

**DEVELOPMENT OF IMPROVED ABC
AND PSO ALGORITHMS FOR SPECIFIC
NIA**

A Thesis

Submitted for the award of the Ph.D. degree

In Computer Science

(Faculty of Science)

to the

University of Kota

By

Kavita Sharma



Under the Supervision of

Dr. P. C. Gupta

Associate Professor

Department of Computer Science and Informatics

UNIVERSITY OF KOTA, KOTA (RAJ.)

2021

*With utmost gratitude and devotion,
I dedicate this Thesis
to
my familymembers*

Supervisor Certificate

I feel great pleasure in certifying that the thesis entitled “**DEVELOPMENT OF IMPROVED ABC AND PSO ALGORITHMS FOR SPECIFIC NIA**” has been carried out by Kavita Sharma under my guidance. She has completed the following requirements as per Ph.D. regulations of the University.

- a Course work as per the University rules
- b Residential requirements of the University (200 days)
- c Regularly submitted the annual progress report
- d Presented her work in the departmental committee
- e Published/Accepted two research papers in a refereed research journal and two papers in conference proceedings

I recommend the submission of the thesis.

Place: Kota

Date:

Dr. P. C. Gupta
Supervisor & Head
Department of CSI

Anti-Plagiarism Certificate

It is certified that Ph.D. Thesis Titled “**DEVELOPMENT OF IMPROVED ABC AND PSO ALGORITHMS FOR SPECIFIC NIA**” by Kavita Sharma has been examined with the anti-plagiarism tool. We undertake the follows:

- a** The thesis has significant new work/knowledge as compared already published or are under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.
- b** The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented as the author’s own work.
- c** There is no fabrication of data or results which have been compiled and analyzed.
- d** There is no falsification by manipulating research materials, equipment or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.
- e** The thesis has been checked using ANTI-PLAGIARISM URKUND software and found within limits as per HEIC plagiarism policy and instructions issued from time to time.

Kavita Sharma
Research Scholar
Department of CSI
Place: Kota
Date:

Dr. P. C. Gupta
Supervisor & Head
Department of CSI
Place: Kota
Date:

Abstract

Swarm intelligence (SI) inspired strategies are based upon the collective intellectual conduct of several species available in nature. These SI-based schemes are executing pretty well to determine the solution for real-world complex optimization problems. For example, the artificial bee colony (ABC) algorithm is one of the efficient members of SI-based theorems. The ABC is based on the mathematical simulation of the food foraging conduct of honey bees. However, like other SI-based strategies, ABC and PSO also suffer from premature convergence, stagnation, and, sometimes unable to unfold the actual outcomes for the optimization problems.

To overcome the existing drawbacks and enhance the execution ability of the ABC and PSO algorithm, in this thesis, the primary ABC and PSO algorithm is redesigned using several approaches. In this research, two new variants of ABC are designed, Limaçon-inspired ABC and Fully Informed ABC, and one new variant of PSO is designed, namely, Fitness-based PSO. Further, these variants are applied to solve a real-world complex optimization problems, namely, job shop scheduling problem (JSSP).

Firstly, an influential local search (LS) technique that is designed by taking inspiration by limaçon a curve is incorporated in ABCA, and the designed strategy is named Limaçon inspired ABC (LABC) algorithm. Then, the exploitation capability of the LABC strategy is tested over 18 complex benchmark optimization problems. Finally, the test results are compared with similar state-of-art algorithms, and statistical analysis shows the LABC can be considered a practical variant of the ABC algorithms to solve the complex optimization problems.

Further, a novel NIA, namely Fully Informed Artificial Bee Colony (FABC) algorithm is developed by taking inspiration from the position update strategy of the fully informed particle swarm optimization algorithm. In the FABC, the onlooker bee process of the Artificial Bee Colony (ABC) algorithm is modified and designed such that the new position of the solution search agent is obtained while learning from all the nearby agents. A new learning mechanism is incorporated with the ABC in which the individuals update the positions through learning from all the neighbouring solutions as well as the best solution exists in the swarm. The FABC is applied to solve the 105 LSJSSP instances in the following experiment. The results obtained by the FABC are compared with the state-of-art algorithms. The results analysis shows that the proposed approach to solving LSJSSP is competitive in the field of swarm intelligent based algorithms (SIA).

In continuation, a fitness-based particle swarm optimization (FitPSO) is developed and applied to solve the LSJSSP problem instances. A fitness-based solution update strategy is incorporated with the PSO strategy to get the desired results in the proposed solution. The obtained outcome is motivating, and through results analysis, confidence is achieved that the proposed FitPSO can be a recommendation to solve the existing and the new LSJSSP instance. A fair comparative analysis is also presented, which also supports the proposed recommendation.

Candidate Declaration

I, hereby, certify that the work, which is being presented in the thesis, entitled “**DEVELOPMENT OF IMPROVED ABC AND PSO ALGORITHMS FOR SPECIFIC NIA**” in partial fulfillment of the requirement for the award of the degree of Doctor of Philosophy, carried under the supervision of Dr. P. C. Gupta and submitted to the University of Kota, Kota represents my ideas in my own words and where others ideas or words have been included. I have adequately cited and referenced the original sources. The work presented in this thesis has not been submitted elsewhere for the award of any other degree or diploma from any Institutions.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Kota

Kavita Sharma

Date:

**This is to certify that the above statement made by Kavita Sharma
Enrol No: 2018/000148 is correct to the best of our knowledge.**

Dr. P. C. Gupta

Place: Kota

Dept. of Computer Science & Informatics

Date:

University of Kota, Kota

Acknowledgements

I would like to extend my gratitude to my supervisor **Dr. P. C. Gupta**, Head and Associate Professor, Department of Computer Science and Informatics, University of Kota for his guidance, support, understanding, and patience. I have been amazingly fortunate to have a best mentor and advisor I could have ever wished. I learned a lot from his advice and useful insights throughout my Ph.D. program. His regular encouragement regarding referred publications helped me a lot to overcome my doubts and to finish this dissertation work. I am very thankful to him for his enormous knowledge, encouragement, and motivation at every stage of my research.

I express my gratitude to **Prof. Neelima Singh**, Honorable Vice-Chancellor, University of Kota, for creating a healthy environment for research in the University.

I am also grateful to faculty members of the Department of Computer Science and Informatics **Prof. Reena Dadhich, Dr. O.P. Rishi**, and **Dr. Krishna Kumar Sharma** for their positive and constructive suggestions during this tenure which helped this thesis in a present shape.

I am also thankful to Director Research and Controller of Examinations for making the process smooth and straightforward. A Special thanks to Dr. Nir-mala Sharma, Assistant Professor, RTU Kota for her consistent support during this period.

Without my family and friends' support, it would not have been possible to achieve this goal, so a big thanks to my parents Sh. S.M. Sharma, Smt. Sulochna Sharma, my in-laws Sh. Shivraj Sharma, Smt. Savitri Sharma, my family members, relatives, my husband, my children, friends for their consistent motivation and support always.

I am thankful to all the members of Department of Computer Science and Informatics for extending their support to me as and when required. Finally, I would like to thank everybody who was important to the successful realization of the thesis. I would like to express my apology to many more persons that I could not mention personally one by one.

Kavita Sharma

Table of Contents

Title	Page No.
List of Figures	i
List of Tables	ii
List of Abbreviations	iii
Chapter 1 Introduction	1
1.1 Introduction	1
1.2 Motivation	2
1.3 Key Research Area	3
1.4 Thesis Organization	3
Chapter 2 Literature Review	5
2.1 Artificial bee colony (ABC) algorithm	5
2.1.1 Initialization of the swarm phase	5
2.1.2 Employed bee phase	6
2.1.3 Onlooker bee phase	6
2.1.4 Scout bee phase	6
2.1.5 Main steps of ABCA	6
2.2 Significant advancements in ABCA	7
2.3 PSO Algorithm	7
2.4 Recent Modifications in Particle Swarm Optimization Algorithm	11
2.4.1 Modifications in PSO	11
2.4.2 Hybridization of PSO	13
2.5 Research gaps and objectives	14
2.5.1 Research gaps	14
2.5.2 Objectives	15
2.6 Research methodology	15
Chapter 3 Limaçon inspired artificial bee colony algorithm for numerical optimization	17
3.1 Introduction	17
3.2 Artificial bee colony algorithm	18
3.2.1 Initialization stage:	18
3.2.2 Employed honey bee stage:	18
3.2.3 Onlooker honey bee stage:	18
3.2.4 Scout honey bee stage:	19
3.3 Limaçon inspired ABCA	19

