IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 10, No. 1, March 2021, pp. 157~165
ISSN: 2252-8938, DOI: 10.11591/ijai.v10.i1.pp157-165

Generalized swarm intelligence algorithms with domain-specific heuristics

P. Matrenin[1], V. Myasnichenko[2], N. Sdobnyakov[3], D. Sokolov[4], S. Fidanova[5], L. Kirilov[6], R. Mikhov[7]
[1]Novosibirsk State Technical University, Russia
[2,3,4]Tver State University, Russia
[5,6,7]Bulgarian Academy of Sciences, Institute of Information and Communication Technologies, Bulgaria

Article Info

Article history:

Received Oct 24, 2020
Revised Jan 13, 2021
Accepted Feb 16, 2021

Keywords:

Domain-specific heuristic
Job-shop scheduling
Nanoclusters
Particle swarm optimization
Potential energy surface
Swarm intelligence

ABSTRACT

In recent years, hybrid approaches on population-based algorithms are more often applied in industrial settings. In this paper, we present the approach of a combination of universal, problem-free swarm intelligence (SI) algorithms with simple deterministic domain-specific heuristic algorithms. The approach focuses on improving efficiency by sharing the advantages of domain-specific heuristic and swarm algorithms. A heuristic algorithm helps take into account the specifics of the problem and effectively translate the positions of agents (particle, ant, bee) into the problem's solution. And a swarm algorithm provides an increase in the adaptability and efficiency of the approach due to stochastic and self-organized properties. We demonstrate this approach on two non-trivial optimization tasks: scheduling problem and finding the minimum distance between 3D isomers.

Corresponding Author:

P. Matrenin
Department of Industrial Power Sypply Systems
Novosibirsk State Technical University
Novosibirsk 630073, Russia
Email: matrenin.2012@corp.nstu.ru

1. INTRODUCTION

Swarm intelligence algorithms are applied in almost every area of science and engineering, since in practice, they have shown their high efficiency in solving, first of all, complex real-life engineering optimization problems [1-3]. Complex optimization problems are understood as high-dimensionality problems with a large-scale complex topology of the solution search space, nonlinear criteria and constraints. Also, these problems may have more than one objective function, stochastic and dynamic properties. A scientist with an understanding of various swarm algorithms' mechanisms, their weak and strong sides can create hybrid algorithms, combining, as a rule, two swarm algorithms or swarm and evolutionary algorithms. It allows achieving a synergistic effect, increasing the overall efficiency of the resulting hybrid algorithm [4]. There are several directions for hybridization.

Each algorithm in the ensemble works according to its own rules. However, they use they use the general population of agents or exchange the best-found solutions, for example, a hybrid of the PSO and the genetic algorithm (GA) [5], PSO + differential evolution [6], PSO + grey wolf optimizer [7], PSO + biogeography-based optimization [8]. This approach does not require a change in the underlying algorithms. They work in parallel, exchanging data, or using a shared (public) population. As a result, the complexity of the algorithms does not increase; the basic algorithms can be easily replaced. Simultaneously, in such an approach, the synergistic effect may be small because of the weak integration of algorithms into the system.

Adding a rule or stage of updating the positions of agents from another algorithm to the main algorithm. In [9], a mutation stage was added to the classical PSO algorithm at the end of each iteration, similar to that in GA; similarly, the mutation was added to the grey wolf optimizer in [10]; the study [11] also added a mutation stage to the PSO algorithm, but according to the principles of the differential evolution. Papers [12-13] describe PSO with an additional allowance for particle acceleration, calculated using formulas from the gravitational search algorithm. In this approach, a new algorithm is created, often using the principles of both swarm intelligence and evolutionary computation.

Hybrid algorithms with a higher degree of integration of the rules and principles of operation of two algorithms are also possible: FireFly + PSO algorithm [14], PSO + artificial bee colony algorithm [15], PSO + ant colony optimization (ACO) [16], PSO + wolf pack algorithm [17]. This method allows us to create an algorithm that considers problem's features being solved and compensates for the shortcomings of one algorithm by adding properties of another to it. Nevertheless, this significantly complicates the created algorithm for solving the problem and selecting the heuristic parameters' algorithm. In other words, the complexity of the hybrid algorithm exceeds the sum of the complexities of the parts that leave it.

Using meta-optimization when one algorithm performs the selection of parameters of another algorithm. For example, in [18], local unimodal sampling was used to select the values of the PSO algorithm parameters; in the article [19], the PSO parameters are selected using GA; in [20], the ACO parameters are selected using the bacterial foraging algorithm. This approach requires a very large expenditure of computational resources.

At each iteration of the algorithm, all agents' positions in the population or only the best agents are used as an initial approximation for the local search algorithm. In [21], the unstringing and stringing operator was used for the ACO with the local search; the use of various local search algorithms in PSO is discussed in [22]; FireFly + gradient descent is used in [23]; The PSO + lin-kernighan algorithm was considered in [24].

