Multi-Objective Vibration-Based Particle-Swarm-Optimized Fuzzy Controller With Application to Boundary-Following of Mobile-Robot Simulation Environment

Liang Ou*
Univ. of Technology Sydney
liang.ou-1@student.uts.edu.au

Guanhua Zeng
Faculty of Engineering and IT,
Univ. of Technology Sydney
guanhua.zeng@student.uts.edu.au

Yu-Cheng Chang*
Student Member
Univ. of Technology Sydney
yu-cheng.chang@uts.edu.au

Chin-Teng Lin*
Fellow
Univ. of Technology Sydney
chin-teng.lin@uts.edu.au

***Abstract*—This paper presents a multi-objective vibration-based particle-swarm-optimization (MO-VBPSO) algorithm with enhanced exploration ability and convergence performance, for training fuzzy-controller (FC) to achieve robot control. The MO-VBPSO applies a reference point-based leader selection schema that assigns leaders for MO-PSOs' searching optimal parameters of the FC. Besides, the MO-VBPSO framework is integrated with a vibration factor to strengthen the exploration ability for resolving the local minima issue, which is inspired by the amplitude of the Firework Algorithm (FWA). The evaluation of MO-VBPSO focuses on the effect of the vibration factor by applying it to training a mobile robot in a simulation environment. The evaluation results are discussed concerning exploration ability, convergence performance, and performance stability. Experimental results reveal that the proposed MO-VBPSO lifts the performance of robot training significantly.**

Keywords—Multi-objective Optimization Particle Swarm Optimization, Firework Algorithm, Fuzzy System, Mobile Robot

I. INTRODUCTION

Fuzzy Systems (FS) are widely applied in mechanical control and proved to be effective. Based on FS, Lin & Lee [1] proposed a Fuzzy Neural Network (FNN) that combines the concept of FS and Neural Network (NN). FNNs leverage membership functions to represent linguistic terms of input variables [7], while traditional NN adopts logistic regression-based functions. One of the popular solutions for FNN optimization is applying Swarm intelligence (SI), which relies on a population of simple agents interacting locally with one another to search the optima in the solution space [2]. Genetic Algorithms (GAs) are a popular type of SI, which is first introduced by Holland [3]. Nowadays, the most popular GAs variations are NSGA-II [4], and NSGA-III [5]. In this paper, the leader selection algorithm is designed inspired by NSGA-III's selection method. However, as stochastic algorithms, GAs is accused to be unstable. This issue is avoided by another optimization method called particle swarm optimization (PSO). PSO is generally used to search optimal solutions in an n -dimensional solution space, which represents n parameters of the fuzzy-controller (FC) in this study.

Although PSO has many advantages, it is designed to solve single-objective problems. For multi-objective problems, PSO must be adjusted. One of the most popular implementations is MOPSO [6], which utilizes Pareto dominance for leader selection. However, the global best particle (gbest) in MOPSO is chosen from a repository with the roulette-wheel mechanism, which brings in instability for the algorithm. In this paper, we will utilize a reference point-based selection method introduce by NSGA-III [5] to alleviate this problem.

The MO-VBPSO model that this paper proposed is developed with 3 goals. First, the primary goal of this paper is to enhance the exploration ability of multi-objective particle swarm optimizations (MO-PSOs). Second, the implementation of MO-VBPSO should boost the convergence speed of the optimization process. Third, MO-VBPSO should have good performance as stable as possible.

The rest of this paper is organized as follows. Section 2 explains the key related works involved in this paper, section 3 elaborate methodology applied by this research, section 4 evaluates and discusses the research results and section 5 concludes the main achievements of this paper.

II. RELATED WORKS

In this section, two major components that the MO-VBPSO works with will be discussed, which are the fuzzy controller (FC) and Pareto-based multi-objective optimization methods. Fig. 1 illustrates their relationships.

