Distributed approach for optimization problems

D Zaporozhets and D Zaruba
Computer-aided design department, Southern Federal University, Rostov-on-Don, Russian Federation

E-mail: duzaporozhets@sfedu.ru, dvzaruba@sfedu.ru

Abstract. The paper deals with a distributed approach for optimization problems based on the use of bioinspired algorithms. This class of algorithms allows us to split solutions into independent subsets and process it in separate streams. One of the main problems is a decomposition and subsequent convolution of the solution. The problem becomes more complex in those cases when several decomposition levels are required. From the point of view of computing power, the simultaneous processing of parallel threads requires significant CPU time and RAM resources. In this regard, the use of several compute nodes that interact via network interfaces contributes to an increase in computational resources and an enhance fault tolerance of the system. The paper proposes the distributed subsystem for NP-complete optimization problems solving, which allows to split a set of input data into subsets in an automated mode, distribute subtasks between computational nodes, and collect the results to solve the original problem. To confirm the system performance, a software implementation was developed in the Java and the message broker RabbitMQ to ensure the interaction of software agents with each other. A series of experiments were carried out, in which studies were conducted with several simultaneously running tasks and agents.

1. Introduction

In the world community, bioinspired search optimization methods, based on the simulation of natural phenomena and biological systems, use to increase the speed solution and to partially solve the problem of preliminary convergence of NP-hard problems [1]. One of the main features of this class of algorithms involves the alternative solutions processing and represent an alternative to classical search algorithms that work with only one solution [2]. Thus, optimal and quasi-optimal solutions can be found in polynomial time. Due to an increase in the amount of input data, the number of possible alternative solutions grows factorially. Therefore, the use of various approaches related to the decomposition the task into parts and parallel processing of these parts is an important task [1-3].

This paper proposes a new, distributed approach to the implementation of bioinspired algorithms for optimization problems. The main idea of the proposed approach is the automatic division the problem into parts according to a given criterion and a certain rule with subsequent convolution. The classic traveling salesman problem, well-known benchmarks (Eilon 50, Eilon 75, Oliver 30) [1-3] and generated randomly graphs with given parameters were chosen as the test problem.

All bioinspired algorithms belong to the class of heuristic algorithms. For these algorithms, convergence to the global optimum has not been proved theoretically, but it has been empirically established that the probability of obtaining the optimal or quasi-optimal solution is high [3,4].
2. A general scheme of bioinspired algorithms
In general, all bioinspired algorithms include the following steps.

- Initialization of the initial set of alternative solutions (population). A given number of alternative solutions, that are close to the desired solution in different degrees, is created in the specified search area. Deterministic or random algorithms can be used to obtain the initial population [4-6].

- Generation of new alternative solutions. With the help of a certain set of operators, new solutions are generated in such a way, that each new solution comes close to the optimum iteratively. This set of operators (mathematical transformations based on different probabilistic approaches) is a feature of each bioinspired algorithm.

- Selection of received solutions. Generation of new alternative solutions increases the size of the population several times (depending on the set parameters). In the general case, the population may have solutions with a low estimate of the objective function or even illegitimate solutions.

- Completion of a search. Verification of the stop criterion is carried out. If the stop criterion is reached, the best solution in the population are taken as an approximate solution of the given problem.

One of the main advantages of bioinspired algorithms is its modular structure. This approach allows to quickly get a large number of new variants of the algorithm by developing new and modifying the existing rules for initializing and generating new agents. All bioinspired algorithms are characterized by the following properties of agents [7]: autonomy, stochastic behaviour, communicability, informational limitations, decentralization.

Bioinspired optimization algorithms have a number of advantages over classical optimization methods, especially when solving non-formalized problems and high-dimensional problems. Under such conditions, bioinspired algorithms provide a high probability of finding the optimal or quasioptimal (approximate) solution in polynomial time. It is an approximate solution that is often sufficient. Nevertheless, with large amounts of input data, a significant increase in the operating time of bio heuristic solutions is observed. This is due to the fact that the time complexity of these algorithms can vary from O(n^2) to O(n^3). The authors have suggested a distributed approach to address the high dimensionality problem. The main idea is to use a message broker and distribute “tasks” between connected software agents.