Swarm and evolutionary metaheuristic algorithms themselves are quite complex in understanding their work and fine-tuning to the features of the tasks being solved. Hybrids from different metaheuristic algorithms turn out to be even more complex and less universal. However, as shown in [17], the metaheuristic algorithm's efficiency can sometimes be increased on the contrary, by simplifying it. Perhaps that is why there are many works in which complex hybrid algorithms are successfully applied to solve a specific problem. Still, none of them has become as widespread as less complicated ACO and PSO among SI algorithms [4] or GA among evolutionary ones.

Our approach to solving optimization problems aims to increase SI algorithms' efficiency and versatility without increasing their complexity. We apply a combination of a universal, problem-free SI algorithm with simple deterministic domain-specific heuristic algorithms or, in other words, greedy heuristics. A hybrid of a stochastic universal SI algorithm and a deterministic simple domain-specific algorithm achieves a greater synergy effect than combining two swarm or evolutionary algorithms.

2. GENERALIZED SCHEME OF SI ALGORITHM AND APPLICATION APPROACH

To implement the proposed approach, it is necessary to make sure that it does not depend on the choice of a SI algorithm and a domain-specific heuristic algorithm. In this case, it will be quite versatile and flexible and will not require any modifications to the algorithms; only their interaction features will change. Therefore, it is necessary to define a generalized SI algorithm model so that various SI algorithms can be easily applied since it is impossible to say in advance which one will be better in a given task. Next, we need to define a general universal scheme of interaction between a SI algorithm and a domain-specific heuristic without strong dependencies.

2.1. Generalized SI algorithm model

An SI algorithm is specified by data structures and operations on them, like many other algorithms. SI uses the swarm, i.e., a population of agents as a basic data structure and rules of moving agents as an algorithm for transforming the data. A distinctive feature of the SI algorithms is using an indirect information exchange between agents. For example, the PSO uses the best-found position in the search space [25]; the ant colony optimization uses a pheromone graph [26]; the monkey search algorithm uses a weighted center of agents' positions [27]. It is possible to formulate basic parts of an SI algorithm:

- A set of agents S.
- An object for the indirect interaction M.
- An algorithm of agents' moving and interaction with optimization problem A.
- Heuristic parameters P.
- An input to obtain data from the optimization problem being solved I.

- An output to send solutions of the optimization problem and the final algorithm result *O*.

Any SI algorithm works according to the following scheme.

- Generation of the initial population. The swarm agents distribute randomly in the solution searching space.
- Calculation of fitness-functions. Each agent receives the values of the optimization problem criterion (criteria). If the agent has a better fitness than all previous finesses of all agents, this solution is saved as the best current solution.
- Agents' movement. The agents move in the search space using indirect interactions.
- If the stop condition is satisfied, the algorithm needs to be finished or otherwise needs to be passed to Step 2. The saved best solution is the algorithm result.

2.2. Interaction between a SI algorithm and a domain-specific algorithm

The following interface and interaction steps were proposed for the interaction of an SI algorithm and a domain-specific heuristic algorithm.

- The SI algorithm receives a dimension of the search space of an optimization task via input *I*. It is used as the length of the vector *X*.
- The SI algorithm generates the variants of the problem solution by agents' movements. A count of variants is a count of agents.
- Each position is sent to a domain-specific heuristic algorithm of an optimized system via output O_{pos}.
- The domain-specific heuristic algorithm encodes the position and uses it for solving an optimization problem and calculating the criterion values. Thus, the SI algorithm works as a meta-optimizer for the domain-specific algorithm.
- All obtained criterion values are sent to the SI via *I*.
- The SI algorithm receives criterion values and performs agents' movement, i.e., we go to the second step of this algorithm.

When a stop-condition is satisfied, then the best agent position is transmitted to output O. The simplest stopping condition is the execution of a given number of iterations.

The interaction's listed steps provide the independence of used algorithms. The approach proposed allows us to implement different SI algorithms quickly. As far as changes in the optimization problem do not cause the necessity of changing anything in the SI algorithm implementation, and vice versa. The absence of close relationships makes it possible to combine any SI algorithm with domain-specific heuristics.

2.3. Mapping of agents' positions

In the proposed approach, we use the search space constraints from 0 to 1 for each dimension. The introduced restrictions are necessary for successful adaptation and simple integration of the algorithm for solving various optimization problems. The fact is that some SI parameters have a different effect depending on the scale of optimization variables. For example, in the firefly optimization algorithm [28], the distance between agents affects agents' attraction nonlinearly. Therefore, if a range of search space is from 0 to 100, the distance from the middle (50) to the high edge (100) will be 75. While if this distance is scaled from 0 to 1, then the same distance will be equal to 0.75, and the degree of attraction between the agents would be completely different. At the same time, this distance is equivalent to the optimization task. Also, the radius of the neighborhood region width of the bee's algorithm [29]. The larger the distance between the boundaries of a search space direction, the greater the neighborhood region width should be. It would be necessary for a non-homogeneous search space to introduce different values of the parameters for different directions.

