaling 0.85, 0.9, 0.8 of and the number of graph nodes rising from 37, 43, 45, respectively.

1.1.7 Simultaneous Perturbation Stochastic Approximation Problem of function minimization is being solved in many applications. Sometimes the extreme values of a function can be found theoretically. In general, engineering systems have to deal with unknown functions, and it is only possible to measure its values in some points.

Let $F(x, w) : \mathbb{R}^q \times \mathbb{R}^p \Rightarrow \mathbb{R}^1$ be the differentiable on the first argument function, $x_1, x_2, \ldots$ be a sequence of arguments of F chosen during optimization procedure at each iteration $n = 1, 2, \ldots$ (the design of an experiment),$\{w_n\}$ is an uncontrollable sequence of random values from $\mathbb{R}^p$ with identical but unknown distribution $\mathbb{P}_w(\cdot)$ which has the finite support. The function $F(\cdot, w_n)$ can be observed with the added noise v_n: $y_n = F(x_n, w_n) + v_n$. The problem is to minimize the function $f(x) = E_w F(x, w) = \int_{\mathbb{R}^p} F(x, w)\mathbb{P}_w(dw)$ based on observations $y_1, y_2, \ldots$. This means to build the sequence of estimates $\hat{\theta}_n$ of unknown vector $\theta_\star$, which minimizes $f(x)$.

Various gradient methods can be used. However, we would like to introduce one optimization method that has attracted considerable international attention, which is Simultaneous Perturbation Stochastic Approximation (SPSA) Unlike other methods, which requires direct measurements of the gradient of the objective function (which are often difficult or impossible to obtain), SPSA needs only two objective function measurements per iteration regardless of the dimension of the optimization problem. Further, SPSA is especially efficient in high-dimensional problems in terms of providing a good solution for a relatively small number of measurements of the objective function. If the number of terms being optimized is p, then the finite-difference method takes $2p$ measurements of the objective function at each iteration (to form one gradient approximation) while SPSA takes only two measurements.

Let the trial simultaneous perturbation $\Delta_n, n = 1, 2, \ldots$, be a random sequence of zero-mean independent vectors from $\mathbb{R}^q$ with distributions $\mathbb{P}_n(\cdot), n = 1, 2, \ldots$, which have a uniformly bounded finite support and independent components. Consider sequences of real positive numbers $\{\alpha_m\}$ and $\{\beta_n\}$. Choose some initial vector $\hat{\theta}_0 \in \mathbb{R}^q$. In [Granichin and Polyak, 2003] the algorithm with two simultaneous perturbations was proposed for construction of sequences of measurement points and estimates:

$$\begin{cases} x_n^\pm = \hat{\theta}_{n-1} + \beta_n \Delta_n, \\ y_n^\pm = F(x_n^\pm, w_n^\pm) + v_n^\pm, \\ \hat{\theta}_n = \hat{\theta}_{n-1} = \alpha_n \Delta_n \frac{y_n^+ - y_n^-}{2\beta_n}. \end{cases} \tag{1}$$

In [Granichin and Amelina, 2015], the SPSA algorithm has been applied to an unconstrained problem of (mean-square) optimal target tracking. It has been shown that SPSA converges even in the presence of an arbitrary unknown-but-bounded noise (typically, tracking algorithms are suitable only for noises with zero mean [Zhu and Spall, 2018]).

1.2 Multi-agent Algorithms

There is a whole class of algorithms inspired by nature. In recent years, such optimization algorithms have gained recognition in the field of machine learning for finding optimal solutions to complex problems in science and technology. To solve the problems of traditional optimization algorithms, recent trends tend to apply nature-inspired optimization algorithms, which are a promising approach to solving complex optimization problems. Recent such algorithms include: Genetic Bee Colony Algorithm (GBC), Fish Swarm Algorithm (FSA), Cat Swarm Optimization Algorithm (CSO), Whale Optimization Algorithm (WOA), Artificial Algae Algorithm (AAA), Elephant Search Algorithm (ESA), chicken swarm optimization (CSOA), moth flame optimization (MFO), and gray wolf optimization (GWO) algorithm. The ant colony algorithm belongs to the same type of algorithms.

All algorithms inspired by nature have one thing in common: numerous living beings or "agents" whose task is to find the optimal solution. On the other hand, a multi-agent system is a system formed by several interacting intelligent agents. Thus, we can conclude that nature-inspired algorithms are multi-agent algorithms.

The main advantage of multi-agent systems is flexibility. The multi-agent system can be supplemented and modified without rewriting a significant part of the program. Also, these systems have the ability to self-heal and are resistant to failures, thanks to a sufficient supply of components and self-organization.