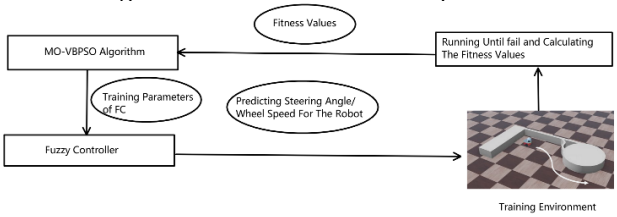


Fig. 1. Components of the robot training model, and their relationships.

* Liang Ou, Yu-Cheng Chang and Chin-Teng Lin are with CIBCI lab, Centre for Artificial Intelligence, FEIT, University of Technology, Sydney, Australia (E-mail: liang.ou-1@student.uts.edu.au, Yu-Cheng.Chang@student.uts.edu.au and Chin- Teng.Lin@uts.edu.au)

Multi-objective optimization methods and PSO are the two key elements that make up the MO-VBPSO algorithm, while the FC is tuned by the MO-VBPSO to guide the robot by outputting steering angles.

A. *Fuzzy Controller*

The design of the FNN-based fuzzy controller refers to Juang & Chang [7], in which the number of zero-order Takagi–Sugeno-type (TSK) fuzzy rules [8] are defined as 10. This setting is for balancing the interpretability and the complexity of the FC [7]. The following formula explains these rules:

$$Ri: If\ x_1\ (k)\ is\ \mu_{i1}\ And\ \dots And\ x_n(k)\ is\ \mu_{in}$$
$$Then\ u(k)\ is\ a_i, \qquad (1.)$$

where $x_1(k)$,…. $x_n(k)$ are inputs of the rule, $u(k)$ is the output, k refers to the timestamp, and a_i is the weight of the rules. μ_{i1} , … , μ_{in} are membership functions that transform inputs into normalized values [7].

B. *Multi-Objective Optimization*

With the PSO definition, identifying gbest and pbest are tricky for multi-objective (MO) problems. Originally, PSO is introduced to solve only single-objective problems, in which gbest and pbest can be found by evaluating all particles on a single fitness function. Pareto optimization methods are one of the most popular types of solutions for MO problems. The proposed MO-VBPSO's optimization method is based on [10].

III. METHODOLOGY

In this section, the design of MO-VBPSO and its framework for robot boundary-following behaviour training will be discussed. The flow chart of the robot training pipeline is shown in Fig. 2. In the beginning, this algorithm will first initialize several hyper-parameters and the swarm of PSO. At this step, c_1 and c_2 that in charge of the learning rate of PSO velocity updating is set to 1.0 for simplicity, the number of divisions $p = 4$. This number p is used for the selection of reference points that introduce by NSGA-III [5]. Details of this selection mechanism will be discussed in section 3-B.

For the swarm initialization, 50 particles will be randomly generated. Each of these particles is a 90-dimensional vector, which defines the 10 fuzzy rules of the FS. After the initialization step, the algorithm executes the MO-VBPSO for a fixed number of iterations. In this paper, the max number of iterations is set to be 100. After 100 iterations of optimization, the training is terminated with final results documented.

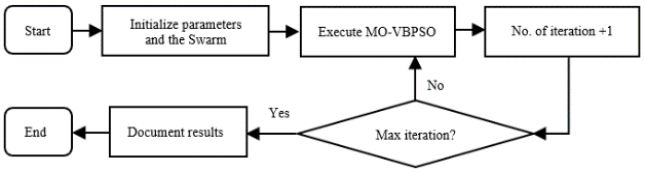
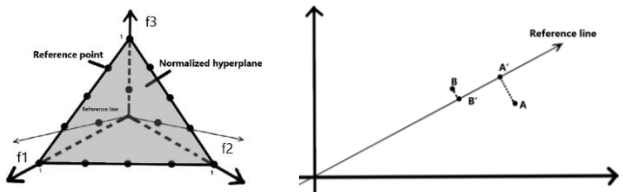


Fig. 2. Flow Chart of the robot training pipeline.