As a result, the algorithm parameters would have to be changed even if a simple change of measurement units in the task (hours to seconds, and kilometers to feet). We suggest specifying the search space bounded from 0.0 to 1.0 in all directions into the algorithm to avoid this. And mapping of the agents' coordinates from the algorithm into the values of the optimized variables. In the simplest variant, this can be done as follows. We denote the coordinate vector of an agent $X = \{x_1, x_2, \ldots, x_L\}$, and the vector of optimized variables $Q = \{q_1, q_2, ..., q_L\}$. Under the restrictions $a_i \le q_i \le b_i$, we obtain a map of the form
$q_i = x_i(b_i - a_i) + a_i$, $i = 1, \ldots, L$.

A more sophisticated variant is shown in sections 3–4. The proposed parameter mapping is necessary to eliminate unnecessary relationships between the optimization problem model and the optimization algorithms.

3. RESULTS AND DISCUSSION

3.1. Application example. job-shop scheduling problem

This subsection introduces its own symbolic notation, which should not be confused with the notation in the descriptions of the SI algorithms (Section II). As mentioned above, a model of an optimization problem and the SI algorithms are completely independent. The job-shop scheduling problem is among the most challenging combinatorial optimization problems. Scheduling tasks may be characterized as one of the most significant optimization problems since plans and schedules need to be arranged in all fields. The problem consists of scheduling a set of jobs N on a set of machines M to minimize the processing time named as the makespan. It is the time needed to process all jobs. Each job includes a number of operations and has a predefined operation order. The order defines a specific machine and time interval for each operation and sequence of operation. Each machine can process only one operation at a time. It needs to find the shortest (quickest) schedule as a distribution of the operations to time intervals on the machines. This problem belongs to the NP-hard class [30]. Let: $N = \{1, \dots, n\}$ is the set of jobs; $M = \{1, \dots, m\}$ is the set of machines; $O = \{o_0, \dots, o_{L+1}\}$ denote the set of the all operations, 0 and L+1 are fictions operations: start and finish, L is the total number of operations of all jobs [31-32].

Each operation $o \in O$ has its processing time $p(o)$. And A denotes the set of ordered pairs of operations constrained by the precedence relations; E_k denotes the set of all pairs of operations which require machine M_k. Next, let $t(o)$ is the start time of the operation o, $t(o_0) = 0$. The problem is to find a schedule:

$$\max(t(o) + p(o)),\ o \in O; \tag{1}$$

$$t(o) \geq 0,\ \forall o \in O; \tag{2}$$

$$t(v) - t(o) \geq p(o),\ \forall (o, v) \in A; \tag{3}$$

$$t(w) - t(o) \geq p(o) \lor t(o) - t(w) \geq p(w), \tag{4}$$

$$\forall (w, o) \in E_k,\ k = 1, \dots, m.$$

The value of $\max(t(o) + p(o))$ is called the makespan.

3.1.1. SI Algorithm with the greedy heuristic for job-shop scheduling problem

The following algorithm can be used as a simple greedy heuristic for solving the problem.

```
Input: $O, A, L, E_k, m, n$
1. $\Psi \leftarrow \{\}$
2. $\Omega_{1:L-1} \leftarrow$ sort($O$) in descending order and preserving the order given in $E_k$, $k = 1,\dots, m$
3. $\Omega_0 \leftarrow o_0$, $\Omega_L \leftarrow o_L$
4. $t(\Omega_0) \leftarrow 0$, $\Psi \leftarrow \{o_0\}$
5. For $i = 1 \dots L$:
    5.1. place the operation $\Omega_i$:
        $t(\Omega_i) \leftarrow \min(t(\Omega_i))$ under conditions:
        $t(v) - t(\Omega_i) \geq p(\Omega_i)$, $\forall(v) \in \Psi$, $\forall(\Omega_i, v) \in A$;
        $t(w) - t(\Omega_i) \geq p(\Omega_i) \lor t(\Omega_i) - t(w) \geq p(w)$, $\forall(w) \in \Psi$, $\forall(w, \Omega_i) \in E_k$, $k = 1, \dots, m$
    5.2. $\Psi \leftarrow \Psi \cup \{\Omega_i\}$
Output: $\Psi$ (vector of distributed operations)
```

As a result, for each operation $o \in O$, the start time $t(o)$ will be determined, and the value of the optimality criterion will be equal to $t(o_L)$.