Consider one of the examples of multi-agent algorithms. A recent scientific breakthrough has shown [Rutherford and Bassler, 2012] that bacteria can communicate with each other. This fact has fundamentally changed the general idea of the existence of simple organisms inhabiting the world. Bacteria use signaling molecules to measure population concentration. The current term "quorum sensing" (QS) describes the signaling molecule process that allows a single cell to sense bacterial activity at the expense of cell density. In nature, different types of bacteria in the same environment use a variety of signaling molecules that allow interaction with other different types of bacteria. Today, these quorum systems are being intensively studied for various categories of bacteria. Recently, extraordinary advances in understanding genetics, genomics, biochemistry and QS signal diversity, including information on links between QS and bacterial sociality. The behavior and evolution of these bacterial communities are perceived as natural examples of multi-agent systems, since interaction between bacteria occurs locally and the density of bacteria can be measured without having to collect all the data in one data center. These bacteria decide the global task (measurement of population density) using only local communication and, so this is an example of multi-agent technologies.

Multi-agent algorithms also appear in social networks [Proskurnikov and Tempo, 2018].

1.2.1 Local Voting Protocol (LVP). Another example of multi-agent algorithm is the Local Voting Protocol. It was studied in [Amelina et al., 2015a] and in [Amelina et al., 2013], where the applicability of the local voting protocol with nonvanishing step-size for decentralized stochastic network load balancing was studied under non-stationary problem formulation. It was shown that LVP working on the principles of the multi-agent algorithms proves its flexibility even if the network is considered to have a switched topology, and the control strategy uses noisy and delayed measurement

2 Multi-sensor Multi-target Problem

2.1 Preliminaries

Let $(\Omega, \mathcal{F}, P)$ be the underlying probability space corresponding to sample space Ω, set of all events $\mathcal{F}$, and probability measure P. $\mathbb{E}$ denotes mathematical expectation.

Let $[\cdot]^{\mathrm{T}}$ be vector or matrix transpose operation, $[\cdot]^{-1}$ be matrix inversion. $\|A\|$ is the Frobenius norm: $\|A\| = \sqrt{\sum_i \sum_j (a^{i,j})^2}$. $\mathbf{1}_d = [1, \ldots, 1]^{\mathrm{T}} \in \mathbb{R}^d$ is the vector of all ones. $\mathbf{e}_i = [\ldots, 0, 1, 0, \ldots]^{\mathrm{T}} \in \mathbb{R}^d$ is the canonical basis vector from $\mathbb{R}^d$, where i-th entry is equal to 1. $I_d \in \mathbb{R}^{d\times d}$ is the identity matrix, $\mathbb{J}_d = \mathbf{1}_d \mathbf{1}_d^{\top} \in \mathbb{R}^{d\times d}$ is the matrix of all ones. $A \otimes B$ is the Kronecker product defined for any matrices A and B. The following notation $A \leq B$ means that matrices are ordered in the sense of quadratic forms.

For a sequence of column vectors $x_1, \ldots, x_k$, let $\mathrm{col}\{x_1, \ldots, x_k\}$ denote the column vector obtained by stacking x_i on top of one another.

A network consisting of n nodes is described by a directed graph $\mathcal{G}_A = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, \ldots, n\}$ is a set of vertices and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is a set of edges. Denote by $i \in \mathcal{N}$ an identifier of i-th node and $(j, i) \in \mathcal{E}$ if there is a directed edge from node j to node i. The latter means that node j is able to transmit data to node i. For a node $i \in \mathcal{N}$, the set of *neighbors* is defined as $\mathcal{N}^i = \{j \in \mathcal{N} : (j, i) \in \mathcal{E}\}$. The *in-degree* of $i \in \mathcal{N}$ equals $|\mathcal{N}^i|$. Hereinafter, $|\cdot|$ is the cardinality of a set, and superscripts stand for agents' indices.

Let $a^{i,j} > 0$ be the weight associated with the edge $(j, i) \in \mathcal{E}$ and $a^{i,j} = 0$ whenever $(j, i) \notin \mathcal{E}$. Let $A = [a^{i,j}]$ be the *weighted adjacency matrix*, or simply *connectivity matrix*, associated with graph $\mathcal{G}_A$. The *weighted in-degree* of $i \in \mathcal{N}$ is defined as $\deg_i^+(A) = \sum_{j=1}^n a^{i,j}$, the maximum in-degree among all nodes contained in graph $\mathcal{G}_A$ as $\deg_{\max}^+(A)$. Introducing the diagonal matrix $\mathcal{D}(A) = \mathrm{diag}_n(\deg_1^+(A), \ldots, \deg_n^+(A))$, matrix $\mathcal{L}(A) = \mathcal{D}(A) - A$ is referred to as the *Laplacian* of graph $\mathcal{G}_A$.

Definition 1. A directed graph $\mathcal{G}_A$ is said to be strongly connected if for every pair of nodes $j, i \in \mathcal{N}$, there exists a path of directed edges that goes from j to i.