(a) (b)

Fig. 3. Reference points' design, (a) shows 15 reference points and their corresponding reference lines, (b) is a 2-D illustration for the selection of leader candidates.

A. *MO-VBPSO*

MO-VBPSO in the training pipeline is responsible to resolve a multi-objective optimization problem. Since in multi-objective optimization problems, an n-dimensional solution is mapped onto a point in an m-dimensional objective space, MO-VBPSO utilized both the solution space and the objective space. Specifically, leader candidates' selection method (step 3) concerns distances of particles in the objective space, while leader assignment method (step 4) concerns distances of particles in the solution space.

There are 7 major steps of MO-VBPSO.

1) *Update External Archive*

At the beginning of each iteration, the external archive that stores all non-dominant solutions will be updated. To perform this, MO-VBPSO will combine the old external archive and the current swarm. Then, it will identify a new Pareto Front from this union by non-dominated sorting [4].

2) *Define Reference Points*

The second step of MO-VBPSO is to identify reference points for further leader allocation. The selection strategy here is similar to that of NSGA-III [5]. First, a 3-dimensional coordinate system will be established, each axis of this coordinate system represents an objective. The position of a particle in this system is defined by the normalized values of

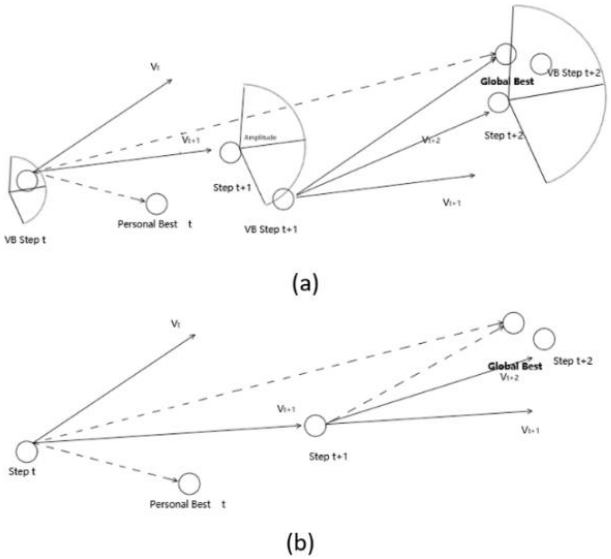


Fig. 4. Comparison between MO-VBPSO (a) and MO-PSOs (b)

its fitness functions. This normalization transformation will be conducted for every iteration, ensuring that all fitness values range from 0 to 1. After the establishment of the coordinate system and the normalization of fitness values of all particles, a triangle plane was created with the apex at (1, 0, 0), (0, 1, 0), and (0, 0, 1). Since the division p is set to be 4, each side of the triangle is divided into 4 sections, and 15 reference points on assigned on the plane, as it is shown in Fig. 3 (a). Since these reference points are evenly distributed across the plane, the objective space can be evenly divided by reference points. Specifics of reference point selection strategy and normalization of fitness values are given by Algorithm 1.

Algorithm 1 Reference Point Selection and fitness value normalization.

Input: nr(number of reference areas), O(non-dominated particles of the front set)
Output: LN(Reference lines),F(the normalized fitness);

initialize: Set LN = array, F = array,
for each $x1 \in [1, nr]$ **do**:
  for each x2$\in [1, nr]$ **do**:
    for each x3$\in [1, nr]$ **do**:
      if x1 + x2 + x3 == nr **then**
        LN.append([x1, x2, x3])
      end if
    end for
    for each $j \in [1, n]$ **do**
      for each $k \in [1, l]$ **do**
        F_{j_k} = the $fitness_j$ of O_k
      end for
    end for

    for each $j \in [1, n]$ **do**
      F_j = minMaxNormalize(F_j)
    end for
return LN, F