3.4	Results analysis	21
3.5	Conclusion	26
Chapter 4 Fully Informed ABC Algorithm		27
4.1	Introduction	27
4.2	Fully Informed ABC	28
4.3	Results Analysis	29
4.3.1	Test problems under consideration	29
4.3.2	Parameter setting for experiments	32
4.3.3	Reported Results	32
4.3.4	Statistical Analysis	34
4.4	Conclusion	37
Chapter 5 FABC Algorithm for Large Scale Job Shop Scheduling		39
5.1	Introduction	39
5.2	Fully Informed ABC	40
5.2.1	Initialization of the solution agents	40
5.2.2	Employed bee phase	41
5.2.3	Onlooker bees phase	41
5.2.4	Scout bees phase	42
5.3	Job shop scheduling problem organisation	42
5.4	FABC for LSJSSP	42
5.4.1	Initialization stage	43
5.4.2	Employed honeybee stage	43
5.4.3	Onlooker honeybee stage	43
5.4.4	Scout honeybee stage	44
5.5	Implementation and experimental results	44
5.6	Conclusion	48
Chapter 6 Fitness based Particle Swarm Optimization		51
6.1	Introduction	51
6.2	Standard PSO Strategy	52
6.3	Fitness Based PSO	53
6.4	Experiments and Results	56
6.4.1	Benchmark functions for testing	56
6.4.2	Experimental setting	56
6.4.3	Results Analysis of Experiments	57
6.4.4	Statistical Analysis	58
6.5	Applications of FitPSO to Engineering Optimization Problems	60
6.5.1	Compression Spring:	61
6.5.2	Lennard-Jones :	61
6.5.3	Welded beam design optimization problem:	62
6.5.4	Experimental Results	62
6.6	Conclusion	63
Chapter 7 FitPSO for Large Scale Job Shop Scheduling		65
7.1	Introduction	65
7.2	Fitness Based PSO	66

7.3	Job shop scheduling problem organisation	69
7.4	FitPSO for LSJSSP	71
7.5	Implementation and experimental results	73
7.6	Conclusion	77
Chapter 8	Conclusion and Future Work	79
8.1	Conclusion	79
8.2	Future Work	80
Summary	81
Bibliography	83
Publications	93

List of Figures

Figure No.	Title	Page No.
2.1	Research Flow	16
3.1	Sum of success versus (T_g)	23
3.2	Boxplots graphs for average number of function evaluation	25
4.1	Average number of function evaluation through Boxplots graph	34
4.2	Performance index (PI) for considered benchmark functions; (a) for case (1), (b) for case (2) and (c) for case (3).	36
5.1	Random Key (RKE) Encoding Scheme	43
6.1	Boxplots graph for average number of function evaluation	58
6.2	Acceleration Rate of FitPSO as compared to ABC and PSO	60
6.3	Convergence graph for ABC, PSO and FitPSO on test problems $f_6, f_9, f_{10},$ f_{11}	60
7.1	Random Key (RKE) Encoding Scheme	71

List of Tables

Table No.	Title	Page No.
2.1	ABC modifications: Introducing new strategies	8
2.2	ABC modifications: Hybridization with other algorithms	9
2.3	ABC modifications: Incorporation of local search strategies	10
3.1	Benchmark Set D: Dimensions, C: Characteristic, U: Unimodal, M: Multi-modal, S: Separable, N: Non-Separable, AE: Acceptable Error	22
3.2	Result assessment of functions with LABC and other state-of-art algorithms .	23
3.2	Result assessment of functions with LABC and other state-of-art algorithms (Cont.)	24
3.3	Acceleration Rate (AR) of LABC as compare to other state of art algorithms	25
3.4	Mann-Whitney U rank sum test on AFEs of LABC and compared strategies, TP: Test Problem.	26
4.1	Test problems	30
4.1	Test problems (Cont.)	31
4.2	Comparison based on average function evaluations, TP: Test Problem.	32
4.3	Comparison based on success rate out of 100 runs, TP: Test Problem.	33
4.4	Comparison based on mean error, TP: Test Problem.	33
4.5	The Mann-Whitney U rank sum test at a $\alpha = 0.05$ significance level was used to do a comparison based on mean function evaluations ('+' means FABC is considerably better, '-' means FABC is much worse, and '=' means there is no significant difference.), TP: Test Problem.	35
4.6	Acceleration Rate (AR) of FABC as compared to the ABC, BSFABC, GABC, and MABC, TP: Test Problems	37
5.1	Comparison in terms of best MS value for SMV instances	45
5.2	Comparison in terms of best MS value for TA instances	46
5.3	Comparison in terms of best MS value for DMU instances	47
5.4	Comparison based upon Average RPE	49
6.1	Test problems; AE: Acceptable Error	56
6.2	Comparison based on average number of function evaluations, TP: Test Problem.	57
6.3	Comparison based on success rate out of 100 runs, TP: Test Problem.	57
6.4	Comparison based on mean error, TP: Test Problem.	58

6.5	Comparison based on mean function evaluations and the Mann-Whitney U rank sum test at a $\alpha = 0.05$ significance level ('+' indicates FitPSO is significantly better, '-' indicates FitPSO is worse and '=' indicates that there is no significant difference), TP: Test Problem.	59
6.6	Acceleration Rate (AR) of FitPSO as compared to ABC and PSO, TP: Test Problems	59
6.7	Comparison of the results of test problems; TP: Test Problems	63
6.8	Comparison based on mean function evaluations and the Mann-Whitney U rank sum test at a $\alpha = 0.05$ significance level ('+' indicates FitPSO is significantly better), TP: Test Problem.	63
7.1	Comparison in terms of best MS value for SMV instances	74
7.2	Comparison in terms of best MS value for TA instances	75
7.3	Comparison in terms of best MS value for DMU instances	76
7.4	Comparison based upon Average RPE	77

List of Abbreviations

ABCA	Artificial Bee Colony Algorithm
ACO	Artificial Immune System
AIS	Ant Colony Optimization
bBCS	Binary Binomial Cuckoo Search
BBO	Biogeography-based optimization
BCS	Binomial Cuckoo Search
bCSA	Binary Crow Search Algorithm
BCSO	Binary Competitive Swarm Optimizer
bGA	Binary Genetic Algorithm
BGWO	Binary Grey Wolf Optimizer
CCA	Canonical Correlation Analysis
CS	Cuckoo Search
DE	Differential Evolution
EAs	Evolutionary Algorithms
FFSS	Fission-Fusion Social Structure
FABC	Fully Informed ABC
FitPSO	Fitness based PSO
GA	Genetic Algorithms
GP	Genetic Programming
GSA	Gravitational Search Algorithm
GWO	Grey Wolf Optimizer
HS	harmony search
LSJSS	Large Scale Job Shop Scheduling
MA	Memetic Algorithms
MBO	Monarch Butterfly Optimization Algorithm
MeSMO	Memetic Spider Monkey Optimization
ML	Machine Learning
MSE	Mean Squared Error
NB	Naive Bayes
NIA	Nature-Inspired Algorithms
NLP	Natural Language Processing
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SD	Standard Deviation
SMO	Spider Monkey Optimization
SMOK	Spider-Monkey-Optimizer with k-means clustering
SSA	Salp Swarm Algorithm

SIA	Swarm intelligence based algorithms
TLBO	Teaching Learning based Optimization Algorithm
WOA	Whale Optimization Algorithm

CHAPTER 1
INTRODUCTION

Chapter 1

Introduction

This chapter discusses an overview of the thesis. It demonstrates the objective of carrying out this work as well as delineates the inspiration behind the research.

1.1 Introduction

Nature has always been a source of motivation for the human being in technological advancements. In recent years, nature-inspired algorithms (NIAs) have gained a lot of interest from many researchers. Many complex optimization problems are not easily addressable using the present deterministic optimization approaches. To find out the near-optimal solution, NIAs are an alternate solution for these kinds of problems. The NIAs are mainly classified as evolutionary algorithms (EA) and swarm intelligence (SI) based algorithms. The EAs are based on various biological evolution mechanisms like selection, mutation, crossover, etc. Differential evolution (DE) [1], genetic algorithm (GA) [2] etc., are some popular evolutionary algorithms. Many natural phenomena that depict the grouping behavior of insects, animals, birds, etc., inspire researchers to develop different sorts of optimization algorithms known as SI based algorithms. These algorithms became popular in a short time due to the advent of computational intelligence. In recent years, SI has gained a lot of interest from many researchers. The collective, cooperative, and intelligent behavior of various swarms of social insects, birds, animals, etc., is the basic building block for SI based algorithms. Artificial bee colony (ABC), spider monkey optimization (SMO), ant colony optimization (ACO), etc., are some popular SI based algorithms. Researchers are continuously working in the field of SI motivating techniques to refine their performance. To enhance the efficiency, the available algorithms are modified by introducing new local search (LS) strategies, by introducing new phase(s), or by hybridizing with a different kind of population-based algorithms [3]. Further, SI based algorithms are proving their efficiency in solving complex real-world optimization problems.