This greedy heuristic allows us to schedule activities to prioritizing more extended activities. The longer the execution time of the operation $p(o)$, the more difficult it is to find a free machine for it; therefore, priority is given to more extended operations. The advantage of the algorithm is its simplicity and the ability to add additional restrictions on the relationship of operations to it. However, such an algorithm can give solutions that are far from optimal.

The algorithm's efficiency in terms of the criterion of the problem (minimization of the makespan) can be significantly increased due to the intelligent prioritization of operations. In this case, the same heuristic scheme can be applied only to order the operations not by their duration but with priorities, which can be determined using SI algorithms. For applying an SI algorithm to the job-shop scheduling problem, an agent position (vector X) is needed to map into a possible schedule. The vector X must contain as many elements as total operations in all jobs (L). The mapping process consists of several steps.

```
Input: $O, X$
1. $OX \leftarrow$ concatenate_element_by_element($O, X$)
2. $OX \leftarrow$ sort($OX$) ascending $x$.
3. $O^* \leftarrow OX_{1:L,1}$ (remove $x$ from pairs $\{o, x\}$
4. For $i = 1 \dots L$:
replace the operation $o_i$ by the operation $o_k$ that should be on the correspondence order for this job.
Output: $O^*$
```

Consider an example of 3 works, each of which consists of 3 operations.
$J_1 = \{o_1, o_2, o_3\}$; $J_2 = \{o_4, o_5, o_6\}$; $J_3 = \{o_7, o_8, o_9\}$.
Let $X = \{0.11, 0.72, 0.57, 0.92, 0.43, 0.21, 0.12, 0.40, 0.3\}$.
Then $O = \{J_1, J_2, J_3\} = \{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8, o_9\}$. After sorting:
$OX = \{\{o_4, 0.92\}, \{o_2, 0.72\}, \{o_3, 0.57\}, \{o_5, 0.43\}, \{o_8, 0.4\}, \{o_9, 0.3\}, \{o_6, 0.21\}, \{o_7, 0.12\}, \{o_1, 0.11\}\}$.
The operation order obtained cannot be used as a schedule since it is possible to violate the sequence of operations within one job.

Finally, taking into account the restoration of order within the works, we obtain $O^* = \{o_0, o_4, o_1, o_2, o_5, o_7, o_8, o_9, o_6, o_3, o_{10}\}$, o_0 and o_{10} are fictions start and finish operations.

The vector O^* does not exactly mean the order of the starting of the stages. It means that operations are extracted from O in the order in which they are located in O^*. For the extracted operation, the moment of the fastest possible start is determined. Thus, the vector O^* does not specify the order of processing operations, but the order of determining the start time for them. Since the time costs for performing each operation are assumed to be constant, the vector O^* uniquely determines the final schedule. Therefore, the vector X is uniquely translated into the required discrete schedule. The advantage of the above scheme is obtaining an allowable schedule for any values of the vector X.

3.1.2. Experimental results

The PSO algorithm was used to solve the set of job-shop problem instances [33] denoted as LA01-LA21 of various sizes provided by OR-Library [34] (LA because the author is S. Lawrence). The sizes of instances are from 10×5 to 15×10 ($n \times m$). Table 1 shows the results (average and the best solutions from ten runs of the algorithm). Also, Table 1 list the best well-known results from the literature [34]. The values of the PSO parameters: population size 50, $\alpha_1 = 1.76$, $\alpha_2 = 1.38$, $\omega = 0.73$, $\beta = 0.28$ [18].

The summary deviation between the best and well-known results is 135 hours (0.71%) using the PSO. The results are close to the best (deviation < 1%) for most tasks (81%). Thus, the PSO algorithm allows solving job-shop scheduling problems very close to the best-known results. Moreover, the PSO algorithm is high-speed, since only 100 PSO iterations were used. Increasing the iteration number can give better results for a longer time. The largest time of solving one instance was about 0.5 seconds using 2.4 GHz Intel CPU i7. The minimum time was 0.08 seconds. Such variation is explained by the different dimensions of the instances.

Table 1. Results of the PSO with the greedy heuristic in job-shop scheduling problem instances

Instance	Avg	Best	Best known
LA01	690.8	666	666
LA02	693.35	658	655
LA03	541.49	597	597
LA04	613.79	590	590
LA05	593	593	593
LA06	926.03	926	926
LA07	902.8	890	890
LA08	882.09	863	863
LA09	955.34	951	951
LA10	958	958	958
LA11	1222.05	1222	1222
LA12	1043.59	1039	1039
LA13	1161.59	1150	1150
LA14	1292	1292	1292
LA15	1283.31	1207	1207
LA16	1011.11	956	945
LA17	819.67	784	784
LA18	930.19	859	848
LA19	865	865	842
LA20	907	907	902
LA21	1128	1128	1046