3) Select Leader Candidates From the External Archive

After reference points are defined in the objective space, leader candidates can be chosen evenly from the external archive. The chosen strategy is illustrated in Fig. 3 (b), which is a 2-D example of the leader selection method. In order to find the sector of the objective space that a point belongs to, several straight lines, defined as "reference lines" are drawn from the origin point to all the reference points. This can be witnessed in Fig. 3. By calculating all the perpendicular distances from a point in the external archive to those reference lines (Fig. 3 (b)), the nearest line to that point is identified. This line represents the sector that the point belongs to. However, it is inevitable that there will be more than one point falling into the same sector. In this situation, the algorithm will choose the one with the longest distance between its projection on the corresponding reference line and the origin point. For example, in Fig. 3(b), If A and B are the two candidate leaders around reference line R, A will be chosen because its projection A' has longer distance from the origin than B'. Note that there is no guarantee that all sectors in the objective space will be associated with leaders from the archive, since the distribution of points in the archive may not be even.

4) Assign Leaders for All Particles in the Swarm

After leader candidates have been selected from the external archive, each particle will be assigned a leader from the candidates. This leader assignment strategy is similar to the grouping method introduce by Lin, Chen & Lin [9], although the method is extended to 3-dimensional fitness space. The most similar candidate (has least L2 distance in the solution space) to the particle is chosen to be the leader of it.

5) Update Personal Best for All Particles in the Swarm

Although the previous step identified leaders for every particle, the personal best (pbest) of a particle is also required for PSO position updating. For each iteration, the algorithm will compare its current fitness values and the fitness values of its pbest through Pareto dominance. If its pbest is dominated by itself, the pbest will be overwritten, otherwise pbest will remain unchanged.

6) Update Vibration Amplitude for all Particles in the Swarm

To solve the inherent local minima problem of MO-PSOs, this paper introduces a vibration factor for enhancing the convergence performance and exploration ability of MO-PSOs, this paper is inspired by the amplitude algorithm of the Firework Algorithm [11]. Taking advantage of the stability characteristics of the MO-PSO algorithm and the advantage of exploration ability of the Firework Algorithm, this article recommends an innovative algorithm which is designed as the schematic diagram (Fig. 4 (a)), which is the key step of MO-VBPSO. This schematic diagram shows the MO-VBPSO is developed based on MO-PSOs' searching strategy (Fig. 4 (b)) with adding a vibration factor to improve the exploration and convergence performance.

In traditional MO-PSOs, the position updating is narrow and predictable, because particles are only attracted by gbest and pbest [6]. This mechanism leads the hunting zone of particles limited because it can only search relatively rectilinear space.

The features of the vibration factor are:

- The amplitude of the vibration is in a restricted range.
- The vibration has the same direction as the velocity to ensure the stability and efficiency of PSO searching.
- The amplitude is negatively correlated with the distance between the particle and the leader as Fig. 6.
- The vibration factor affects only the update of the positions of particles, but not the velocity. This is because velocity changing has an incremental impact on the instability of the algorithm.

The vibration algorithm implemented in our design is inspired by the Firework Algorithm proposed by [11]. With the vibration algorithm, an Amplitude Vector for particles is given by:

$$A^j = B^j \frac{1 - d(s) + \xi}{\sum_i^n d_i + \xi}, \qquad (2.)$$

where d is the distance between the current particle s and its gbest in the objective space, A^j is the j^{th} component of the Amplitude Vector for the current particle s, which is bounded

by the minimum distances of the j^{th} component of particle s to its boundary B^j, giving by

$$B^j = min\left(\left(s^j_{max} - s^j\right), \left(s^j - s^j_{min}\right)\right), \quad (3.)$$

where s^j_{max} and s^j_{min} are the upper bound and lower bound of the j^{th} component. Formula (3) ensures that the vibrated component will not exceed the limitations of the corresponding parameter. In addition, ξ is a small constant value to avoid zero-division-error.