The ABC algorithm is efficient in the arena of SI based strategies. The ABC algorithm was presented by D. Karaboga in the year 2005 [4]. The ABC algorithm is mathematically simulated on the food foraging conduct of natural honey bees. As per the available literature, the ABC algorithm is an efficient algorithm to discover the solutions of several complex real-world optimization problems [5, 6, 7]. There is always another side of the coin, ABC also may sometimes stop advancing close to global optimum although the solutions did not converge to local optimum [8]. It is reported in the literature that the search procedure of ABC is favorable in terms of exploration behavior but not in terms of exploitation behavior [9].

Further, It can be observed that sometimes the ABC algorithm may confine in the situation of stagnation and premature convergence [9, 7]. So there is always a possibility to get jump of the actual solution. The researchers are working to restructure the ABC algorithm to refine its performance. Thus, the incorporation of LS methodology may improve the exploitation capacity of the ABC and subsequently decreasing the chance of jumping the actual solution. This research work is also a contribution to boost the performance of the ABC algorithm.

Firstly, an influential local search (LS) technique that is designed by taking inspiration by limaçon a curve is incorporated in ABCA, and the designed strategy is named Limaçon inspired ABC (LABC) algorithm. Then, the exploitation capability of the LABC strategy is tested over 18 complex benchmark optimization problems. Finally, the test results are compared with similar state-of-art algorithms, and statistical analysis shows the LABC can be considered a practical variant of the ABC algorithms to solve the complex optimization problems.

A novel swarm intelligence-based algorithms (SIA) is applied to solve the 105 LSJSSP instances in the following experiment. The selected SIA is the Fully Informed Artificial Bee Colony (FABC) algorithm. The FABC algorithm is developed by taking inspiration from the GABC algorithm position update process. In the FABC, the onlooker bee process of the Artificial Bee Colony (ABC) algorithm is modified and designed such that the new position of the solution search agent is obtained while learning from all the nearby agents. The results obtained by the FABC are compared with the state-of-art algorithms. The results analysis shows that the proposed approach to solving LSJSSP is competitive in the field of SIA.

In continuation, a fitness-based particle swarm optimization (FitPSO) is developed and applied to solve the LSJSSP problem instances. A fitness-based solution update strategy is incorporated with the PSO strategy to get the desired results in the proposed solution. The obtained outcome is motivating, and through results analysis, confidence is achieved that the proposed FitPSO can be a recommendation to solve the existing and the new LSJSSP instance. A fair comparative analysis is also presented, which also supports the proposed recommendation.

1.2 Motivation

The real-world engineering optimization problems are modeled with multi-dimensional functions. These problems are not easily addressed. The properties associated with such problems are discontinuities, lack of analytical representation of the objective function, and noise dissemination. The salient features to numerically optimize these problems through algorithmic operations are computation accuracy, time criticality, and implementation efforts. In these circumstances, the available conventional mathematical strategies do not perform well to solve these problems. The conventional strategies are nonrobust and time-consuming or sometimes unable to solve these problems. This motivates the researchers to design and develop a new class of algorithms namely SI based algorithms. The SI based algorithms are robust enough to apply to various sets of problems and are fast enough to execute near-optimal solutions. The SI based algorithms do also have the drawbacks of stagnation and premature convergence. This further motivates researchers to develop an efficient SI based algorithm that can be applied to solve these problems.

1.3 Key Research Area

There are two key elements responsible for the execution of any SI based algorithm to drive the members of a swarm in terms of potential solution update: the *variation process*, responsible to explore different search space areas that is social learning and the *selection process*, responsible to ensures the exploitation of the erstwhile experience that is self intelligence.

The reviewed literature during the research demonstrated that SI based algorithms at seldom may stop converging to the global optimum even though the set of potential solutions has not converged to a local optimum. Therefore, to eliminate the existing drawbacks of SI based algorithms researchers are working to nurture a genuine equilibrium amid the exploration and exploitation capabilities. Sequentially, in this research a significant SI based algorithm namely, ABC and PSO algorithms are chosen as a key research area. The reviewed literature also reports that ABC and PSO are quite reliable and efficient algorithm in the field of SI based algorithms. Distinguishable growth in the publications on these algorithms since the last decade is enough to justify its wide applicability to real-world optimization problems. The literature reports that the performance of the existing variants of ABC and PSO algorithms to be enhanced. The competitiveness of the designed new variants of ABC and PSO algorithms may be established using a set of benchmark problems. Further, the designed variant of ABC and PSO algorithms may be practiced to solve significant real-world engineering problems specifically, job shop scheduling problem (JSSP) to authenticate their existence.

1.4 Thesis Organization

The thesis is organized into a total of eight chapters. A brief outline of each chapter is as follows:

Chapter 1: This chapter introduces a source of motivation to carry out the research. The key research area is defined. Further, research objectives are defined with the research methodology.

Chapter 2: This chapter presents an overview of the ABC and PSO algorithms and an extensive literature review on ABC and PSO. This chapter also discusses their applications.

Chapter 3: This chapter presents Limacon inspired artificial bee colony algorithm for numerical optimization.

Chapter 4: This chapter presents a modified ABC based algorithm namely, Fully Informed ABC (FABC).

Chapter 5: In this chapter, the designed FABC is applied to solve the 105 significant instances of large scale JSSP.

Chapter 6: This chapter presents an efficient PSO variant namely, Fitness based PSO (FitPSO).

Chapter 7: In this chapter, the FitPSO strategy is tested over 105 large scale job shop scheduling instances. Further, a comparative analysis of FABC and FitPSO is presented over the JSSP.

Chapter 8: This chapter summarizes the key outcomes and main contributions of the thesis and lists further research paths.

CHAPTER 2
LITERATURE REVIEW

Chapter 2

Literature Review

This chapter presents the structure of the ABC and PSOAs. Further, it demonstrates the recent advancements designed and proposed in these two algorithms by various researchers to boost its efficiency and execution abilities.

2.1 Artificial bee colony (ABC) algorithm

SI borne strategies rely upon the collective sensible act of various species found in nature. Due to the advent of computational intelligence, SI based algorithms are very popular strategies in a short period. Latest SI algorithms include ABC [4], particle swarm optimization (PSO) [10], teaching learning-based optimization (TLBO) [11], SMO [12], and so on.

In the ABCA, the food source's position demonstrates a conceivable solution for the optimization problem. The nectar measure of a food source proportionates to the fitness of the solution [8]. The colony of the artificial bees is subdivided into three groups namely employed bees, onlooker bees, and scout bees. The employed bees and onlooker bees are equal to the number of food sources. The employed bees randomly search for the position of a food source in the search space. They further communicate their experience with the onlooker bees to assist them in the searching process. Once the food is exhausted, the scout bees randomly search the food source by its internal motivation [8].

ABC is an iterative procedure alike other population-based metaheuristic algorithms. For a complete execution, it performs cycles of the four phases namely, initialization of the swarm phase, employed bee phase, onlooker bee phase, and scout bee phase [13]. The explanation of these phases is presented in succeeding sections.

2.1.1 Initialization of the swarm phase

At first, ABC generates an evenly scattered initial population of N solutions where every solution or food source x_i ($i= 1, 2, . . . ; N$) represents a vector of dimension D. Here, D denotes the total decision variables of the optimization problem and x_i denotes the i^{th} food source of the population. The food sources are originated by the following equation 2.1:

$$x_{ij} = x_{minj} + rand[0, 1](x_{maxj} - x_{minj}) \quad (2.1)$$

The limits of x_i are x_{minj} and x_{maxj} in j^{th} direction further, $[0, 1]$ represents an evenly scattered arbitrary number in the range $[0, 1]$.

2.1.2 Employed bee phase

In this duration, the current solution is updated relied on the information furnished by the experience of the individual and the fitness value of the newly generated solution i.e. nectar measure. If the fitness value of the new solution is higher in comparison with the old solution, the bee replaces its position with the new one and rejects the old one [13]. For i^{th} candidate solution the position update equation is

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (2.2)$$

Where, $k \in \{1, 2, \dots, N\}$ and $j \in \{1, 2, \dots, D\}$ are arbitrarily selected indices, k must be distinct from i , and ϕ_{ij} is an arbitrary number amid $[-1, 1]$.

2.1.3 Onlooker bee phase

The employed bees share their knowledge with onlooker bees waiting in the hive. The onlooker bees look over the received knowledge and they choose a food source according to the probability value which is defined as a function of fitness. The probability p_i may be evaluated as equation 2.3.

$$p_i = \frac{0.9 \times fit_i}{maxfit} + 0.1 \quad (2.3)$$

Where, fit_i depicts the fitness value of the i^{th} solution and $maxfit$ is the maximum fitness value in the swarm. Now, it creates a modification in the position in its memory and checks for the fitness of the candidate source like employed bees did earlier. If the new fitness is higher than that of the previous one, the bee memorizes the newly generated position and overlooks the old one.