3. Architecture of distributed system for optimization problems
To increase the algorithms’ speed for optimization problems, the authors have suggested a system allows to automatically decompose tasks into subtasks and reconcile it [6-8]. As an example, a solution of the traveling salesman problem is proposed based on the preliminary decomposition of input data. A generalized scheme of the algorithm is presented in the figure 1.
The system is based on a message broker. It is proposed to use the well-establish open source message broker RabbitMQ [6-9].

The term “task” determines the following set of data: decomposition conditions and rules, reconsilation and fitness function calculation rules, rules for decision calculation etc.

The task $t_i$ is represented as a following tuple:

$$t_i = < \text{uid}, R, C, Us, E, d_i, p_i, o >.$$  \hspace{1cm} (1)

If condition $C$ in Eq.(1) is met, then the rule $R$ will be applied to decompose the set of input data $d_i$ into the set $D'_i$ and create the set of child subtasks $T'_i = \{ t'_i \} = R(d_i)$.

The result of the convolution rule is a set of output data (solutions). These solutions can be the result of both subtasks and root tasks

$$o_i = \begin{cases} U(T'_i), & \text{if } C = \text{true} \\ E(d_i), & \text{if } C = \text{false} \end{cases}.$$  \hspace{1cm} (2)

In this paper it is proposed to use the uid of the parent task. If for the current task $t_i$ the reference to the parent task $p_i$ is an empty set, then this task is the root. Its solution is the solution to the original applied task as a whole.

In the general case, the life cycle diagram of the task $t_i$ is presented in Figures 2 and 3.
The original task goes to the message broker in the calculation queue. The computation module of one of agents selects the next task from the queue, registers it in the data warehouse with a unique identifier $u_{id}$ and the status IN_PROGRESS and checks the decomposition rule $C$. If the rule $C$ is executed, the original task is divided into subtasks $t_j$ with $p_j = u_{id}$ and sent to the message queue. Otherwise, $E$ calculations are performed. The results are stored in the repository for the current task with the status DONE. To convolve the obtained results, each agent queries the data store in each time interval $\Delta t$ and selects data according to the sample rule $S$, that can be described as follows:

- To count the number of tasks with the same $p_i$, a query is made to the repository. Also this request counts the number of all registered tasks that are children of one task.
- To count the number of tasks with the status DONE and with the same $p_i$, a query is made to the repository. Also this request counts the number of all registered completed tasks that are children of one task.
- If for each $p_i$ the total number of tasks coincides with the number of completed tasks, then for all tasks with the same $p_i$ can be execute the convolution rule. If there are several such groups, the current agent selects the first from the list.

In order for the convolution rule not to be applied simultaneously for the same subtasks by different agents, the data warehouse denies the access for other agents during the implementation of the sample rule $S$.

Let us consider an example of the sample rule $S$ for a relational data warehouse with the use of MySQL.

There is a table TASK with the following fields: ID is a unique identifier of the task, PARENT_ID is a unique identifier of the parent task, STATUS is the current status of the task.

Consequently, the following queries will be applied:

- SELECT PARENT_ID, COUNT(*) FROM TASK GROUP BY PARENT_ID;
- SELECT PARENT_ID, COUNT(*) FROM TASK WHERE STATUS='DONE' GROUP BY PARENT_ID;
- Search among the results obtained in steps 1 and 2 of completely identical entries for which $\text{PARENT_ID}_1 = \text{PARENT_ID}_2 & \text{COUNT} (*)_1 = \text{COUNT} (*)_2$.

4. Experiments

To confirm the mechanism, proposed in this work, a distributed system to solve optimization problems was implemented. The RabbitMQ [7-9] has been chosen as the message broker. The DBMS MySQL [8, 9] has been chosen as the data storage. Software agents are implemented in Java. To obtain estimates of the system operation time and the solutions quality, the classical optimization problem, namely the search of a Hamiltonian cycle in a graph, has been chosen. Benchmarks Eilon_50, Oliver_30, as well as graphs generated by specified parameters, are considered as test graphs. As an algorithm to solve the problem, a simple genetic algorithm has been chosen. The condition for decomposition is $N_t > 0.2 * N_0$.