Denote the eigenvalues of Laplacian $\mathcal{L}(A)$ by $\lambda_1, \ldots, \lambda_n$ and arrange them in ascending order of real parts: $0 \leq Re(\lambda_1) \leq Re(\lambda_2) \leq \ldots \leq Re(\lambda_n)$. It is known, that if the graph is strongly connected then $\lambda_1 = 0$ and all other eigenvalues of $\mathcal{L}$ are in the open right half of the complex plane (see, e.g., [Lewis et al., 2013]). The eigenvalue of matrix A with maximum absolute magnitude is defined as $\lambda_{\max}(A)$.

2.2 Problem Statement

We consider a network of n spatially-distributed sensors in a field, namely, agents, capable of measuring parameters (e.g., distance, heading, etc), performing local computations, and exchange information with neighboring nodes. In this field, there are m moving targets. Each sensor $i \in \mathcal{N} = \{1, \ldots, n\}$ has its own hypothesis regarding the state of the targets (i.e., their positions) or more simply an estimate of the states. The goal of the network is to accurately estimate the unknown parameters of the targets. The sensors must also act together as a team to achieve this goal.

Let $\mathbf{s}_t^i = [s_t^{i,1}, \ldots, s_t^{i,d}]^{\mathrm{T}} \in \mathbb{R}^d$ be the state of sensor i at time instant t, $\mathbf{r}_t^l = [r_t^{l,1}, \ldots, r_t^{l,d}]^{\mathrm{T}} \in \mathbb{R}^d$ be the state of target $l \in \mathcal{M} = \{1, \ldots, m\}$, and $\theta_t = \mathrm{col}\{\mathbf{r}_t^1, \ldots, \mathbf{r}_t^m\}$ be the vector consisting of all states to be estimated. Suppose that each sensor measures a scalar quantity, which is the distance between its own position and position of a target:

$$\rho(\mathbf{s}_t^i, \mathbf{r}_t^l) = ||\mathbf{r}_t^l - \mathbf{s}_t^i||^2, \;\; \forall i \in \mathcal{N}, l \in \mathcal{M}. \qquad (2)$$

Note that the proposed approach can be used for other types of measuring parameters (e.g., bearing/azimuth).

In general, the problem is to find an estimate $\hat{\theta}_t$ of an unknown parameter θ_t:

$$\hat{\theta}^\star_t = \arg\min_{\hat{\theta}_t} ||\hat{\theta}_t - \theta_t||^2. \tag{3}$$

3 Consensus-based Distributed SPSA Algorithm

In section 1, many pluses of randomized and multi-agent algorithms were proposed. In order to use advantages of both algorithms, they need to be combined. So, this idea was implemented for SPSA and Local Voting Protocol in [Amelina et al., 2020], where a consensus-based distributed SPSA algorithm for multi-target tracking was proposed.

3.1 Basic Requirements for the Algorithm

In [Amelina et al., 2020], a more difficult problem setting was considered than the one presented in section 2.2. The following requirements for the new algorithm were introduced:

1. the solution of the optimization problem (3) needs to be found in a distributed way;
2. the following *communication constraints* are imposed: at time instant t, each sensor $i \in \mathcal{N}$ is able to measure the squared distance to not more than one target. In practice, due to certain constraints, the number of communication channels that can be used is usually less than the dimension of space or equal to it. Without loss of generality, in this paper, we assume that each sensor is able to collect data only from d neighbors. In this case, and if there is no noise, we can use standard triangular approaches to determine the target position. However, if positions of all m targets need to be computed, then we have to simultaneously collect $m(d-1)$ measurements, and it is often impossible in practice;
3. the *unknown-but-bounded* noise is considered to be involved in the measuring process.

3.2 Strong Communication Constraints

In [Sergeenko et al., 2020], stronger communication constraints were introduced, and parameter optimization was provided.

In that paper, the problem setting of [Amelina et al., 2020] was generalized and a situation in which the number of neighboring agents may be less than the dimension of space (it was strictly equal in [Amelina et al., 2020]) was considered. The following point was changed in the requirements:

2*. each sensor is able to collect data only from p neighbors, where $p \leq d$. In this case, communication channels become even less loaded, which is essential for using this algorithm in a real life.

More formal problem setting for the requirements $1, 2^*, 3$ is presented further.

3.2.1 Measurements

We assume that at time instant t sensor i is able to measure the squared distance

$$\rho(\mathbf{s}^i_t, \mathbf{r}^l_t) = \|\mathbf{r}^l_t - \mathbf{s}^i_t\|^2 = \sum_{d'=1}^{d} \left(r^{l,d'}_t - s^{i,d'}_t\right)^2$$

