IJARCCE

ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 9, September 2021

DOI: 10.17148/IJARCCE.2021.10909

NGPSO algorithm is improved based on difficult NP problem

Vu Van Huan

Hanoi University of Natural Resources and Environment, Vietnam

Abstract: Optimal solving of the NP-hard problems strongly motivates both the researchers and the practitioners to try to solve such problems heuristically, by making a trade-off between computational time and solution's quality. Among the classes of heuristic methods for NP-hard problems, the polynomial approximation algorithms aim at solving a given NP-hard problem in polynomial time by computing feasible solutions that are, under some predefined criterion, as near to the optimal ones as possible. P is the class of decision problems that we can solve in polynomial time and NP is the class of problems for which we can check the solution in polynomial time. Visually, problems that are easy to solve are also problems that are easy to test. Therefore, P ∈ NP. In this paper, we propose an approach to the P vs NP question through a class of the most difficult problems among the problems in the NP class. Such problems are called NP-complete problems.

Keywords: NP class, NP-complete class, Karp, NP-hard problems, heuristics.

I. INTRODUCTION

To date, none of the proposed algorithmic solutions can be considered effective and valid for all data. In the world, the resource ("human", "machine", etc.) management problem arises in various contexts, and the literature is of interest, which is an NP-hard problem [3], [4] and proposes various practical applications [1]. In general, the classical assignment problem consists in finding a one-to-one correspondence between a set of tasks and a set of agents, whose objective is to minimize the total cost of the agents' work. Several examples of variance of the assignment problem can be found in the literature, such as: Assignment of jobs to machines, jobs to workers, workers to machines, students to groups, or computations to compute nodes... Like other combinatorial optimization problems, has been studied and solved by exact methods, specific heuristics, meta-heuristics and by the different ways of hybridization of these approaches [1]. The objective is to find the optimal assignment that minimizes the total cost or maximizes the overall benefit [5].

The resolution of difficult combinatorial optimization problems very often relies on so-called "approximate" methods. They do not aim at solving a problem in an optimal way. A combinatorial optimization problem is NP-hard if and only if its decision variant is an NP-complete problem, or if the complexity of an optimization problem is related to that of the decision problem associated with it. In particular, if the decision problem is NP-complete, then the optimization problem is said to be NP-hard [2]. On the other hand, we can say that a problem is NP-complete if and only if it belongs to NP and NP-hard. Also, a decision problem P is NP-complete if and only if it satisfies these two conditions: P ∈ NP and ∃ P′ ∈ NP such as P′ is reduced to P by a polynomial algorithm [10].

Exact (optimal) algorithms, that compute optimal solutions for the problems but run in exponential time; such algorithms are based upon either search tree-based methods (branch-and-bound, branch-and-cut, branch-and-price, etc.), or upon dynamic programming etc. Heuristic methods, that run faster than the exact algorithms and compute suboptimal solutions (that are, sometimes, unfeasible); the most notorious among these methods being:

• The polyhedral methods,
• The metaheuristics (simulated annealing, genetic algorithms, tabou, etc.),
• The polynomial approximation algorithms with (a priori) performance guarantees.

II. COMPLEXITY THEORY

2.1 Complexity Classes

Definition of NP class Problem: - The set of all decision-based problems came into the division of NP Problems who can't be solved or produced an output within polynomial time but verified in the polynomial time. NP class contains P class as a subset. NP problems being hard to solve.

Definition of P class Problem: - The set of decision-based problems come into the division of P Problems who can be solved or produced an output within polynomial time. P problems being easy to solve

Definition of Polynomial time: - If we produce an output according to the given input within a specific amount of time such as within a minute, hours. This is known as Polynomial time.

ISSN (O) 2278-1021, ISSN (P) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 10, Issue 9, September 2021
DOI: 10.17148/IJARCCE.2021.10909

Definition of Non-Polynomial time: - If we produce an output according to the given input but there are no time constraints is known as Non-Polynomial time. But yes output will produce but time is not fixed yet.
Definition of Decision Based Problem: - A problem is called a decision problem if its output is a simple "yes" or "no" (or you may need this of this as true/false, 0/1, accept/reject.) We will phrase many optimization problems as decision problems. For example, Greedy method, D.P., given a graph G= (V, E) if there exists any Hamiltonian cycle.
Definition of NP-hard class: - Here you to satisfy the following points to come into the division of NP-hard
1. If we can solve this problem in polynomial time, then we can solve all NP problems in polynomial time
2. If you convert the issue into one form to another form within the polynomial time

Definition of NP-complete class: - A problem is in NP-complete, if
1. It is in NP
2. It is NP-hard

2.2 Hamiltonian cycle problem:
Consider the Hamiltonian cycle problem. Given an undirected graph G, does G have a cycle that visits each vertex exactly once? There is no known polynomial time algorithm for this dispute.