2.1.4 Scout bee phase

The food source is thought to be neglected if the position of a food source is not modified up to a predetermined limit i.e number of cycles and then the scout bee phase begins. In this duration, the food source is substituted by an arbitrarily picked food source within the specified region. Assume that the neglected food source is x_i and $j \in \{1, 2, \dots, D\}$ then the scout bee substitutes this food source with x_i . This execution can be characterized as equation 2.4.

$$x_{ij} = x_{minj} + rand[0, 1](x_{maxj} - x_{minj}) \quad (2.4)$$

The limits of x_i are x_{minj} and x_{maxj} in j^{th} direction further, $[0, 1]$ represents an evenly scattered arbitrary number in the range $[0, 1]$.

2.1.5 Main steps of ABCA

The searching mechanism of ABCA consists of three control parameters namely, total food sources (N), the threshold limit, and the total number of cycles. The ABC is demonstrated as per Algorithm 2.1:

Algorithm 2.1 ABC Algorithm (ABCA) Structure:

ABCA variables initialization
while Criteria for termination do

- Phase 1: Employed bee Step
- Phase 2: Onlooker bee Step
- Phase 3: Scout bee Step
- Phase 4: Identify the best solution

end while
Output the best solution available in the swarm

2.2 Significant advancements in ABCA

The ABC is a significant contribution since it's inception, the researchers have been continuously working to refine the performance of ABC to make it more efficient. As per the available literature, the studies on ABC may be categorized into three subfields:

- Introducing new strategies/phases
- Hybridization with other algorithms
- Incorporation of LS strategies

The significant contribution in this field is tabulated.

2.3 PSO Algorithm

Particle Swarm Optimization Algorithm (PSOA) is a swarm intelligence based strategy which is designed while taking inspiration from the birds flocking behavior. PSOA consists active, interactive individuals with relatively little innate intelligence. In PSO, the entire group is referred to as a *swarm*, while each individual is referred to as a *particle*, which represents a potential candidate's answer. By analysing the behaviour of adjacent birds who looked to be near the food source, the swarm discovers food for itself through social learning.

Initially, each particle is randomly started inside the search area and remembers information about its personal best position, p_{best} , swarm best position, g_{best} , and current velocity, V , with which it is travelling. Each particle adjusts its location based on these three variables. In this manner, whole swarm moves in better direction while following collaborative trail and error method and converges to single best known solution.

The i^{th} particle of the swarm is represented by a D-dimensional vector, $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$, in a D-dimensional search space. Another D-dimensional vector $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ represents the velocity of this particle. $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$ is the previously best visited place of the i^{th} particle. g is the index of the best particle in the swarm. For movement, the PSO swarm employs two equations: the *velocity update equation* and the *position update*

Table 2.1: ABC modifications: Introducing new strategies

Year/Author	Title	Research Contribution
Zhu et. al (2010) [9]	“G-best guided ABC for numerical function optimization”	Incorporates the information of global best (gbest) solution into the solution search equation to improve the exploitation.
Minghao Yin et al. (2011) [14]	“Modified artificial bee colony algorithm based on mutation operations”	Three mutation operations with local search in ABC.
Yu Liu et.al (2012) [15]	“Improved ABCA with mutual learning”	It adjust the produced candidate food source with the higher fitness between two individuals selected by a mutual learning factor.
Kazunori Mizuno et. al (2013) [16]	“Experimental Evaluation of Artificial Bee Colony with Greedy Scouts for Constraint Satisfaction Problems”	Two main improvements are adopted: (1) a hybrid algorithm with greedy local search technique, called GSAT is combined and (2) in the scout bee phase, greedy scout bees are introduced.
Bansal et al. (2014) [17]	“Self-adaptive artificial bee colony”	A adaptive strategy is developed.
Alkin et.al (2015) [18]	“An Enhanced Artificial Bee Colony Algorithm with Solution Acceptance Rule and Probabilistic Multisearch”	A new solution acceptance rule is proposed where, instead of greedy selection between old solution and new candidate solution, worse candidate solutions have a probability to be accepted.
Lie et al. (2016) [19]	“Artificial bee colony algorithm with memory”	A memory mechanism is developed in which a bee memorizes its previous successful experiences of foraging behaviour.
Xiang et al. (2018)[20]	“An improved artificial bee colony algorithm based on the gravity model”	Inspired by the gravity model, an attractive force model is proposed for choosing a better neighbor of a current individual to improve the exploitation ability of ABC.
Bajer et al. (2019) [21]	“An effective refined ABCA for numerical optimisation”	Introduces a new solution update equation and an extended scout bee phase focusing the search on more prominent solutions without introducing new control parameters.
Santana et al. (2019) [22]	“A novel binary artificial bee colony algorithm”	Presents a novel mechanism which limits the number of dimensions that can be changed in the employed and onlookers bees phase.

equation. The velocity of the i^{th} particle is updated by solving the velocity update equation (2.5). Equation (2.6) is used to update the position.

$$v_{ij} = v_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (p_{gj} - x_{ij}) \quad (2.5)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (2.6)$$

where $j = 1, 2, \dots, D$ represents the dimension and $i = 1, 2, \dots, S$ represents the particle index. S is the swarm’s size, and c_1 and c_2 are constants (typically $c_1 = c_2$), often known as cognitive and social scaling parameters or simply acceleration coefficients, respectively. r_1 and r_2 are uniformly distributed random numbers in the range $[0, 1]$.

There are three terms on the right hand side of the velocity update equation (2.5). The first term, v_{ij} , reflects the particle’s memory of its previous movement direction, which may be thought of as a momentum term that prevents the particle from abruptly reversing direction. The second term, $c_1 r_1 (p_{ij} - x_{ij})$, refers to the cognitive component of persistence, which pulls particles back to their previous optimum condition and enables swarms to explore locally. The last term, $c_2 r_2 (p_{gj} - x_{ij})$, is referred to as the social component and is in charge of global search. Individuals might use it to compare themselves to others in their peer group. The following is a description of the Pseudo-code for Particle Swarm Optimization:

Table 2.2: ABC modifications: Hybridization with other algorithms

Year/Author	Title	Research Contribution
Ali R.Yildiz (2013) [23]	“A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing”	ABC is hybridized with Taguchi method to obtain better performance.
Farahani et al. (2014) [24]	“A hybrid artificial bee colony for disruption in a hierarchical maximal covering location problem”	ABC is hybridized with 2-opt as a local search to find the local optimum in each iteration.
Li et al. (2015) [25]	“Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm”	A novel hybrid algorithm (TABC) that combines the artificial bee colony (ABC) and tabu search (TS) to solve the hybrid flow shop (HFS) scheduling problem with limited buffers.
Gao et al. (2016) [26]	“Enhanced ABCA through differential evolution”	Global best ABC and DE algorithm are hybridized to achieve the better performance.
Jadon et al. (2017) [27]	“Hybrid artificial bee colony algorithm with Differential Evolution”	A hybrid algorithm (HABCDE) based on ABC and DE is proposed.
Meng et al. (2018) [28]	“A hybrid artificial bee colony algorithm for a flexible job shop scheduling problem with overlapping in operations”	A dynamic scheme is introduced to fine-tune the search scope adaptively. A modified migrating birds optimisation algorithm (MMBO) integrated into the search process.
Wang et al. (2019) [29]	“An efficient hybrid artificial bee colony algorithm for disassembly line balancing problem with sequence-dependent part removal times”	A new rule is used to initialize a bee colony population with certain diversity, and a dynamic neighbourhood search method is introduced to guide the employed/onlooker bees to promising regions. To rapidly leave the local optima, a global learning strategy is employed to explore higher quality solutions.
Karaboga et al. (2019) [30]	“Training ANFIS by using an adaptive and hybrid artificial bee colony algorithm (aABC) for the identification of nonlinear static systems”	An adaptive and hybrid artificial bee colony (aABC) algorithm is employed in ANFIS training. The aABC algorithm uses arithmetic crossover and adaptive neighborhood radius in the solution generating mechanism.
Li et al. (2019) [31]	“Hybrid Artificial Bee Colony Algorithm for a Parallel Batching Distributed Flow-Shop Problem With Deteriorating Jobs”	A novel scout bee heuristic that considers the useful information that is collected by the global and local best solutions is investigated.