3.2. Application example. exploring potential energy surface of nanocluster isomers

This subsection discusses two related problems. The first problem is the local minima search on the potential energy surface, which are isomers for a cluster of a given composition. The second problem is calculating the «reaction path» as the energetically most favorable path of the phase point movement between two known local minima. The lower the minimum, the more stable corresponding configuration of atomic nuclei. There are specialized software solutions in the field of bioinformatics for exploring ligand binding/unbinding pathway [35]. Isomers are two (or more) molecules (nanoclusters) with the same molecular formula, i.e., atomic composition. Isomers are divided into two main categories. Structural isomers (or constitutional isomer) have the same formula, but the atoms are bonded together in different orders. Stereoisomers have the same connectivity but the different spatial arrangements [36]. Isomers can have very different physical properties, such as boiling point, melting point, and chemical reactivity. In metal systems, the properties of stereoisomers, as a rule, do not differ.

The act of a structural phase transition in a nanocluster is a mutual geometric rearrangement of atoms occurring in time [37]. Accordingly, the cluster's energy and its physicochemical properties substantially depend on the geometric arrangement of the atoms. Even insignificant changes in the spatial structure of a cluster can lead to rather large changes in its total energy. The initial geometry's suitable choice is crucial for obtaining reliable, calculated values of a cluster's physicochemical characteristics in a stationary (equilibrium) state. For a successfully calculated geometry, changes in energy values during the optimization process should be insignificant (less than a given value). Also, for an optimized geometry, the first energy derivatives (gradient) with respect to all geometric distortions should be close to zero. It suggests that we are on a flat energy surface. When considering the broader features of the potential energy surface (PES), including topology and topography, it has become usual to refer to the PES as the «potential energy landscape» [38].

Energy surfaces and landscapes hold the key to understanding a wide range of molecular phenomena. To construct the PES, it is convenient to use the so-called internal coordinates, which include bond lengths or other interatomic distances, bond and dihedral angles of a cluster/molecule. The number of $3n$-6 (where n is the number of atomic nuclei) of independent coordinates should include those structural parameters that change most dramatically during the transformation under investigation. The paper [39] describes a potential energy surface in terms of local minima and the transition states that connect them provides a conceptual and computational framework for understanding and predicting observable properties.

A potential barrier to structural transformation is the energy of the transition state (saddle point) relative to the initial minimum of the potential energy surface. It is the potential barrier that determines the rate of the process of structural transformation. In the Arrhenius equation, these are the activation energies E_A: $k = k_0\exp(-E_\mathrm{A} / k_\mathrm{B}T)$; E_A is the energy activation, k_0 is the pre-exponential factor for the reaction, k_B is the Boltzmann constant, T is the absolute temperature (in Kelvins).

To construct a minimum energy path between two known constitutional isomers, an efficient solution to the assignment problem in a bipartite graph is required. Each atom of isomer A (reagent) is associated with one and only one atom of isomer B (product). The problem can be solved as a global optimization problem. Applying traditional numerical methods is impractical because they need huge amounts of computational resources like time and memory [40].

3.2.1. Greedy heuristic

Thus, the search for isomers corresponding to local minima and the construction of the minimum energy path between them are the basic stages of the PES construction process. It is necessary to find the permutation function P, which would associate with each atom a_i from A ($i = 1, \ldots, n$) one and only one atom $b_{p(i)}$ from B, so that root mead squared distance, $RMSD(f)=(\Sigma_i{}^n{}_{=1}|a_i - b_j^*|^2)^{1/2} \rightarrow min$, $b_i^*=b_{P(i)}$

Additional restrictions are imposed on collisions of the transition paths of atoms. The minimum distance between the centers of the vectors connecting the atoms a_i and $b_{P(i)}$ ($i = 1, \ldots, n$) must be no less than $c_{min} = c_\mathrm{dist}\cdot\min(min\_disance\_A, min\_disance\_B)$, where c_dist is an empirical coefficient (~0.8), *min_disance_X* – the minimum distance among all pairs of atoms of isomer X (A or B). This limitation, high dimensionality and the presence of several types of atoms do not allow the existing methods for solving assignment problems without their significant modification.

We combine a straightforward greedy heuristic algorithm and the universal PSO algorithms. The heuristic used involves sequentially sorting the atoms of isomer B and matching each of them with the nearest atom from A that has not yet been matched. This heuristic can be written:

```
Input: $A$, $B$, $n$, $L$ ($L$ list of numbers from 1 to $n$ without repetitions)
1. $tabu \leftarrow \{\}$.
2. For $i = 1 \ldots n$; $j = 1 \ldots n$:
$D_{ij} \leftarrow |a_i - b_j|^2$ (for atoms of different sorts, the distance is infinite)
```

```
3. For $i = 1 \dots n$:
3.1. $j \leftarrow$ argmin($D_{L_i j}$), $j \notin tabu$
3.2. $P(L_i) \leftarrow j$
3.3. $tabu \leftarrow tabu \cup \{j\}$.
Output: P
```

Step 3.2 means that L_i atom of *B* goes over to the position of the *j*-th atom of *A*.