to moving target $\mathbf{r}^l_t$. Note, the proposed approach can be used for other sensing modalities (e.g., bearing/azimuth). Consider that sensor i gets similar data from p other sensors $j_1, \dots, j_p \in \mathcal{N}^i$, which are its neighbors, about the state $\mathbf{r}^l_t$. For each such column $\mathbf{u} = \mathrm{col}\{i, j_1, \dots, j_p, l\}$ of $(p+2)$ naturals denote $\bar{\rho}^q_t(\mathbf{u}) = \rho(\mathbf{s}^i_t, \mathbf{r}^{l(\mathbf{u})}_t) - \rho(\mathbf{s}^{j_q}_t, \mathbf{r}^{l(\mathbf{u})}_t),\ q = 1, \dots, p$. Here and after, $l(\mathbf{u})$ is the map defining the last component of $\mathbf{u}$. In this case, there are p equations

$$\bar{\rho}^q_t(\mathbf{u}) = \sum_{d'=1}^{d} (s^{j_q,d'}_t - s^{i,d'}_t)(2r^{l(\mathbf{u}),d'}_t - s^{j_q,d'}_t - s^{i,d'}_t),$$

$q = 1, \dots, p$, which allow us to derive

$$C^{\mathbf{u}}_t \mathbf{r}^{l(\mathbf{u})}_t = D^{\mathbf{u}}_t \Rightarrow C^{\mathbf{u}\,\mathrm{T}}_t C^{\mathbf{u}}_t \mathbf{r}^{l(\mathbf{u})}_t = C^{\mathbf{u}\,\mathrm{T}}_t D^{\mathbf{u}}_t \Rightarrow$$
$$I^{\mathbf{u}}_t \mathbf{r}^{l(\mathbf{u})}_t = H^{\mathbf{u}}_t, \tag{4}$$

where $I^{\mathbf{u}}_t = [C^{\mathbf{u}\,\mathrm{T}}_t C^{\mathbf{u}}_t]' C^{\mathbf{u}\,\mathrm{T}}_t C^{\mathbf{u}}_t, H^{\mathbf{u}}_t = [C^{\mathbf{u}\,\mathrm{T}}_t C^{\mathbf{u}}_t]' C^{\mathbf{u}\,\mathrm{T}}_t D^{\mathbf{u}}_t$,

$$C^{\mathbf{u}}_t = 2\begin{bmatrix} (\mathbf{s}^{j_1}_t - \mathbf{s}^i_t)^{\mathrm{T}} \\ \cdots \\ (\mathbf{s}^{j_p}_t - \mathbf{s}^i_t)^{\mathrm{T}} \end{bmatrix}, \quad D^{\mathbf{u}}_t = \begin{bmatrix} \bar{\rho}^1_t(\mathbf{u}) + \|\mathbf{s}^{j_1}_t\|^2 - \|\mathbf{s}^i_t\|^2 \\ \cdots \\ \bar{\rho}^p_t(\mathbf{u}) + \|\mathbf{s}^{j_p}_t\|^2 - \|\mathbf{s}^i_t\|^2 \end{bmatrix}.$$

Using the introduced notations, the measurements of sensor $i \in \mathcal{N}$ at time instant t are defined as follows:

$$y^i_t = F^i_t(\mathbf{u}^i_t, \mathbf{x}^i_t) + v^i_t = \|\hat{\mathbf{r}}^{h(\mathbf{u}^i_t)}_t - \mathbf{r}^{h(\mathbf{u}^i_t)}_t\|^2 + v^i_t, \tag{5}$$

where v^i_t is the unknown-but-bounded additive noise, $\mathbf{x}^i_t$ is the measurement point depending on currently available estimate $\hat{\mathbf{r}}^{l(\mathbf{u}^i_t)}_t$ at time instant t. For example, $\mathbf{x}^i_t = \hat{\mathbf{r}}^{l(\mathbf{u}^i_t)}_t$.

3.2.2 Distributed Optimization

Denote by $\mathcal{F}_{t-1}$ the σ-algebra of all probabilistic events, which happened up to time instant t. $\mathbb{E}_{\mathcal{F}_{t-1}}$ denotes the conditional expectation with respect to the σ-algebra $\mathcal{F}_{t-1}$. This σ-algebra is generated by the values of all random variables (i.e., position of targets, noise, changes in communication topology) at time instants $\tau = \{1, 2, \dots, t\}$.

Let $\mathbf{u}_t = [\mathbf{u}^1_t, \dots, \mathbf{u}^n_t]^{\mathbf{T}}$ be the common vector defining the sets of neighbors used to collect measurements from each sensor. The multi-sensor multi-target problem

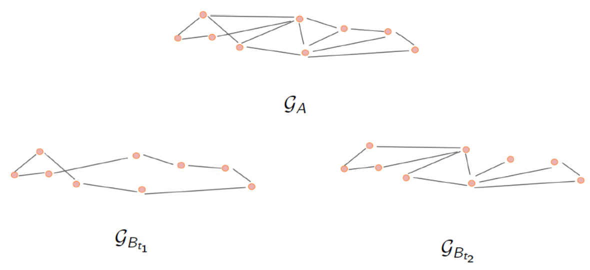

Figure 3. Communication graph $\mathcal{G}_A$, which describes all possible connections between sensors, and randomly chosen subgraphs $\mathcal{G}_{B_t}, t = \{t_1, t_2\}$, where each sensor has a limited number of neighboring sensors. As it seen, the connections may change with time.