Initially, two variants of PSO were published in the literature based on neighbourhood size: the global version of PSO (PSO-G), which is the original PSO, and the local version of PSO (PSO-L) [39]. The term p_g in the social component of the velocity update equation (2.5) is the only difference between PSO-G and PSO-L. It refers to the best particle of the whole swarm in PSO-G, and the best particle of the individual’s vicinity in PSO-L. The PSO-social G’s network is based on the star topology, which allows for faster convergence but is more prone to converge prematurely. PSO-L, on the other hand, employs a ring social network architecture, with smaller regions specified for each particle. It is clear that because PSO-L has less particle interconnectivity, it is less prone to being caught in local minima, albeit at the cost of delayed convergence. PSO-G is better for unimodal situations whereas PSO-L is better for multimodal ones.

The balance between PSO’s exploration and exploitation capabilities is determined by the velocity update equation. Because no velocity restrictions were set in Basic PSO, particles distant from gbest will take huge steps in early iterations and are extremely likely to abandon the search space. The velocity clamping idea was proposed to regulate velocity so that particle update step size is balanced. When velocity exceeds its boundaries, velocity clamping is used to keep it there. To prevent velocity clamping and strike a balance between

Table 2.3: ABC modifications: Incorporation of local search strategies

Year/Author	Title	Research Contribution
Kang et al. (2011)[32]	“Artificial Bee Colony Algorithm with Local Search for Numerical Optimization”	Hooke and Jeeves pattern search method is added.
Bansal et al. (2013) [33]	“Memetic search in artificial bee colony algorithm”	A new local search phase is integrated with the basic ABC in which the step size of the best solution is controlled by golden section search approach.
Sharma et al. (2014) [34]	“Power law-based local search in artificial bee colony”	Power law-based local search strategy is proposed and integrated with ABC.
Sharma et al. (2016) [35]	“Lvy flight artificial bee colony algorithm”	A Lvy flight inspired search strategy is proposed and integrated with ABC. New solutions are generated around the best solution.
Dogan et al. (2019) [36]	“Improved Self-adaptive Search Equation-based Artificial Bee Colony Algorithm with competitive local search strategy”	Integrates three strategies into the ABC algorithm: 1. Self-adaptive strategy 2. Competitive local search selection 3. Incremental population size
D. Karaboga et al. (2014)[37]	“A quick artificial bee colony (qABC) algorithm and its performance on optimization problems”	So, it can be said that onlookers choose their food sources in a different way from employed bees. in qABC algorithm, a new definition is introduced for the behaviour of onlookers.
Neha et al. (2017)[38]	“Improved local search based modified ABC algorithm for TSP problem”	After the scout bee phase, one supplementary phase in the form of mutation operator (which is the part of genetic algorithm) is used.

exploration and exploitation, a new parameter called inertia weight [40] was added to the velocity update equation, as follows:

$$v_{ij} = w * v_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (p_{gj} - x_{ij}) \quad (2.7)$$

where inertia weight is denoted by w . In subsequent section, the proposed PSOA is explained in details.

Algorithm 2.2 PSO Algorithm (PSOA):

PSOA parameters, w , c_1 and c_2 are initialized.
 In the search space, initialise the particle locations and velocities.
 Individual particle fitness is assessed.
 Memorize gbest and pbest;
while Termination condition(s) not true **do**
 for each particle, X_i **do**
 for each dimension j , x_{ij} **do**
 (i) Update the velocity v_{ij} ;
 (ii) Update the position x_{ij} ;
 end for
 end for
 Calculate the fitness of the updated particles.
 Through greedy selection approach get new values of gbest and pbest;
end while
 Return the solution to the particle with the best fitness;

2.4 Recent Modifications in Particle Swarm Optimization Algorithm

The PSO is now one of the most commonly used optimization techniques. Advances on PSO are end less, here we classify all the modifications in two different classes. The categorisation is as follow:

1. Modifications of PSO, including bare-bones PSO, chaotic PSO, fuzzy PSO, quantum-behaved PSO, opposition-based PSO and other significant modifications,
2. Hybridization of PSO with other nature inspired algorithms including ABC, ACO, DE, GA, artificial immune system (AIS), Tabu search (TS), simulated annealing (SA), biogeography-based optimization (BBO), and harmonic search (HS),

2.4.1 Modifications in PSO

Jau et al.[41] proposed a modified quantum-behaved particle swarm optimization for parameters estimation of generalized nonlinear multi-regressions model based on Choquet integral with outliers. They modified the popular variant of quantum-behaved PSO and used a high breakdown regression estimator and a least-trimmed-squares method to eliminate the influence caused by observations containing outliers. Jamalipour et al. [42] presented QPSO with differential mutation operator (QPSO-DM) for optimizing WWER-1000 core fuel management. The results showed that QPSO-DM performs better than the others. Bagheri et al. [43] used QPSO to forecast financial time series, especially for the foreign exchange market. Tang et al. [44] proposed an improved QPSOA for continuous nonlinear large-scale problems based on memetic algorithm and memory mechanism. The memetic algorithm was used to make all particles gain some experience through a local search before being involved in the evolutionary process, and the memory mechanism was used to introduce a bird kingdom with memory capacity, both of which can improve the global search ability of the algorithm. Davoodi et al. [45] proposed a new approach, based on a hybrid algorithm combining of improved QPSO and simplex algorithms. QPSO was the main optimizer of algorithm, which can give a good direction to the optimal global region. Nelder-Mead simplex method was used as a local search to fine-tune the obtained solution from QPSO.

J Kennedy proposed a variant of PSO namely bare-bones PSO (BBPSO) [46]. In which the velocity and position update rules are substituted by a procedure that samples a parametric probability density function. Zhang et al. [47] used both mutation and crossover operators of DE algorithm to modify original BBPSO in order to update certain particles in the population. The performance of the resulting algorithm was tested on 10 benchmark functions and applied to 16 vapor-liquid equilibrium problems. Zhang et al. [48] analyzed the sampling distribution in BBPSO, based on which they propose an adaptive version inspired by the cloud model, which adaptively produced a different standard deviation of the Gaussian sampling for each particle according to the evolutionary state in the swarm, which provided an adaptive balance between exploitation and exploration on different objective functions. Zhang et al. [49] proposed three global optimization algorithms for phase and chemical equilibrium calculations, which played a significant role in the simulation, design, and optimization of separation processes in chemical engineering. The proposed algorithms were unified BBPSO (UBBPSO), integrated DE (IDE), and IDE without Tabu list and radius (IDE_N).

Chaos theory have been integrated with PSO to improve its performance. This type of PSO variant is called chaotic PSO (CPSO). Chuang et al. [50] introduced chaotic maps into catfish particle swarm optimization. The proposed method increased the search capability via the chaos approach. Zhang and Wu [51] proposed adaptive CPSO (ACPSO) to train the weights/biases of two-hidden-layer forward neural network in order to develop a hybrid crop classifier for polarimetric synthetic aperture radar images. Dai et al. [52] proposed a novel adaptive chaotic embedded PSO (ACEPSO) and applied it in wavelet parameters estimation. ACEPSO embedded chaotic variables in standard PSO and adjusted parameters nonlinearly and adaptively. It also estimated particles whether being focusing or discrete by judging the population fitness variance of particle swarm and average distance amongst points; then chaotic researching was applied to escaping from premature convergence. Li et al. [53] proposed a novel chaotic particle swarm fuzzy clustering (CPSFC) algorithm based on a new CPSO and gradient method. The new CPSOA is the combination of adaptive inertia weight factor (AIWF) and iterative chaotic map with infinite collapses (ICMIC) based chaotic local search. The CPSFC algorithm utilized CPSO to search the fuzzy clustering model, exploiting the searching capability of fuzzy c-means (FCM) and avoiding its major limitation of getting stuck at locally optimal values. Meanwhile, gradient operator is adopted to accelerate convergence of the proposed algorithm.

In order to make PSO more powerful, it was combined with fuzzy sets theory. This type of PSO variant is called fuzzy PSO (FPSO). Juang et al. [54] proposed an adaptive FPSO (AFPSO) algorithm. The proposed AFPSO utilized fuzzy set theory to adjust PSO acceleration coefficients adaptively and was thereby able to improve the accuracy and efficiency of searches. Incorporating this algorithm with quadratic interpolation and crossover operator further enhanced the global searching capability to form a new variant called AFPSO-Q1. Alfi and Fateh [55] presented a novel improved FPSO (IFPSO) algorithm to the intelligent identification and control of a dynamic system. The proposed algorithm estimated optimally the parameters of system and controlled by minimizing the mean of squared errors. The PSO was enhanced intelligently by using a fuzzy inertia weight to rationally balance the global and local exploitation abilities. Every particle dynamically adjusted inertia weight according to particles best memories using a nonlinear fuzzy model. Yang et al. [56] proposed a novel FPSOA based on fuzzy velocity updating strategy in order to optimize the machining parameters. The proposed FPSOA achieved good results on few benchmark problems and obtained significant improvements on two illustrative case studies of multipass face milling.