The decomposition rule is to split the set of input data in half, namely, the random partition of the graph from task $t$ into two subgraphs and the initialization of tasks $t'$ and $t''$, in which the input data $d$ are the subgraphs obtained by splitting.

The control parameters for the genetic algorithm have the following meanings: The population size is $N_0 * N_0$; A number of iterations is 1000; A probability of a crossover operator is 85%; A probability of a mutation operator is 15%; The number of simultaneously running agents is 3.

In the first series of experiments, the dependencies of the system speed on the number of simultaneously performed tasks were identified. $N_t$ identical tasks are simultaneously added in the system. The next run is performed for $N_t + \text{delta}$, where $\text{delta}$ is the increment of the number of simultaneously running tasks.

The dependence of time on the number of tasks in the system is shown in the figure 4.
The dependence of the speed on the number of simultaneously operating agents were identified in the second series of experiments. Five identical tasks have simultaneously added in the system. The dependence of time on the number of agents in the system is shown in the figure 5.

![Figure 4](image1.png) ![Figure 5](image2.png)

**Figure 4.** Dependence of time on the number of simultaneously running tasks. **Figure 5.** Dependence of time on the number of simultaneously running agents.

5. **Conclusion**

The paper presents the bioinspired distributed system for optimization problems based on the automated decomposition input data according to given rules for subtasks’ solving. Then, obtained solutions merge in the initial task solution automatically. The main feature of the system is that the engineer who solves a specific applied task does not need to implement software modules supporting the algorithm, but to concentrate directly on the applied methods and algorithms. The paper presents the main stages of the system, as well as the schemes of the decomposition and convolution mechanisms. The format of the input data has described in detail. The prototype of the distributed system for optimization problems solving has implemented based on described mechanisms. The dependence of the system speed on the number of simultaneously running agents and simultaneously running tasks was carried out. The results of experiments showed that for the most efficient operation of the system, the number of simultaneous agents should be the same as the expected number of simultaneously solved tasks. The further development of the distributed system for optimization problems consists in the integration of non-relational high-performance DBMS as a data warehouse to increase the system speed as a whole.

**References**

[1] Kureichik V, Kureichik Jr. V and Zaruba D 2017 Hybrid bioinspired search for schematic design, *J. Adv. in Intel. Sys. and Comp.* 451 249-55

[2] Zaporozhets D, Zaruba D and Kulieva N 2018 Parallel approach for bioinspired algorithms *J. of Phys: Conf. Ser.* "Int. Conf. Inf. Tech. in Bus. and Ind. 2018 – Ent. Inf. Sys." 42-65

[3] Kim H 2018 Parallel genetic algorithm with a knowledge base for a redundancy allocation problem considering the sequence of heterogeneous component *J. Exp. Sys. with App.* 113 328-38

[4] Donelli M 2017 *Handbook of Research on Soft Computing and Nature-Inspired Algorithms*

[5] Gekeler S and Steinbuch R 2015 Future tasks in optimization and simulation *Bionic Optimization in Structural Design: Stochastically Based Methods to Improve the*
Performance of Parts and Assemblies eds. Gekeler S and Steinbuch R (Berlin, Heidelberg, Germany/Springer-Verlag) 147-53

[6] Acetozi J 2017 Pro Java clustering and scalability: Building real-time apps with Spring, Cassandra, Redis, WebSocket and RabbitMQ (New York, NY, US/Apress) p 149

[7] Bender A, Ward J R, Worrall S, Moreyra M, Konrad S, Masson F and Nebot E 2016 A Flexible System Architecture for Acquisition and Storage of Naturalistic Driving Data J. IEEE Trans. on Intel. Trans. Sys 17 1748-61

[8] Das N et al. 2019 NoSQL Overview and Performance Testing of HBase Over Multiple Nodes with MySQL J. Emer. Tech. in Data Min. and Inf. Sec. Springer, Singapore 269-79

[9] Bell C 2018 Introducing InnoDB Cluster: Learning the MySQL High Availability Stack (New York, NY, US/Apress) p 488