can be formulated as the following minimization problem: to find estimate $\hat{\theta}_t = \mathrm{col}\{\hat{\mathbf{r}}_t^{l(\mathbf{u}_t^1)}, \dots, \hat{\mathbf{r}}_t^{l(\mathbf{u}_t^m)}\}$ that minimizes the following loss function

$$\bar{F}_t(\hat{\theta}_t, \mathbf{u}_t) = \mathbb{E}_{\mathcal{F}_{t-1}} \sum_{i\in\mathcal{N}} F_t^i(\mathbf{u}_t^i, \hat{\mathbf{r}}_t^{l(\mathbf{u}_t^i)}) \to \min_{\hat{\theta}_t}. \quad (6)$$

Usually, during optimization, each sensor fuses the needed information from all available neighboring nodes. In this problem setting, the communication constraint that prohibit such communication strategy of the sensors were mentioned. These communication constraints arise due to hardware and physical limitations since the bandwidths of communication channels are not unlimited. When a large number of sensors send and receive messages at the same time, communication becomes a bottleneck. To deal with this, communication links between sensors were chosen randomly. More formally, for each sensor $i \in \mathcal{N}$, the communication topology described by graph $\mathcal{G}_A$ was randomized at each time instant t to satisfy topology constraints such as the maximum number of links equals to p. A randomly chosen subgraph $\mathcal{G}_{B_t} \subset \mathcal{G}_A$ associated with adjacency matrix $B_t = [b_t^{i,j}]$ was used, where the rows contain no more than p nonzero entries (see Figure 3). Afterwards, the observable target at time instant t contained in $\mathbf{u}_t^i$ is generated from a uniform distribution independently for each sensor $i \in \mathcal{N}$ as in gossip algorithm, described in section 1.1.4. The communication topology described by graph $\mathcal{G}_A$ was randomized based on the strategy similar to one presented in [Amelina et al., 2014].

3.3 Weighted Version of the Algorithm

In [Erofeeva et al., 2021], a weighted version of the consensus-based distributed SPSA algorithm was proposed, taking into account the heterogeneity of targets. In practice, one may need to track a group of targets consisting of fixed-wing drones and rotor ones, which have different dynamics and speed. Furthermore, the covariance matrix of the residuals was evaluated, extending the approaches proposed in [Polyak, 1977,Granichin, 2004].

Let $\mathbf{u}_k^i$ and $\mathbf{\Delta}_k^i \in \mathbb{R}^d$, $k = 1, 2, \dots,\ i \in \mathcal{N}$, be independent random variables. We generate $\mathbf{\Delta}_k^i$ called the *simultaneous test perturbation* from Bernoulli distribution with each component independently taking values $\pm\frac{1}{\sqrt{d}}$ with probabilities $\frac{1}{2}$. Let $\mathbf{e}_{l(\mathbf{u}_k^i)} \in \mathbb{R}^m$ be the sparse vector corresponding to the current target that sensor i observes, then $\hat{\Delta}_k^i = \mathbf{e}_{l(\mathbf{u}_k^i)} \otimes \mathbf{\Delta}_k^i$. In this case, $\hat{\Delta}_k^i$ is the vector of all zeros except for the rows that corresponds to $l(\mathbf{u}_k^i)$.

Let $\mathbb{U}^{i,l}$ be a set containing all possible subsets $\bar{\mathcal{N}}_t^i$ for target l. The neighborhood of sensor i at time instant t is defined by the i-th row of matrix B_t associated with graph $\mathcal{G}_{B_t}$. This row is defined by subset $\bar{\mathcal{N}}_t^i$ generated from the uniform distribution on the set $\mathbb{U}^{i,l}$.

Next, a weighted version of consensus-based SPSA distributed algorithm is presented. Let us choose the diagonal matrix $\Psi = [\psi_{ij}]$, where $\psi_{ij} > 0$ if $i = j$ and $\psi_{ij} = 0$ otherwise. At initialization step, for each $i \in \mathcal{N}$, we choose initial vector $\widehat{\theta}_0^i \in \mathbb{R}^{md}$, positive step-size α, matrix Ψ, gain coefficient γ, and the scale of perturbation $\beta > 0$.