Opposition-based learning (OBL) theory was integrated with PSO, and the new variant was dubbed opposition-based PSO (OPSO). Dhahri and Alimi [57] proposed the OPSO using the concept of opposite number to create a new population during the learning process. They combined OPSO with BBFNN. The results showed that the OPSO-BBFNN produced a better generalization performance. Wang et al. [58] presented an enhanced PSO called GOPSO, which employed generalized OBL (GOBL) and Cauchy mutation. GOBL provided a faster convergence and the Cauchy mutation with a long tail helped trapped particles escape from local optima. Dong et al. [59] proposed an evolutionary circle detection method based on a novel chaotic hybrid algorithm (CHA). The method combined the strengths of PSO, GA, and chaotic dynamics and involved the standard velocity and position update rules of PSOs, with the ideas of selection, crossover, and mutation from GA. The OBL was employed in CHA for population initialization. In addition, the notion of species was introduced into the proposed CHA to enhance its performance in solving multimodal problems. Gao et al. [60] proposed a novel PSO called CSPSO to improve the performance of

PSO on complex multimodal problems. Specifically, a stochastic search technique was used to execute the exploration in PSO. In addition, to enhance the global convergence, when producing the initial population, both opposition-based learning method and chaotic maps were employed.

Some researchers make tentative research on improving the optimization performance of PSO by other efficient strategies. For example, Chuang et al. [61] proposed a novel catfish PSO, the mechanism of which is dependent on the incorporation of a catfish particle into the linearly decreasing weight particle swarm optimization. Unlike other ordinary particles, the catfish particles initialized a new search from the extreme points of the search space when the gbest fitness value had not been changed for a given time, which resulted in further opportunities to find better solutions for the swarm by guiding the whole swarm to promising new regions of the search space and accelerating convergence. Shi and Liu [62] proposed a hybrid improved PSO, in which chaos initialization was introduced to improve the population diversity, and adaptive parameters control strategy was employed to make it independent from specific problem. Besides, novel acceptance policy based on Metropolis rule was taken to guarantee the convergence of the algorithm. Zhang et al. [63] proposed a new adaptive PSO (APSO) that could dynamically follow the frequently changing market demand and supply in each trading interval. A numerical example served to illustrate the essential features of the approach.

2.4.2 Hybridization of PSO

PSO was combined with some traditional and evolutionary optimization algorithms in order to take the advantages of both methods and compensate the weaknesses of each other. This type of PSO is called hybridized PSO.

Kuo and Hong [64] presented a two-stage method of investment portfolio based on soft computing techniques. The first stage used data envelopment analysis to select most profitable funds, while hybrid of GA and PSO was proposed to conduct asset allocation in the second stage. Chen and Kurniawan [65] presented a two-stage optimization system to find optimal process parameters of multiple quality characteristics in plastic injection molding. Taguchi method, BPNN, GA, and combination of PSO and GA (PSO-GA) were used in this study to find optimum parameter settings. Nazir et al. [66] extracted facial local features using local binary pattern (LBP) and then fused these features with clothing features, which enhanced the classification accuracy rate remarkably. In the following step, PSO and GA were combined to select the most important features set that more clearly represented the gender and thus the data size dimension was reduced.

Tang et al. [67] presented a novel dynamic PSOA based on improved artificial immune network (IAINPSO). Based on the variance of the populations fitness, a kind of convergence factor was adopted in order to adjust the ability of search. The experimental results showed that not only did the new algorithm satisfy convergence precision, but also the number of iterations was much less than traditional scheme and had much faster convergent speed, with excellent performance in the search of optimal solution to multidimensional function. Zhang et al. [68] proposed a more pragmatic model for stochastic networks, which considered not only determinist variables but also the mean and variances of random variables. In order to accelerate the solution of the model, they integrated PSO with chaos operator and AIS.

Chen and Chien [69] presented a new method, called the genetic simulated annealing ant colony system with particle swarm optimization techniques, for solving the TSP. The

experimental results showed that both the average solution and the percentage deviation of the average solution to the best known solution of the proposed method were better than existing methods. Xiao et al. [70] considered the features of the MRCMPSP problem. They employed ant colonies labor division to establish a task priority-scheduling model firstly. Then, they used improved PSO to find out the optimum scheduling scheme. The approach integrating the above two algorithms had abilities of both local search and global search.

El-Abd [71] tested a hybrid PSO and ABC algorithm on the CEC13 testbed. The hybridization technique was a component-based one, where the PSO was augmented with an ABC component to improve the personal best of the particles. Sharma et al. [72] proposed a variant called Local Global variant ABC (LGABC) to balance the exploration and exploitation in ABC. The proposal harnessed the local and global variant of PSO into ABC. The proposed variant was investigated on a set of thirteen well-known constrained benchmarks problems and three chemical engineering problems, which showed that the variant can get high-quality solutions efficiently. Kiran and Gndz [73] presented a hybridization of PSO and ABC approaches, based on recombination procedure. The global best solutions obtained by the PSO and ABC were used for recombination, and the solution obtained from this recombination was given to the populations of the PSO and ABC as the global best and neighbor food source for onlooker bees, respectively.

Maione and Punzi [74] proposed a two-step design approach. First, DE determined the fractional integral and derivative actions satisfying the required time-domain performance specifications. Second, PSO determined rational approximations of the irrational fractional operators as low-order, stable, minimum-phase transfer functions with poles interlacing zeros. Extensive time- and frequency-domain simulations validated the efficiency of the proposed approach. Fu et al. [75] presented a hybrid DE with QPSO for the unmanned aerial vehicle (UAV) route planning on the sea. It combined DE algorithm with the QPSOA in an attempt to enhance the performance of both algorithms. Experimental results demonstrated that the proposed method was capable of generating higher quality paths efficiently for UAV than any other tested optimization algorithms.

2.5 Research gaps and objectives

This section represents the research gaps that are identified based on the inherent drawbacks reported in the reviewed literature of the ABC and PSO. Further, the objectives are devised based on identified research gaps.

2.5.1 Research gaps

Following research gaps are identified based on extensive literature review:

1. If a population based algorithm is capable of balancing between exploration and exploitation of the search space, then the algorithm is regarded as an efficient algorithm. From this point of view, basic ABC is not an efficient algorithm [8, 76, 6]. Dervis Karaboga and Bahriye Akay [76] compared the different variants of ABC for global optimization and found that ABC shows a poor performance and remains inefficient in exploring the search space. The problems of premature convergence and stagnation is a matter of serious consideration for designing a comparatively efficient ABC algorithm.

2. In ABC, the step size plays a vital role in modifying the potential solution within the current swarm. The step size is a function of an arbitrary number $\phi_{ij} \in [-1, 1]$, current solution, and an arbitrarily chosen neighbouring solution. The caliber of the modified solution relies on this step size. The large gap between the current solution and arbitrarily chosen solution with a high absolute value of ϕ_{ij} represents that the step size is too large, in this case, the modified solution may outpace the actual solution. Further, the lesser step size may decrease the convergence rate of ABC algorithm [9, 7, 13].
3. Most of the population based stochastic algorithms have the inherent drawback of premature convergence. PSO is not exceptions. Any population based algorithm is regarded as an efficient algorithm if it is capable of balancing between exploration and exploitation of the search space or is fast in convergence and able to explore the maximum area of the search space. From this point of view, researchers suggested that PSO is not away from the risk of premature convergence [77][78]. The problems of premature convergence and stagnation is a matter of serious consideration for designing a comparatively efficient PSOA.
4. Different Pseudo random search algorithms have optimized the different standard benchmark problems at different extents. This inspires researchers to develop new efficient or more hybrid algorithms, which could optimize as many functions as possible with better effectiveness and efficiency as compared to the existing algorithms.
5. As there is no fully efficient nature-inspired algorithms so it is highly required to develop a new efficient nature-inspired algorithm compared to the existing algorithms.
6. Different population based Pseudo-intelligent random search algorithms could optimize the given standard benchmark functions to different extents. This inspires researchers to develop more and more hybrid algorithms, which could optimize as many functions as possible with better effectiveness and efficiency.

2.5.2 Objectives

Objectives are formulated according to the research gaps as identified. In the proposed methodology, designed Objectives are as follows.

1. Modification in existing nature-inspired algorithms.
2. Development of new nature-inspired algorithm.
3. Testing the modified or newly developed algorithms over well-known benchmark optimization problems.
4. Implement the existing or modify or newly developed algorithms to solve real world complex optimization problem like job shop scheduling problem (JSSP), single machine total weighted tardiness problem (SMTWTP) etc.

2.6 Research methodology

This section illustrates the complete research methodology which is adopted to carry out the research work. Every step of the research process is briefly described in Fig. 2.1.