Formula (2) offer the particle higher possibility to escape from the local minima areas by vibrating it outside. The Amplitude Vector A^j is negatively correlated with the distance between the particle and its leader. This is because the closer the distance the higher change it trapped into a local minimum. This amplitude design provides the particle with an increasing incentive to vibrate out of the potential local minima area as it moves towards the gbest.

7) Velocity and Position Updating

Tradition PSOs update a particle with a predictable path, which potentially leads the particle into a local minimum. In MO-VBPSO, vibration is chosen on random dimensions to enhance particles' exploration ability. The vibrated dimensions are chosen with:

$$z = round\left(D \cdot rand(0,1)\right), \quad (4.)$$

where D is the total number of dimensions of particles.

On one hand, for those vibrated dimensions, the position updating formula is designed as

$$s^j_i(t+1) = s^j_i(t) + v^j_i(t+1) + sign\left(v^j_i(t+1)\right)\gamma^j_3 \cdot A^j, \quad (5.)$$

where $v^j_i(t+1)$ is the updated velocity and $sign\left(v^j_i(t+1)\right)$ is the direction of velocity. Moreover, the amplitude A^j is resized with a random factor γ^j_3 ranged (0,1) for diversity enhancement.

On the other hand, for those non-vibrated dimensions, positions are updated by standard position updating equation as below,

$$\vec{s}(t+1) = \vec{s}(t) + \vec{v}(t+1). \quad (6.)$$

For all dimensions, the velocity is updated by

$$\vec{v}(t+1) = w \cdot \vec{v}(t) + c_1\overrightarrow{\gamma_1} \cdot \left(\vec{p}(t) - \vec{s}(t)\right) + c_2\overrightarrow{\gamma_2} \cdot \left(\vec{g}(t) - \vec{s}(t)\right). \quad (7.)$$

where w is a velocity constrain factor, set to be 0.8 as the convention [8], c_1 is the personal learning coefficient, c_2 is the global learning coefficient, $\overrightarrow{\gamma_1}$ and $\overrightarrow{\gamma_1}$ are two n-dimensional vectors with each dimension ranged from 0 to 1, $\vec{p}$ is the pbest position and $\vec{g}$ is the gbest position.

B. Features of MO-VBPSO

To sum up, the two major features of MO-VBPSO are the vibration factor and the reference point-based leader selection method. In consideration of the first goal of this paper, both features are designed to lift the exploration ability. Specifically, the reference point-based leader selection method guarantees that particles in the swarm are guided by leaders from diverse areas in the objective space. The implementation of the vibration factor widens the searching area of the PSO, as it is shown in Fig. 4. In consideration of the second goal of this paper, convergence speed, both features have impacts on it. The diverse distribution of leaders helps particles escape from potential local minima so that the algorithm has less opportunity to be stuck in local minima. Besides, the vibration factor adds a moving speed to the velocity of particles, thus boosts the convergence speed. Considering the third goal, performance stability, it may potentially be sacrificed. This is because the dimension selection of vibration, formula (4), increased the level of randomness. This will be discussed in section 4.

C. Robot Controller Training and Fitness Functions Design

1) Robot and Training Environmental Settings

Experiments of this research are conducted on the robot emulation software "Webots-R2019b" (https://cyberbotics.com/). Pioneer 3DX-p1 is chosen to be the robot model, which sat up with two wheels. A planform vision of the robot model is shown in Fig. 5 (a). The coverage, S_1 to S_8, of lidar sensor refers to the setting of [7]. The training purpose is to achieve boundary-following behaviour by the robot. The robot is expected to successfully execute the boundary-following behaviour near the wall as smooth as it can. Fig. 7 illustrates the training environment, in which the robot will be initiated in the centre at the beginning of each trial, towards the right. A constant speed (0.524 m/s) will be assigned to the robot, and the steering angle of the robot will be adjusted by the fuzzy controller at each timestamp. A trial ends when it succeeds (robot has run more than 4000 timestamps) or fails (robot has moved too far or too close to the obstacle). Fail of a trail is defined by breach of a distance limitation defined by:

$$min(S_1, S_2, S_3, S_4) \geq D_{min}, \quad and\ S_1 \leq D_{max}, \quad (8.)$$

where S_1, S_2, S_3 and S_4 are the reads of the 4 sub-coverages of the lidar sensor, D_{min} is set to 0.5m and D_{max} is set to be 5m. After a trial is stopped, 3 fitness values will be

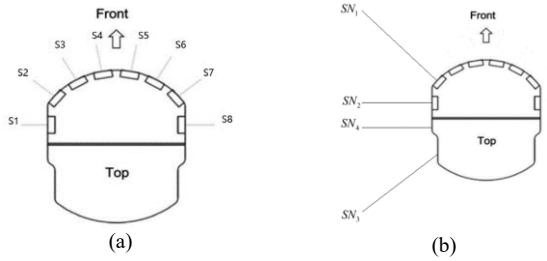


Fig. 5 (a) Lidar sensor of the Robot and its coverage, divided into S_1 to S_8 [7]. (b) Auxiliary sensors added in the robot training process.

documented for the training of the VBPSO model, then a new trial will begin with the updated VBPSO model.

2) Fitness Functions Design

To evaluate the performance of the wall-following behaviour, and guide the algorithm searching in the solution space, which is 90-dimensional vector space that represents tunable 90 parameters in the FNN, 3 fitness functions are designed by:

$$f_1 = T_c,$$
$$f_2 = \sum_{t=1}^{T_c} \frac{T_c}{\left|\frac{1}{2}\left(\frac{SN_1(t)}{SN_3(t)} + \frac{SN_2(t)}{SN_4(t)}\right) - 1\right|},$$
$$f_3 = \sum_{t=1}^{T_c} \frac{T_c}{|SN_2(t) - 0.2|}, \quad (9.)$$

where T_c is the timestamp. Fitness1 is a simple measure of the number of timestamps the robot run until distance limitation is breached. Based on the set up that discussed previously, values of f_1 range from 1 to 4000. For Fitness 2, SN_1, SN_2, SN_3 and SN_4 are read of auxiliary sensors that are shown in Fig. 5 (b). This fitness design refers to Juang, Jeng & Chang [12] for evaluating the smoothness. Fitness 3 is a measurement of the average distance between the wall and the robot. In this paper, we set 0.2m to be the ideal distance.

IV. Results and Discussion

Since the key implementation of this research is introducing the "Vibration Factor" into PSO updating, the comparison is between the result of MO-VBPSO with the "Vibration Factor" and the result of MO-VBPSO without the "Vibration Factor". For both tests, we conducted 10 times of experiments with 100 learning iteration. At the end of each experiment, the external archive, which includes all Pareto optimal solutions, is documented.

TABLE I. Evaluation results of the improvement percentage by the implementation of vibration factor

Evaluation Results	
Evaluation Criterion	***Performance Improvement***
C-Metric Dominance	0.4
S-Metric Dominance	0.44
Average Dominance	0.42

TABLE II. Statistics comparison for S-Metric Results

S-Metric	***With vibration***	***Without Vibration***
Min	0.933	0.830
Max	1.522	1.414
Mean	1.195	1.050
Std	0.204	0.184

TABLE III. Statistics comparison for Number of leaders

Number of Leaders	***With vibration***	***Without Vibration***
Min	8	7
Max	10	10
Mean	9.1	8.5
Std	0.831	1.024

In this paper, the evaluation criteria are "C-Metric" dominance [13] and "S-Metric" dominance [13]. The comparison is between the algorithm with "Vibration Factor" and the algorithm without "Vibration Factor".