The resulting table function *P* is the desired permutation. The advantage of such heuristic is a very high speed of operation so the solution is obtained in one pass through the atoms of isomer *B*. If the phase structure of two isomers is close and isomer *B* is not rotated relative to isomer *A*, this solution method allows to find the optimal solution regardless of the number of atoms and number sorts of atoms. But it is not suitable for more complex problems since at the very first iterations of the work, it can match atoms of isomer *B* to such atoms of isomer A, which, if optimally solved, should be matched with other atoms of isomer *B*.

3.2.2. SI algorithm with the greedy heuristic

The resulting algorithm can be written:

```
Input: A, B, n, m, pso_iters
1. $X_m \leftarrow \{x_1, x_2, \dots, x_n\}$, $x_{mk} \leftarrow$ random(0, 1), $m = 1, \dots, |S|$, $k=1, \dots, n$
2. For $i = 1 \dots pso\_iters$:
  2.1. For $m=1, \dots, |S|$.
    2.1.1. $L \leftarrow$ create_order_list$\{x_{m1}, x_{m2}, \dots, x_{mn}\}$.
    2.1.2. P ← greedy_heuristic($A, B, n, L$)
    2.1.3. $fitness \leftarrow$ RMSD($A, P(B)$)
    2.1.4. If $fitness < fitness\_min$
      $fitness\_min \leftarrow fitness$
      $P_{best} \leftarrow P$
  2.2. Particles' movement
Output: $P_{best}$
```

The developed method has been tested on some isomer configurations from 7 to 39 atoms. Table 2 shows the values of the RMSD obtained by the greedy heuristic only and by the PSO with the greedy heuristic. The mathematical modeling confirmed that the combination of greedy heuristics based on the physical properties of the problem and SI algorithms significantly improved the results regarding applying these methods separately.

Table 2. Results of finding the shortest distance between isomers of the bimetallic cluster

Instance	Greedy heuristic, RMSD, Angstrom	PSO with the greedy heuristic, RMSD, Angstrom
Ag7	9,78	9,02
Au19	24,52	20,80
Au20	no valid solution found	26,88
Au12Ag12	no valid solution found	98,41
Cu15Ag15	63,85	61,72
Co16Cu16	no valid solution found	67,61
Au6Ag33	no valid solution found	48,17

4. CONCLUSION

In this article, a hybrid approach is proposed utilizing the strengths of SI and domain-specific heuristic algorithms. The main idea behind developing is: (a) to help the SI algorithm to take into account the specifics of the solved problem thanks to the domain-specific algorithm and (b) to improve the efficiency of the domain-specific algorithm by introducing stochastic and self-organized properties from the SI algorithm. Different SI algorithms can be easily applied due to SI algorithm's low coupling and the domain-specific heuristic algorithm. The approach's application is demonstrated on the job-shop scheduling problem and the problem of construction of the minimum energy path between two isomers. The experiments confirmed that the proposed approach allows the use of SI algorithms as a meta-optimizer that increases domain-specific heuristic algorithms' efficiency.

ACKNOWLEDGEMENTS

This work was partially funded by Russian Federation of Basic Research, project 20-37-70007; by the Ministry of Science and Higher Education of the Russian Federation in the framework of the State

Program in the Field of the Research Activity project 0817-2020-0007; National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", Ministry of Education and Science – Bulgaria and the Bulgarian NSF under the grant DFNI-DN 02/10; and Novosibirsk State Technical University development program, project C20-20.

REFERENCES

[1] X. S. Li, *et al.*, "Analysis and Simplification of Three-Dimensional Space Vector PWM for Three-Phase Four-Leg Inverters," *IEEE Transactions on Industrial Electronics*, vol. 58, pp. 450-464, Feb 2011, doi: 10.1109/TIE.2010.2046610.

[2] Yan-fei Zhu and Xiong-min Tang, "Overview of swarm intelligence," *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 9, pp. 400-409, 2010, doi: 10.1109/ICCASM.2010.5623005.

[3] Xin-She Yang, Z Cui, Renbin Xiao, Amir H Gandomi, "*Swarm Intelligence and Bio-Inspired Computation. Theory and Applications*," 1st ed., Elsevier, Netherlands, pp. 450, 2013.

[4] A. Slowik and H. Kwasnicka, "Nature Inspired Methods and their Industry Applications - Swarm Intelligence Algorithms," *IEEE Trans. on Industrial Informatics*, vol. 14, no. 3, pp. 1004-1015, 2018, doi: 10.1109/TII.2017.2786782.