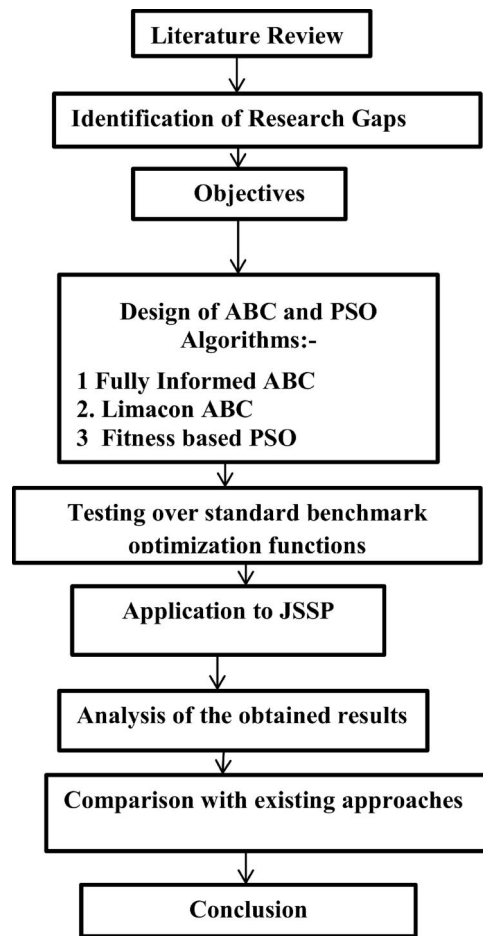


Figure 2.1: Research Flow

CHAPTER 3

LIMACON INSPIRED ARTIFICIAL BEE COLONY
ALGORITHM FOR NUMERICAL OPTIMIZATION

Chapter 3

Limaçon inspired artificial bee colony algorithm for numerical optimization

The artificial bee colony algorithm (ABCA) has established itself as a signature algorithm in the area of swarm intelligence based algorithms. The hybridization of the local search technique enhances the exploitation capability in the search process of the global optimization strategies. In this chapter, an effective local search (LS) technique that is designed by taking inspiration by limaçon curve, is incorporated in ABCA and the designed strategy is named Limaçon inspired ABC (LABC) algorithm. The exploitation capability of the LABC strategy is tested over 18 complex benchmark optimization problems. The test results are compared with similar state-of-art algorithms and statistical analysis shows the LABC can be considered an effective variant of the ABC algorithms to solve the complex optimization problems.

This chapter provides detailed descriptions of LABC and its applications in engineering optimization. Section 3.3 presents the proposed Limaçon inspired ABC. Further, the Experimental Setup and Computational outcomes are described in section 3.4. Finally, the chapter is concluded in section 3.5.

3.1 Introduction

The swarm intelligence (SI) derived techniques are impressive methods to deal with complex optimization problems. The SI based strategies do rely upon the social intellectual conduct of natural species. Artificial bee colony algorithm (ABCA) is a prominent SI technique, developed by taking inspiration from the intelligent communication of honey bees [79, 7]. In the past, the ABCA has been applied to many real-world complex optimization problems but it also suffers the common problems of SI based optimization techniques like stopping to move at the global optima and skipping the true solution due to high explorative nature of the solution search process[8, 80, 81, 17, 34].

The local search (LS) hybridization with the global search optimization algorithms boosts the exploitation capability of the global search algorithms which further decreases the chance of skipping the true solution. So, to enhance the local searchability in the ABCA solution search process, in this article a limaçon arc inspired LS (LLST) hybridized with ABCA.

The hybridized algorithm is titled as Limaçon inspired artificial bee colony (LABC) algorithm. The solution searchability of the proposed LABC is evaluated through numerous experimentations in form of accurateness, reliability, and consistency.

3.2 Artificial bee colony algorithm

Artificial bee colony algorithm (ABCA) is a significant strategy in the field of SI centered strategies. ABCA was proposed by D. Karaboga in the year 2005 [4]. It is motivated by the food foraging activities of the honey bees. There are three types of honey bees in the colony of bees that are employed honey bees, onlooker honey bees, and scout honey bees. At the initial stage, employed honey bees go for searching the food sources. They collect the nectar with all the associated information and return to the hive. They transfer knowledge associated with the food sources with the onlooker honey bees staying at the hive. Scout honey bees search the food sources randomly depending upon the internal motivation. Like other meta heuristic approaches, ABCA is also an iterative process that consists of following cycles of four stages:

3.2.1 Initialization stage:

The initialization of all the N solutions take place during this stage in the D dimensional space as per the lower bound and upper bound of all the decision parameters of the optimization problem. The initialization for w_i (i^{th} candidate solution where $i = 1, \dots, N$) is as per the following equation 3.1.

$$w_{ij} = w_{lowj} + rand[0, 1](w_{upperj} - w_{lowj}) \quad (3.1)$$

where, w_{lowj} and w_{upperj} respectively represent bounds of w_i in j^{th} direction further, $rand[0, 1]$ is an evenly scattered arbitrary number in the bound 0 to 1.

3.2.2 Employed honey bee stage:

During the employed honey bee stage, each solution of the search space is updated as per the equation 3.2. The solution is modified based upon the information obtained from any arbitrary solution of the search region. The fitness value of the newly produced solution (nectar amount) is calculated [13]. If the fitness value of the newly produced solution is greater than that of the previous solution, the new solution is selected for the next generation and the old one is discarded. Equation 3.2 represent position update equation for the i^{th} candidate solution.

$$w_{ij} = w_{ij} + \phi_{ij}(w_{ij} - w_{neighj}) \quad (3.2)$$

Where, $neigh \in \{1, 2, \dots, N\}$ and $j \in \{1, 2, \dots, D\}$ are arbitrarily selected indices, $neigh$ must be distinct from i , and ϕ_{ij} is an arbitrary number between $[-1, 1]$.

3.2.3 Onlooker honey bee stage:

When all the employed honey bees complete their task, they share all the information regarding the nectar with the onlooker honey bees. Onlooker honey bees analyze the information received from employed honey bees and select a food source based on a probability value

$Prob_i$. The probability $Prob_i$ is a fitness function that may be calculated using the equation 3.3.

$$Prob_i = \frac{Fitness_i}{\sum_{i=1}^N Fitness_i} \quad (3.3)$$

where $Fitness_i$ denotes the fitness value associated with the i^{th} solution. A solution's location is updated based on its probability value, just as it was in the previous phase. The freshly created solution's fitness value is calculated. The answer for the following generation is chosen using a greedy selection method. For the following generation, the best-fitting option is chosen.

3.2.4 Scout honey bee stage:

When a food supply does not change its location up to a certain point, the Scout honey bee stage is triggered. In this scenario, the abandoned food supply and the bee that is linked with it are referred to as scout honey bees. In the search space, that food source has been re-initialized. If w_i and $j \in \{1, 2, \dots, D\}$ are the rejected food sources, the food source is re-initialized according to the equation 3.4:

$$w_{ij} = w_{lowj} + rand[0, 1](w_{upperj} - w_{lowj}) \quad (3.4)$$

3.3 Limaçon inspired ABCA

According to the literature, the ABCA has issues with early convergence, stagnation, and occasionally being unable to find the real solution to an optimization problem citezhu2010gbest. LS methods, control settings, and hybridization with other search strategies have all been used in the past to improve the solution searchability of ABCA [82, 5].

Recently, a Limaçoninspired local search (LLST) technique developed from the Limaçonarc equation was published in the literature ([83]). A Limaçon is shown in LLST as a roulette traced by the locus of a point on the perimeter of a circle rolling around the periphery of another circle of equal radius. It's also known as the roulette effect, which occurs when a circle spins around another circle with half its radius, resulting in the little circle being trapped within the larger circle. The natural species limaçon, more often known as a snail, inspired this mathematical curve. The limaçonvertical and horizontal axis contour formulae are shown in equations 3.5and 3.6, respectively, [84].

$$z = p \pm q \sin \phi \quad (3.5)$$

$$z = p \pm q \cos \phi \quad (3.6)$$

The limaçon's distance from the beginning point is marked by z , two constants are denoted by p and q , and the angle of revolution is denoted by phi . The value of q determines the curve's transient phases; $q = 0, 1, and > p$, respectively, represent a circular, cardioid, and noose curve.

The value of q is the foundation for the curve's transient phases. For $q = 0$ to $q = 1$, a circle emerges, and for $q > p$, a noose on a curve appears.