In order to get estimates $\{\widehat{\theta}_t^i\}$ of overall state vectors $\{\theta_t^i\}$ based on measurement points $\{\mathbf{x}_t^i\}$, the weighted algorithm with two measurements of distributed subfunctions $F_t^i(\mathbf{u}_t^i, \mathbf{x}_t^i)$ was proposed:

$$\begin{cases} \mathbf{x}_{2k}^i = \widehat{\theta}_{2k-2}^i + \beta\hat{\Delta}_k^i,\ \mathbf{x}_{2k-1}^i = \widehat{\theta}_{2k-2}^i - \beta\hat{\Delta}_k^i, \\ \widehat{\theta}_{2k-1}^i = \widehat{\theta}_{2k-2}^i, \\ \widehat{\theta}_{2k}^i = \widehat{\theta}_{2k-1}^i - \alpha\Psi\Big[\hat{\Delta}_k^i \frac{y_{2k}^i - y_{2k-1}^i}{2\beta} + \\ \qquad\qquad \gamma\sum_{j\in\bar{\mathcal{N}}_{2k-1}^i} b_{2k-1}^{i,j}(\widehat{\theta}_{2k-1}^i - \widehat{\theta}_{2k-1}^j)\Big]. \end{cases} \quad (7)$$

Consider the last equation of the algorithm (7): the first part is similar to SPSA-like algorithm from [Granichin and Amelina, 2015] (section 1.1.7) and the second one coincides with Local Voting Protocol (LVP) from [Amelina et al., 2015b] (section 1.2.1), where it was studied for stochastic networks in the context of load balancing problem. The SPSA part represents a stochastic gradient descent of sub-functions $F_t^i(\mathbf{u}_t^i, \mathbf{x}_t^i)$, and LVP part is determined for each agent i by the weighted sum of differences between the information about the current estimate $\widehat{\theta}_{2k-1}^i$ of the agent i and available information about the estimates of its neighbors. Thus, the algorithm is the combination of randomized and multi-agent approaches.

3.4 Acceleration of SPSA

In [Erofeeva et al., 2022], the line of the previous works was continued. One of the biggest disadvantage of all stochastic algorithms, which include SPSA, is the slow convergence. So, the accelerated version of SPSA was introduced in [Erofeeva et al., 2022], where SPSA is combined with Nesterov acceleration method [Kosaty et al., 2019].

The advantages of such combination are the following:

1. it has faster convergence than the original SPSA algorithm;
2. it is applicable for the non-stationary problem statement;
3. the convergence does not depend on the type of noise, as long as it is bounded.

Let $\mathbf{\Delta}_n \in \mathbb{R}^d$, $n = 1, 2, \dots$ be independent random variable, i.e., simultaneous test perturbation, drawn from Bernoulli distribution. Each component of the vector independently takes value $\pm\frac{1}{\sqrt{d}}$ with probability $\frac{1}{2}$. Let us choose initial estimate $\hat{\theta}_0 \in \mathbb{R}^d$, and parameters $\gamma_0 > 0$, $h > 0$, $\beta > 0$, $\eta \in (0, \mu)$, $\alpha_0 \in (0, 1)$. Also, let us define $z_0 = \hat{\theta}_0$ and $H = h - \frac{h^2 L}{2}\left[(\frac{a}{2\beta} + 1)^2 + \frac{\epsilon^2}{2}\right]$, where $\epsilon > 0$. At each n, α_n can be found using the equation

$$\alpha_n^2 = 2(\frac{2\beta H}{a} - \frac{\epsilon}{2})((1 - \alpha_n)\gamma_n + \alpha_n(\mu - \eta))$$

and $\gamma_n = (1 - \alpha_{n-1})\gamma_{n-1} + \alpha_{n-1}(\mu - \eta)$.

The algorithm with two observations of functions $f_{\xi_t}(\cdot)$ is considered for constructing sequences of measurement points $\{\mathbf{x}_t\}$ and estimates $\{\widehat{\theta}_t\}$ at $n \geq 1$:

$$\begin{cases} \tilde{\mathbf{x}}_{2n-2} = \frac{1}{\gamma_{n-1}+\alpha_n(\mu-\eta)}\Big(\alpha_n\gamma_{n-1}\mathbf{z}_{2n-2} + \gamma_n\widehat{\theta}_{2n-2}\Big), \\ \mathbf{x}_{2n} = \tilde{\mathbf{x}}_{2n-2} + \beta\mathbf{\Delta}_n,\ \mathbf{x}_{2n-1} = \tilde{\mathbf{x}}_{2n-2} - \beta\mathbf{\Delta}_n, \\ \tilde{\mathbf{x}}_{2n-1} = \tilde{\mathbf{x}}_{2n-2},\ \widehat{\theta}_{2n-1} = \widehat{\theta}_{2n-2}, \\ \mathbf{g}_{2n} = \mathbf{\Delta}_n\frac{y_{2n}-y_{2n-1}}{2\beta}, \\ \widehat{\theta}_{2n} = \tilde{\mathbf{x}}_{2n-1} - h\mathbf{g}_{2n}, \\ \mathbf{z}_{2n} = \gamma_n^{-1}\Big[(1-\alpha_n)\gamma_{n-1}\mathbf{z}_{2n-2} + \\ \alpha_n(\mu-\eta)\tilde{\mathbf{x}}_{2n-1} - \alpha_n\mathbf{g}_{2n})\Big]. \end{cases} \tag{8}$$

4 Simulation

In this section, we present a numerical experiment, which illustrates the performance of the algorithm (8) combined with the Local Voting Protocol as in (7) to make Accelerated consensus-based SPSA algorithm which has advantages of both randomized and multi-agent algorithms but also converges faster.