[5] Xinsheng Lai and Mingyi Zhang, "An efficient ensemble of GA and PSO for real function optimization," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, Beijing, pp. 651-655, 2009, doi: 10.1109/ICCSIT.2009.5234780.

[6] L. Li, B. Xue, B. Niu, L. Tan, and J. Wang, "A novel PSO-DE-based hybrid algorithm for global optimization," in *Advanced Intelligent Computing Theories and Applications: With Aspects of Artificial Intelligence*, vol. 5227 of Lecture Notes in Computer Science, pp. 785-793, Springer, Berlin, Germany, 2008, doi:10.1007/978-3-540-85984-0_20.

[7] N. Singh and S. B. Singh, "Hybrid Algorithm of Particle Swarm Optimization and Grey Wolf Optimizer for Improving Convergence Performance," *Jour. of Appl. Math.*, vol. 2017, 2017, doi:10.1155/2017/2030489.

[8] Abuelrub, M. Khamees, J. Ababneh, and H. Al-Masri, "Hybrid energy system design using greedy particle swarm and biogeography-based optimisation," *IET Renewable Power Generation*, vol. 14, no. 10, pp. 1657-1667, 2020, DOI:10.1049/iet-rpg.2019.0858.

[9] Ahmed, A. Esmin, and S. Matwin, "HPSOM: a hybrid particle swarm optimization algorithm with genetic mutation," *Intern. Jour. of Innovative Computing, Information and Control*, vol. 9, no. 5, pp. 1919-1934, 2013.

[10] M. A. Tawhid and A.F. Ali, "A Hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function," *Memetic Comp.* vol. 9, pp. 347-359, 2017, doi:10.1007/s12293-017-0234-5.

[11] X. Yu, J. Cao, H, Shan, L. Zhu, and J. Guo, "An Adaptive Hybrid Algorithm Based on Particle Swarm Optimization and Differential Evolution for Global Optimization," *The Scientific World Journal*, vol. 2014, 2014, doi:10.1155/2014/215472.

[12] S. Mirjalili and S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," in *Proc. of Intern. Conf. on Computer and Information Application*, Tianjin, pp. 374-377, 2010, doi: 10.1109/ICCIA.2010.6141614.

[13] A. A. Nagra, F. Han, Q. Ling and S. Mehta, "An Improved Hybrid Method Combining Gravitational Search Algorithm with Dynamic Multi Swarm Particle Swarm Optimization," *IEEE Access*, vol. 7, pp. 50388-50399, 2019, doi: 10.1109/ACCESS.2019.2903137.

[14] M. B. Agbaje, A. E. Ezugwu and R. Els, "Automatic Data Clustering Using Hybrid Firefly Particle Swarm Optimization Algorithm," *IEEE Access*, vol. 7, pp. 184963-184984, 2019, doi: 10.1109/ACCESS.2019.2960925.

[15] Y. Gao, "An Improved Hybrid Group Intelligent Algorithm Based on Artificial Bee Colony and Particle Swarm Optimization," in *Proc. of Intern. Conf. on Virtual Reality and Intelligent Systems*, Changsha, pp. 160-163, 2018, doi: 10.1109/ICVRIS.2018.00046.

[16] N. Holden and A. A. Freitas, "A hybrid PSO/ACO algorithm for discovering classification rules in data mining," *Journal of Artificial Evolution and Applications*, vol. 2008, 2008, doi:10.1155/2008/316145.

[17] L. Zhen, Y. Liu, W. Dongsheng and Z. Wei, "Parameter Estimation of Software Reliability Model and Prediction Based on Hybrid Wolf Pack Algorithm and Particle Swarm Optimization," *IEEE Access*, vol. 8, pp. 29354-29369, 2020, doi:10.1109/ACCESS.2020.2972826.

[18] M. Pedersen and A. Chippereld, "Simplifying Particle Swarm Optimization," *Applied Soft Computing*, vol. 10, no. 2, pp. 618-628, 2010, doi:10.1016/j.asoc.2009.08.029.

[19] P. V. Matrenin and V. G. Sekaev, "Particle Swarm optimization with velocity restriction and evolutionary parameters selection for scheduling problem," in *Proc. of Int. Siberian Conf. Control and Communications*. 2015 International Siberian Conference on Control and Communications (SIBCON), 21-23, May 2015.

[20] P. Li and H, Zhu, "Parameter Selection for Ant Colony Algorithm Based on Bacterial Foraging Algorithm," *Mathematical Problems in Engineering*, vol. 2016, pp. 1-12, 2016, DOI: 10.1155/2016/6469721.