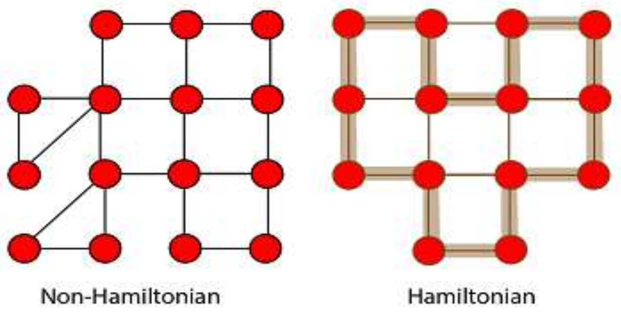

Fig 1: Hamiltonian Cycle

2.3 Relation of P and NP classes
1. P contains in NP
2. P=NP

1. Observe that P contains in NP. In other words, if we can solve a problem in polynomial time, we can indeed verify the solution in polynomial time. More formally, we do not need to see a certificate (there is no need to specify the vertex/intermediate of the specific path) to solve the problem; we can explain it in polynomial time anyway.
2. However, it is not known whether P = NP. It seems you can verify and produce an output of the set of decision-based problems in NP classes in a polynomial time which is impossible because according to the definition of NP classes you can verify the solution within the polynomial time. So this relation can never be held.

Reductions:
The class NP-complete (NPC) problems consist of a set of decision problems (a subset of class NP) that no one knows how to solve efficiently. But if there were a polynomial solution for even a single NP-complete problem, then every problem in NPC will be solvable in polynomial time. For this, we need the concept of reductions.
Suppose there are two problems, A and B. It is impossible to solve problem A in polynomial time.

$$(A \notin P) => (B \notin P)$$

References: An input of a decision problem A is called a Yes-instance if the corresponding output of this input is Yes. Otherwise, we call it a No-instance of problem A. We will use it to denote the length of the input .
Karp Reduction: A decision problem A is said to be reducible to decision problem B in polynomial time if, for each input x of the problem M, there exists an algorithm that, in time $pyly\ |x|$, produces M_x such that x is a Yes-instance of if and only if a Yes-instance of problem B. We will denote:

$$A \preccurlyeq_{Karp} B$$

Remarks: Induction has many different forms, but here we only consider Karp (Karp reduction) induction defined above. Therefore, we will briefly refer to referencing.
The transitiveness of Karp inference:

If

$$A \preccurlyeq_{Karp} B$$

and

$$B \preccurlyeq_{Karp} C$$

Then

$$A \preccurlyeq_{Karp} C$$

III. IMPROVED NGPSO ALGORITHM

3.1 **An Overview of the PSO**

For a long time, population intelligence and evolutionary algorithms have attracted scholars in various research fields. Researchers can derive inspiration from the behavior of biological populations in nature or the laws of biological evolution to design algorithms [4]. As one of many swarm intelligence algorithms, PSO has been proven effective in various optimization fields [7]. PSO is an innovative global optimization algorithm first proposed by Dr. Kennedy and Dr. Eberhart in 1995 [3]. It conducts a collaborative global search by simulating the foraging behavior of biological populations such as birds and fish [5]. The PSO algorithm can enhance the information exchange between in the population individuals by exchanging learning information, and then promote the evolution direction of the population to be consistent. This mode of information exchange resulting from population individuals gives the PSO algorithm a strong search capability and a higher degree of adaptability to optimization problems. In a d-dimensional solution space [6], a particle *i* includes two vectors: one is a speed vector $Vi = (vi1, vi2, vi3, \cdot \cdot \cdot, vid)$, and the other is a position vector

$$Xi = (xi1, xi2, xi3, \cdot \cdot \cdot, xid)$$

Each individual in the population will iterate through two formulas.
The particle speed and position update formula is as follows:

$$v_{id}^{t+1} = wv_{id}^{t} + c_1 r_{1d}(pbest_{id}^{t} - x_{id}^{t}) + c_2 r_{2d}(pbest_{id}^{t} - x_{id}^{t}) \ (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \ (2)$$

Where, w is the inertia weight, which is an important parameter affecting the search performance of the algorithm [1], its value indicates the amount of particles inheriting the current individual speed. c_1 and c_2 are called acceleration factors, c_1 is called cognitive coefficient, which represents the self cognitive experience of particles, c_2 is called social coefficient, which represents the capability of particles to learn from the current global optimal solution; $r1$ and $r2$ are two independent random numbers with sizes between [0, 1] [2]; $pbestid$ is the extreme value of particle *i* in the d-th dimension, $gbestid$ is the global extremum of all particles in the d-th dimension.