We define a distributed network of 9 sensors tracking 20 moving targets. Each sensor may have only one active communication channels for the information exchange. Each sensor also choose a random target that it tracks at the current time instant.

Let $\theta_t = \text{col}(\mathbf{r}_t^1, \dots, \mathbf{r}_t^{20}) \in \mathbb{R}^{40}$ be the common state vector of all targets, $\hat{\theta}_t = \text{col}(\hat{\mathbf{r}}_t^1, \dots, \hat{\mathbf{r}}_t^9) \in \mathbb{R}^{18}$ be a common vector of estimates. Each target $l \in \mathcal{M}$ changes the position according to the following dynamics:

$$\mathbf{r}_t^l = \mathbf{r}_{t-1}^l + \zeta_{t-1}^l, l \in \mathcal{M}, \tag{9}$$

where ζ_{t-1}^l are random vectors uniformly distributed in a ball. We've defined ζ_t^l as a random vector uniformly distributed on the ball of radius equal to 0.2 for targets with odd identifiers and 0.6 for targets with even identifiers. This means that the targets are heterogeneous and behave differently. The targets start their motion at a position randomly chosen from the interval $[100; 300]$. The sensors are stationary and their coordinates are random values uniformly distributed in interval $[20; 25]$. We consider random type of noise, i.e. uniformly distributed random variable falling within the interval $[-1; 1]$. We have set the following parameters of the algorithm: $h = 0.08, \beta = 0.1, \eta = 0.95, \alpha_x = 0.1, \gamma_0 = 2.0$, $L = 2, \mu = 2, a = 2, b = 2, c = 1, \gamma = 1, \Psi = \text{diag}_{20}(\text{col}\{2, 10, \dots, 2, 10\})$.

We consider three types of noise: uniformly distributed random variable falling within the interval $[-1; 1]$ (1), an unknown constant (2), and hybrid noise which is uniformly distributed around constants that change with time, e.g. $v_k^i = \pm 1 + 0.1 * sin(k)$, where the sign in front of 1 switches each 50-th iteration (3).

In the simulation, the new algorithm is compared with the previous one from [Erofeeva et al., 2021]. Figure 4 shows the behavior of the residual obtained by the Accelerated consensus-based SPSA algorithm and by the consensus-based SPSA algorithm for various types of noise. Both presented algorithms have the same initial parameters, random values of targets and noises at each iteration. The only difference is the algorithm itself. It is well seen that the new algorithm converges faster than the previous one: while the new algorithm is converged approximately by step 100, the old one converges approximately by step 400. Moreover, the convergence does not depend on the noise.

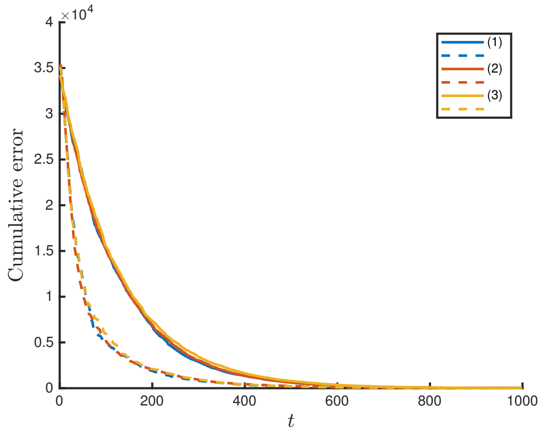

Figure 4. Typical behavior of the residuals. The blue lines indicate random noise, red – unknown constant, yellow – oscillations. The solid line is the algorithm from [Erofeeva et al., 2021], the red one shows the proposed new accelerated version.

5 Conclusion

In this paper, a description of randomized and multi-agent algorithms was presented. Different combined algorithms, which are applicable for the multi-sensor multi-target tracking problem were shown. The new algorithm, which is an Accelerated consensus-based SPSA algorithm, was validated through the simulation. It was shown that it converges faster than non-accelerated one. Furthermore, the convergence does not depend on the type of noise.

Acknowledgements

Theoretical results in Sections 1–3 were supported by the Institute of Problems of Mechanical Engineering RAS by Russian Science Foundation (project no. 21-19-00516). Experimental results in Section 4 was supported by the St. Petersburg State University (project ID 93024916).

References

Adleman, L. (1994). Molecular computation of solutions to combinatorial problems. *Science, New Series*, **266** (5187), pp. 1021–1024.

Aldawoodi, N. (2008). An approach to designing an unmanned helicopter autopilot using genetic algorithms and simulated annealing.