[21] M. Mavrovouniotis, F. M. Muller, and S. Yang. "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Trans. on Cybernetics*, vol. 47, no. 7, pp. 1743-1756, 2017, doi: 10.1109/TCYB.2016.2556742.

[22] C-H. Yang, Y-S. Lin, L-Y. Chuang, and H-W. Chang. "A particle swarm optimization-based approach with local search for predicting protein folding," *Journal of Computational Biology*, vol. 24, no. 10, pp. 981-994, 2017, DOI: 10.1089/cmb.2016.0104.

[23] V. Z. Manusov, P. V. Matrenin, and L. S. Atabaeva, “Firefly algorithm to optimal distribution of reactive power compensation units,” *Intern. Jour. of Electrical and Computer Engineering*, vol. 8, no 3, pp. 1758-1765, 2018, doi:10.11591/ijece.v8i3.pp1758-1765.

[24] M. Rosendo and A. Pozo, “A hybrid Particle Swarm Optimization algorithm for combinatorial optimization problems,” in *Proc. of IEEE Congress on Evolutionary Computation,* Barcelona, 2010, doi: 10.1109/CEC.2010.5586178.

[25] J. Kennedy, R. Eberhart, “Particle Swarm Optimization,” in *Proc. IEEE Intern. Conf. on Neural Network*, Piscataway, NJ, pp. 1942-1948, 1995.

[26] M. Dorigo, V. Maniezzo and A. Colorni, “Ant system: optimization by a colony of cooperating agents,” in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29-41, Feb. 1996, doi: 10.1109/3477.484436.

[27] Z. Ruiqing and T. Wansheng, “Monkey algorithm for global numerical optimization,” *Jour. of Uncertain Systems*, vol. 2, no. 3, pp. 165-176, 2008.

[28] X. Yang, “Firefly algorithm, Stochastic Test Function and Design Optimisation,” *Inter. Jour. of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78-84, 2010, DOI: 10.1504/IJBIC.2010.032124.

[29] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim and M. Zaidi, “Bee Algorithm: A Novel Approach to Function Optimisation,” The Manufacturing Engineering Centre, Cardiff University, Cardiff, Tech. Rep. MEC 0501, 2005.

[30] M. Garey, D. Johnson, and R. Sethy, “The complexity of flowshop and job shop scheduling,” *Mathematics of Operations Research*, vol. 1, pp. 117-129, 1976, doi:10.1287/moor.1.2.117.

[31] D. Pezzella and E. Merelli, “A tabu search method guided by shifting bottleneck for the job shop scheduling problem,” *European Journal of Operational Research*, vol. 120, no. 2, pp. 297-310, 2000, doi:10.1016/S0377-2217(99)00158-7.

[32] P. Pardalos, O. V. Shylo, and A. Vazacopoulos, “Solving job shop scheduling problems utilizing the properties of backbone and “big valley,” *Computational Optimization and Applications*, vol. 47, no. 1, pp. 61-76, 2010, DOI: 10.1007/s10589-008-9206-5.

[33] S. Lawrence, “Supplement to “Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques,” GSIA, Carnegie Mellon University, Tech. Rep., Oct. 1984.

[34] J. E. Beasley, “OR-Library: distributing test problems by electronic mail,” *Journal of the Operational Research Society*, vol. 41, pp. 1069-1072, 1990, doi:10.1057/jors.1990.166.

[35] Marcos-Alcalde, E. López-Vinas, and P. Gomez-Puertas, “MEPSAnd: minimum energy path surface analysis over n-dimensional surfaces,” *Bioinformatics*, vol. 36, no. 3, pp. 956-958, 2020, doi: 10.1093/bioinformatics/btz649.

[36] R. H. Petrucci, W.S. Harwood, F. G. Herring, and J. Madura, “General chemistry: principles and modern applications,” 9th ed., Bergen County, NJ: Prentice Hall, 2019.

[37] N. Yu. Sdobnyakov, V.S. Myasnichenko, S. Cheng-Hung, et. al., “Simulation of phase transformations in titanium nanoalloy at different cooling rates,” *Materials Chemistry and Physics*, vol. 238, 2019, doi:10.1016/j.matchemphys.2019.121895.

[38] M. A. Miller, J. P. Doye, D. J. Wales, “Structural relaxation in Morse clusters: Energy landscapes,” *Journal of Chemical Physics*., vol. 110, no 1, pp. 328-334, 1998.

[39] D. J. Wales, “Decoding the energy landscape: extracting structure, dynamics and thermodynamics,” *Philosophical. Trans. of the Royal Society A*, vol. 370, no 1969, pp. 2877-2899, 2012, doi: 10.1098/rsta.2011.0208.

[40] J. P. K. Doye, “Physical perspectives on the global optimization of atomic Clusters,” *Global Optimization. Nonconvex Optimization and Its Applications*, vol. 85, pp. 103-139, 2006.