3.2 NGPSO Algorithm

The performance of the hybrid meta-heuristic optimization algorithm relies on the complementary advantages of optimization strategies, that is, some strategies have better local search capabilities, and the other strategies have better global search capabilities. As a verification index describing the performance of individual iterations in swarm intelligence algorithms, how to improve both the local optimization and global optimization capabilities of the algorithm has become the key to improving the performance of the algorithm [7]. From the perspective of the iterative process of the algorithm, the global search capability is essentially the full-area search level determined by the iterative individual's breadth-first principle, which shows that the algorithm can search a larger area in the solution space to obtain a better solution [8]. On the other hand, the local search capability is essentially the sub-area search level determined by the iterative individual's depth-first principle, which can use the searched area to improve the problem solution, prevent the algorithm from falling into a stagnation state, and provide the possibility for the algorithm to continue to iterate in the search area and finally obtain a high-precision optimal solution. Therefore, balancing the full-area search level and the sub-area search level becomes the key to optimize the search capability and alleviate the premature convergence of the algorithm. PSO is a very successful algorithm, often applied for solving various practical problems. Nevertheless, similar to other optimization algorithms, PSO also has shortcomings [9], we need to adopt appropriate strategies to improve its shortcomings. The inertial weight w, as a learning level and distribution proportion for balancing particle behavior, indicates the degree of influence of the pre-particle speed on the current particle speed, and determines the search ability of the algorithm. For this reason, we can choose a suitable inertia weight as the improvement strategy of the algorithm, which can enhance the convergence rate and solution accuracy of PSO. At present, the linear decreasing strategy of inertia weight is commonly used, also called linear adjustment *w* strategy [10]. In an actual iteration process, *w* will show a linear decreasing trend as the number of individual iterations increases.When considering the actual optimization problem, the global search is always used first to quickly converge the population iteration space to a certain area, then to obtain high-precision solutions through local search. The *w* update formula is as follows:

$$w = w_{max} - \frac{iter \times (w_{max} - w_{min})}{iter_{max}} \quad (3)$$

where *wmax* and *wmin* are the maximum and minimum inertia weights respectively; *iter* and *itermax* are the current and maximum times of iterations respectively. Generally, the value of *w* is initialized to 0.9 and then linearly decreased to 0.4, which will get better results. However, the search process for the solution of the problem in practical problems is often non-linear. Therefore, the linearly decreasing strategy of inertia weight is not suitable for solving practical problems. Therefore, in this paper, a nonlinear inertia weight adjustment curve is used to adjust *w*, as shown in Equation :

$$w = w_{max} - \frac{iter \times (w_{max} - w_{min})}{iter_{max}} \times \sin\frac{iter.\pi}{2.iter_{max}} \quad (4)$$

Gaussian distribution, also called normal distribution, is usually expressed as *N*(*m*, *s*2), where *m* and *s*2 is mean value and standard deviation respectively, and the Gaussian distribution has 3 – *s* rules. Because the diversity of population increases, the entropy of population becomes higher, which reduces the energy of population and makes the population have higher stability and adaptability. The Gaussian 3 - *s* distribution rule provides a good mathematical theoretical support for optimizing the distribution of individuals in the solution space. If the variable *x* follows Gaussian distribution, the probability density function of the random variable *x* is shown in Equation :

$$f(x) = \frac{1}{\delta.\sqrt{2\pi}}\exp[-\frac{(x-\mu)^2}{2\sigma^2} \quad (5)$$

The Gaussian distribution function of the random variable *x* is as follows:

$$\phi(x) = \frac{1}{\delta.\sqrt{2\pi}}\exp[\int_{-\infty}^{x} -\frac{(t-\mu)^2}{2\sigma^2} dt \quad (6)$$

Gaussian mutation is a mutation strategy composed of random disturbances generated by Gaussian distribution based on the original individual. In this paper, the d-dimensional Gaussian mutation operator is:

$$N_i = (n_{i1}, n_{i2}, n_{i3}, \dots, n_{id} \quad \text{, and } N_i' = (n_{i1}', n_{i2}', n_{i3}', \dots, n_{id}')$$