Amelina, N., Erofeeva, V., Granichin, O., Ivanskiy, Y., Jiang, Y., Proskurnikov, A., and Sergeenko, A. (2020). Consensus-based distributed algorithm for multisensor-multitarget tracking under unknown-but-bounded disturbances. In *Proc. of IFAC World Congress 2020*.

Amelina, N., Fradkov, A., Jiang, Y., and Vergados, D. (2015a). Approximate consensus in stochastic networks with application to load balancing. *IEEE Trans. Inform. Theory*, **61** (4), pp. 1739–1752.

Amelina, N., Fradkov, A., Jiang, Y., and Vergados, D. J. (2015b). Approximate consensus in stochastic networks with application to load balancing. *IEEE Transactions on Information Theory*, **61** (4), pp. 1739–1752.

Amelina, N., Granichin, O., Granichina, O., and Jiang, Y. (2014). Differentiated consensuses in decentralized load balancing problem with randomized topology, noise, and delays. In *53rd IEEE Conference on Decision and Control*, IEEE, pp. 6969–6974.

Amelina, N., Granichin, O., and Kornivetc, A. (2013). Local voting protocol in decentralized load balancing problem with switched topology, noise, and delays. In *52nd IEEE Conference on Decision and Control*, IEEE, pp. 4613–4618.

Barricelli, N. A. (1954). Esempi numerici di processi di evoluzione. *Methodos*, pp. 45–68.

Barricelli, N. A. (1963). Numerical testing of evolution theories. part ii. preliminary tests of performance, symbiogenesis and terrestrial life. *Acta Biotheoretica*, p. 99–126.

Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., and Terry, D. (1987). Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '87, New York, NY, USA, Association for Computing Machinery, p. 1–12.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **26** (1), pp. 29–41.

Ermakov S.M. (1971). *Monte Carlo method and related issues*. Nauka, Moscow.

Erofeeva, V., Granichin, O., Proskurnikov, A., and Sergeenko, A. (2021). Weighted spsa-based consensus algorithm for distributed cooperative target tracking. In *Proc. of the 2021 European Control Conference*.

Erofeeva, V., Granichin, O., Sergeenko, A., Tursunova, M., and Jiang, Y. (2022). Accelerated simultaneous perturbation stochastic approximation for tracking under unknown-but-bounded disturbances. In *Proc. of the 2022 American Control Conference*.

Granichin, O. (2004). Linear regression and filtering under nonstandard assumptions (arbitrary noise). *IEEE Transactions on Automatic Control*, **49** (10), pp. 1830–1837.

Granichin, O. and Amelina, N. (2015). Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances. *IEEE Transactions on Automatic Control*, **60** (6), pp. 1653–1658.

Granichin, O. and Polyak, B. (2003). *Randomized algorithms of estimation and optimization under almost arbitrary noise*. Nauka.

Kosaty, D., Vakhitov, A., Granichin, O., and Yuchi, M. (2019). Stochastic fast gradient for tracking. In *2019 American Control Conference (ACC)*, pp. 1476–1481.

Lewis, F. L., Zhang, H., Hengster-Movric, K., and Das, A. (2013). *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media.

Moral, P. D. and Doucet, A. (2004). Particle motions in absorbing medium with hard and soft obstacles. *Stochastic Analysis and Applications*, **22** (5), pp. 1175–1207.

Nicholas Metropolis and Stanislaw Ulam (1949). The monte carlo method. *Journal of the American Statistical Association*, **44** (247), pp. 335—341.

Polyak, B. (1977). Convergence and rate of convergence of interative stochastic algorithms. ii. the linear case. *Autom. Remote Control*, **38** (4), pp. 537–542.

Proskurnikov, A. and Tempo, R. (2018). A tutorial on modeling and analysis of dynamic social networks. Part II. *Annual Reviews in Control*, **45**. 166–190.

Rastrigin L. A. (1968, pages = 376). *Statistical search methods*. Nauka, Moscow.

Rutherford, S. and Bassler, B. (2012). Bacterial quorum sensing: its role in virulence and possibilities for its control. *Cold Spring Harbor Perspectives in Medicine*, **2**.

Sergeenko, A., Granichin, O., and Proskurnikov, A. V. (2020). Advanced spsa-based algorithm for multi-target tracking in distributed sensor networks. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 2424–2429.

Sergeenko, A., Yakunina, M., and Granichin, O. (2020). Hamiltonian path problem solution using dna computing. *Cybernetics and Physics*, pp. 69–74.

Smyth, H. D. (1945). Atomic energy for military purposes (the smyth report). `https://www.atomicarchive.com/resources/documents/smyth-report/smyth\_v.html`. Online; accessed 03 May 2022.

Watson, J. and Crick, F. (1953). A structure for deoxyribose nucleic acid. *Nature*, **171** (346), pp. 737—-738.

Zhu, J. and Spall, J. C. (2018). Probabilistic bounds in tracking a discrete-time varying process. In *2018 IEEE Conference on Decision and Control (CDC)*, pp. 4849–4854.