Table I shows the overall evaluation results of the two criteria. It shows that performance is lifted tremendously under both criteria, and 42% improvement is achieved on average. With C-metric dominance at 0.4, the implementation of the vibration factor lifts 40% algorithm performance concerning C-Metric. Another front comparison criterion is "S-Metric", also known as hypervolume metrics. The approach for comparing S-Metric values for the 20 experimental results is similar to that of C-Metric. From Table I, we can see that the algorithm with the vibration factor is 44% better than the algorithm without the vibration factor. In a visualized way, Fig. 6 compares the S-Metric evolving curve. While the curve of the algorithm without Vibration stuck in a local minimum from iteration 60 to iteration 80 and eventually ends with 0.917, the algorithm with vibration evolve persistently and ended with 1.422. This graph proves that the introduction of Vibration lifts the convergence speed. Statistics of S-Metric evaluation are shown in Table II. The Implementation of Vibration Factor lifts about 10% performance concerned with S-Metric.

A. Trajectory demonstration

A more practical and intuitive demonstration of the impact of the vibration factor is shown in Fig. 7, which compares trajectories of the vibrational algorithm with the non-vibrational algorithm. In the figure, the robot moves smoother under the vibration-based algorithm than the non-vibration

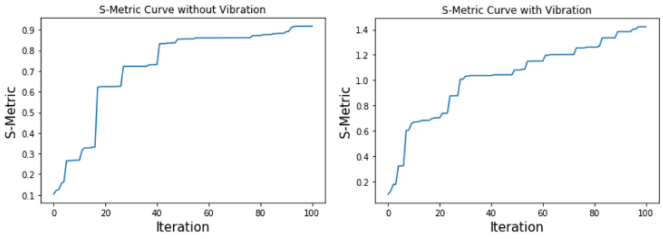


Fig. 6. Examples of S-Metric evolving cure, (a) for algorithm without vibration and (b) for the algorithm with vibration

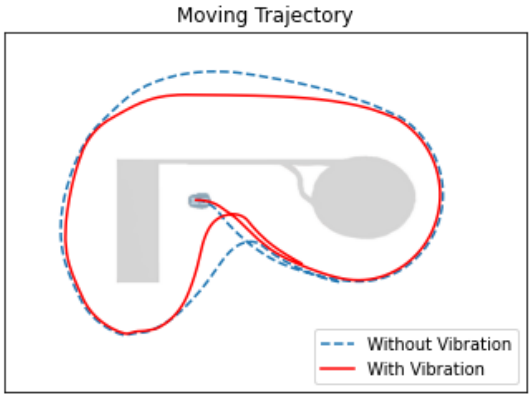


Fig. 7. Moving trajectory comparison

algorithm. This is typically witnessed when the robot observes a straight wall and a U-shaped wall. In other words, this phenomenon can be explained as the enhancement of fitness 2 and 3.

B. Discussion

Since the goals of this research are improving exploration ability, convergence and stability of MO-PSOs through the implementation of vibration factor, results will be discussed concerning these 3 perspectives.

1) Exploration Ability

According to C-Metric evaluation results discussed in section 4-B, fronts found with vibration are 40% better than fronts found without vibration. From S-Metric evaluation, Table III, we can see that Min, Max, Mean of S-Metric of the front with vibration are all higher than fronts without vibration. In this sense, fronts with vibration cover larger space than front without vibration. Diversity is another major criterion of algorithm exploration ability, discussed in section 1. According to Table III, the number of leaders identified by the algorithm with vibration is, on average, larger than the algorithm without vibration. With more leaders guiding the swarm, diversity of exploration is enhanced.