Follows the standard Gaussian distribution N(m, s2). Gaussian mutation of d-dimensional individual $P_i = (p_{i1}, p_{i2}, p_{i3}, \dots, p_{id})$ can be expressed as follows:

$$P_i' = P_i + P_i \times N_i \quad (7)$$

where: $P_i . N_i = (p_{i1} . n_{i1}, p_{i2} . n_{i2}, p_{i3} . n_{i3}, \dots, p_{id} . np_{id})$

Pi, is the individual after Gaussian mutation. Then we use the greedy selection strategy to prevent the algorithm population from degenerating, the specific operations are as follows:

$$P_i^{t+1} = \begin{cases} P_i' \; if & fit(P_i') > fif\,(P_i) \\ P_i & otherwise \end{cases} \quad (8)$$

3.3. The Main Process of NGPSO

The main process of the proposed hybrid NGPSO is given below:

Step 1: Initialize the population and related parameters, including the population size and algorithm termination conditions;

Step 2: Initialize the speed and position of the particles, and record the pbest and gbest;

ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 9, September 2021

DOI: 10.17148/IJARCCE.2021.10909

Step 3: Update the position and speed of all individuals according to the individual speed update Equation (1) and position update Equation (2) in the PSO;

Step 4: In sequence to prevent the algorithm from premature convergence, the Gaussian mutation is performed on individual individuals to generate new populations;

Step 5: Re-evaluate the adaptability of individuals in the new population. If the fitness of the new individual is better than that of the previous generation, replace the previous generation with the new individual, otherwise the original individual will not be changed;

Step 6: Update pbest and gbest;

Step 7: Judge whether the algorithm reaches the termination condition, if the individual iteration accuracy reaches the termination condition, then end the algorithm, otherwise execute Step 3.

IV. CONCLUSION

This paper proposes the NGPSO algorithm to solve the Gaussian mutation (GM) strategies are introduced to basic PSO, improve the local exploration of the NGPSO algorithm, the GM is used to improve the global exploration and maintain the population diversity of the NGPSO. In other words, GM hav jointly improved the exploitation and exploration of PSO. In order to verify the effectiveness of the introduced strategies, firstly, the proposed NGPSO is used to solve 33 JSSP benchmark instances. It can be concluded that the proposed strategies can improve the performance of the PSO in solving JSSP problems.
NGPSO algorithm can achieve better results than the comparison algorithm. On the whole, in the future, th algorithm needs to be further applied to other scheduling problems, such as flexible job shop scheduling problem. In addition, the algorithm needs more rigorous formal proof, such as the convergence analysis of the algorithm

REFERENCES

[1] Pan, Q.; Tasgetiren, M.F.; Liang, Y. A discrete particle swarm optimization algorithm for single machine total earliness and tardiness problem with a common due date. *IEEE Int. Conf. Evol. Comput.* 2006, 3281–3288.
[2] Abdel-Kader, R.F.An improved PSO algorithm with genetic and neighborhood-based diversity operators for the job shop scheduling problem. *Appl. Artif. Intell.* 2018, *32*, 433–462.
[3] Jiang, T.; Zhang, C.; Zhu, H.; Gu, J.; Deng, G. Energy-efficient scheduling for a job shop using an improved whale optimization algorithm. *Mathematics* 2018, *6*, 220.
[4] Deepa, O.; Senthilkumar, A. Swarm intelligence from natural to artificial systems: Ant colony optimization. *Int. J. Appl. Graph Theory Wirel. Ad Hoc Netw. Sens. Netw.* 2016.
[5] Qin, Q.; Cheng, S.; Zhang, Q.; Li, L.; Shi, Y. Biomimicry of parasitic behavior in a coevolutionary particle swarm optimization algorithm for global optimization. *Appl. Soft Comput.* 2015.
[6] Hema, C.R.; Paulraj, M.P.; Yaacob, S.; Adom, A.H.; Nagarajan, R. Functional link PSO neural ntwork based classification of EEG mental task signals. *Int. Symp. Inf. Technol.* 2008.
[7] Zhu, L.J.; Yao, Y.; Postolache, M. Projection methods with linesearch technique for pseudomonotone equilibrium problems and fixed point problems. *U.P.B. Sci. Bull. Ser. A.* 2020.
[8] Fan, S.S.; Chiu, Y. A decreasing inertia weight particle swarm optimizer. *Eng. Optim.* 2007.
[9] El Sehiemy, R.A.; Selim, F.; Bentouati, B.; Abido, M.A. A Novel multi-objective hybrid particle swarm and salp optimization algorithm for technical-economical-environmental operation in power systems. *Energy* 2019.
[10] Shi, Y.H.; Eberhart, R.C.; Shi, Y.; Eberhart, R.C. Empirical study of particle swarm optimization. *Congr. Evol. Comput.* 1999.