2) Convergence Speed

Convergence can be witnessed in Fig. 6. It is obvious that the ascending speed of the S-Matric of the algorithm with vibration is much faster than the algorithm without vibration. This is because the implementation of the vibration factor enhances the algorithm's capacity for escaping from local minima so that plateaus on the curve are largely shortened

3) Performance Stability

From Table II, we can see that the standard deviation of the results for S-Matric with vibration is about 10% higher than the results without vibration. This is an indicator of the instability of MO-VBPSO. However, for the number of leaders, the standard deviation of results with vibration is about 15% lower than results without vibration (Table III). This indicates the stability of MO-VBPSO in finding leaders.

V. Conclusion

This paper proposes a MO-VBPSO to design an FC for a mobile robot to perform boundary-following behaviour. The MO-VBPSO is based on features from MOPSO [6], NSGA-III [5] and FWA [11]. Specifically, while position updating, velocity updating and fitness comparison schemas are the same with MOPSO, the leader selection strategy is designed based on NSGA-III's sorting approach. The most influential part of MO-VBPSO is its vibration factor implementation, which is inspired by FWA. To evaluate the effect of the implementation of the vibration factor, experiments are conducted on a robot simulation environment. Results show that exploration ability and convergence speed have been improved to a large extent. For stability, the vibration factor does bring in some variations for S-Metric values, while it still stabilizes the diversity of distribution for leaders. In the future, the test of MO-VBPSO should be compared with other popular algorithms and evaluated under more types of problems. The robot training environment can also be complex, with its training results tested in a real-world environment. In addition, the MO-VBPSO can be tested on a type-2 fuzzy neural controller [14] in the future.

References

[1] C. -T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," in *IEEE Transactions on Computers*, vol. 40, no. 12, pp. 1320-1336, Dec. 1991.

[2] C. Chang and X. Wu. *A Survey of Group Intelligence Optimization Algorithms, Advances in Intelligent Systems and Computing*, vol. 928, pp. 1024-1033, 2020.

[3] J. Holland, "adaptation in natural and artificial systems, university of michigan press, ann arbor,"" *Cité page,* vol. 100, 1975.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* Article vol. 6, no. 2, pp. 182-197, 2002.

[5] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation,* Article vol. 18, no. 4, pp. 577-601, 2014, Art. no. 6600851.

[6] C. A. Coello Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, 2002, vol. 2, pp. 1051-1056.

[7] C. Juang and Y. Chang, "Evolutionary-Group-Based Particle-Swarm-Optimized Fuzzy Controller With Application to Mobile-Robot Navigation in Unknown Environments," in *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 2, pp. 379-392, April 2011.

[8] J. Y. Jhang, C. J. Lin, C. T. Lin, and K. Y. Young, "Navigation Control of Mobile Robots Using an Interval Type-2 Fuzzy Controller Based on Dynamic-group Particle Swarm Optimization," *International Journal of Control, Automation and Systems,* Article vol. 16, no. 5, pp. 2446-2457, 2018.

[9] T.-C. Lin, C.-C. Chen, and C.-J. Lin, "Navigation control of mobile robot using interval type-2 neural fuzzy controller optimized by dynamic group differential evolution," *SAGE Journals*, vol. 10, no. 1, p. 1687814017752483, 2018.

[10] K. Parsopoulos and M. Vrahatis, "Multi-objective particle swarm optimization approaches," 2008, pp. 20-42.

[11] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *International conference in swarm intelligence*, 2010: Springer, pp. 355-364

[12] C. F. Juang, T. L. Jeng, and Y. C. Chang, "An interpretable fuzzy system learned through online rule generation and multiobjective ACO with a mobile robot control application," *IEEE Transactions on Cybernetics,* Article vol. 46, no. 12, pp. 2706-2718, 2016.

[13] E. Zitzler, *Evolutionary algorithms for multiobjective optimization: Methods and applications.* Citeseer, 1999.

[14] M. Sugeno and T. Takagi, "Fuzzy identification of systems and its applications to modelling and control," *Readings in Fuzzy Sets for Intelligent Systems,* vol. 15, no. 1, pp. 387-403, 1993.