The position update equation is formed using equation 3.5 with minor modifications in the suggested approach. As seen in equation 3.7, the group's best solution has an opportunity

to update position during LS.

$$w_{new} = w_{best} \pm (w_{best} - w_{neigh}) \times \sin(\phi) \quad (3.7)$$

Throughout the local searchability of the suggested method, w_{best} and w_{new} are the location of best solution and modified location of best position, respectively. The revolution angle is represented by the word phi . The community influence component is represented by the expression $(w_{best} - w_{neigh})$. The value of phi is calculated as follows:

$$\phi = \frac{\pi}{2} \times \left(1 - \frac{C_g}{T_g}\right) \quad (3.8)$$

C_g and T_g are the current and total LS generations counts, respectively. The value of T_g is determined through a thorough investigation, as shown in the section 3.4. Algorithm 3.1 illustrates the LLST approach.

Algorithm 3.1 Limaçon Local Search Technique(LLST):

w_{best} is the swarm's best solution, and $Min f(x)$ is the input function that has to be optimised.
Set the counter for local search generation $C_g = 0$ and the total number of local search generations T_g ;
while $C_g < T_g$ **do**
 Using the equation 3.8, find the value of phi .
 Produce two new solutions w_{new1} and w_{new2} through Algorithm 3.2.
 Determine the $f(w_{new1})$ and $f(w_{new2})$ objective values.
 if $f(w_{new1}) < f(w_{best})$ **then**
 $w_{best} = w_{new1}$;
 else if $f(w_{new2}) < f(w_{best})$ **then**
 $w_{best} = w_{new2}$;
 end if
end while
Return w_{best} .

As stated in Algorithm 3.1, the step size decreases as phi increases. In either a positive or negative direction, the value of phi ranges from 90° to 0° .

The lower step size is shown by the less angular step. This means that the neighbouring search space of the best solution is used during the local search operation.

Algorithm 3.2 New particle generation:

Input $Sign$ and the swarm's best particle w_{best} .
Choose a solution w_{neigh} at random from the population that is $best \neq k$.
for $j = 1$ to D **do**
 if $R(0, 1) < c_r$ **then**
 $w_{newj} = w_{bestj}$;
 else
 $w_{newj} = w_{bestj} + Sign(w_{bestj} - w_{neighj}) \times \sin\phi$;
 end if
end for
Return w_{new}

In Algorithm 3.2 c_r is the perturbation rate that controls the significance of disturbance in the best particle. $R(0, 1)$ is a random integer with a uniform distribution in the range 0 to 1.

In ABCA, the introduced state is put after the scout bee state. The remaining phases

Algorithm 3.3 Limaçon ABC:

```
Parameter initialization;  
while Stopping criteria do  
  Step 1: Basic ABC Steps.  
  Step 4: LLST Phase using Algorithm 3.1.  
end while  
Output the best particle.
```

are the same as in the basic form of ABCA. The suggested strategy's primary contribution is to improve ABCA's exploitation capabilities. Algorithm 3.3 demonstrates the created LABC algorithm.

3.4 Results analysis

18 different benchmark numerical optimization functions (fn_1 to fn_{18}) are chosen to assess the execution of the expected LABC method, as shown in Table 3.1 [33].

Table 3.1: Benchmark Set D: Dimensions, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable, AE: Acceptable Error

Objective function	search Range	Optimum Value	D	AE	C
$f_{n_1}(x) = \sum_{i=1}^D x_i^2$	[-5.12, 5.12]	$f(\vec{0}) = 0$	30	1.0E-05	S, U
$f_{n_2}(x) = \sum_{i=1}^D i \cdot (x_i)^4$	[-5.12, 5.12]	$f(\vec{0}) = 0$	30	1.0E-05	S, M
$f_{n_3}(x) = 10D + \sum_{i=1}^D [x_i^2 - 10 \cos(2\pi x_i)]$	[-5.12, 5.12]	$f(\vec{0}) = 0$	30	1.0E-05	N, M
$f_{n_4}(x) = \sum_{i=1}^D x_i \sin x_i + 0.1x_i $	[-10, 10]	$f(\vec{0}) = 0$	30	1.0E-05	S, M
$f_{n_5}(x) = \sum_{i=1}^D x_i^2 - 0.1(\sum_{i=1}^D \cos 5\pi x_i) + 0.1D$	[-1, 1]	$f(\vec{0}) = -D \times 0.1$	30	1.0E-05	S, M
$f_{n_6}(x) = \sum_{i=1}^D x_i^2 + (\sum_{i=1}^D \frac{i x_i}{2})^2 + (\sum_{i=1}^D \frac{i x_i}{2})^4$	[-5.12, 5.12]	$f(\vec{0}) = 0$	30	1.0E-02	N, M
$f_{n_7}(x) = \sum_{i=1}^D (x_i^2)^{\frac{2^{(i+1)z} + 1}{2^{i+1}}} + x_{i+1}$	[-1, 4]	$f(\vec{0}) = 0$	30	1.0E-05	U, N
$f_{n_8}(x) = 1 - \cos(2\pi \sqrt{\sum_{i=1}^D x_i^2}) + 0.1(\sqrt{\sum_{i=1}^D x_i^2})$	[-100, 100]	$f(\vec{0}) = 0$	30	1.0E-01	N, M
$f_{n_9}(x) = \sum_{i=1}^D x_i ^{\frac{1}{i+1}} \left(\exp\left(\frac{-(x_i^2 + x_{i+1}^2 + 0.5e^{x_i x_{i+1}})}{8}\right) \times \mathbf{1} \right)$	[-1, 1]	$f(\vec{0}) = 0$	30	1.0E-05	S, M
$f_{n_{10}}(x) = -\sum_{i=1}^D \left(\exp\left(\frac{-(x_i^2 + x_{i+1}^2 + 0.5e^{x_i x_{i+1}})}{8}\right) \times \mathbf{1} \right)$	[-5, 5]	$f(\vec{0}) = D + 1$	10	1.0E-05	N, M
$f_{n_{11}}(x) = \sum_{i=1}^D (x_i - i)^2$	[-30, 30]	$f(1, 2, 3, \dots, D) = 0$	30	1.0E-05	S, U
$f_{n_{12}}(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$	[-4.5, 4.5]	$f(3, 0, 5) = 0$	2	1.0E-05	N, M
$f_{n_{13}}(x) = p(x_2)(1 + p(x_1)) + (x_1 + 50p(x_2)(1 - 2p(x_1))) + (x_2 + 50(1 - 2p(x_2))) $	[-100, 100]	$f(0, 50) = 0$	2	1.0E-04	N, M
$f_{n_{14}}(x) = \sum_{i=1}^D \frac{z}{4000} - \prod_{i=1}^D \cos(\frac{z}{\sqrt{2}}) + 1 + f_{bias}$, $z = (x - o)$, $x = [x_1, x_2, \dots, x_D]$, $o = [o_1, o_2, \dots, o_D]$	[600, 600]	$f(o) = f_{bias} - 180$	10	1.0E-05	N, M
$f_{n_{15}}(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^D \cos(2\pi z_i)) + 20 + e + f_{bias}$, $z = (x - o)$, $x = (x_1, x_2, \dots, x_D)$, $o = (o_1, o_2, \dots, o_D)$	[-32, 32]	$f(o) = f_{bias} - 140$	10	1.0E-05	S, M
$f_{n_{16}}(x) = 10^9 x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^{-9}(x_1^2 + x_2^2)^4$	[-20, 20]	$J(0, 15) = J(0, -15) = 24777$	2	5.0E-01	N, M
$f_{n_{17}}(x) = \sum_{i=1}^9 (\frac{x_1 x_3 x_4}{1 + x_1 x_i + x_2 x_i} - y_i)^2$	[-10, 10]	$f(3, 13, 15, 16, 0, 78) = 0.4E-04$	3	1.0E-03	U, N
$f_{n_{18}}(x) = -[A \prod_{i=1}^D \sin(x_i - z) + \prod_{i=1}^D \sin(B(x_i - z))]$, $A = 2.5$, $B = 5$, $z = 30$	[0, 180]	$f(90 \mp z) = (A + 1)$	10	1.0E-02	N, M

To certify the competitiveness of the introduced LABC approach, a comparative analysis is performed amid LABC, ABCA, and 4 of its significant revised versions, namely best so far ABC (BSFABC) [85], black hole ABC (BHABC) [86], Gbest guided ABC (GABC) [9], levy flight ABC (LFABC) [35], and LSMO [83]. The experimental setting is listed below:

- Simulations/run =100.
- Colony size $NP = 50$ and Number of food sources $N = NP/2$.
- limit= $D \times N$ [13].
- Parameter setting for all the considered algorithms are kept same as mentioned in their native research article.
- The value of T_g (Total number of local search generation) is decided by experimentation. The results in the form of sum of success rate are analyzed for tested functions on distinct values of T_g . $T_g = 18$ provides better outcomes (in terms of value of sum of success) as depicted in Figure 3.1.

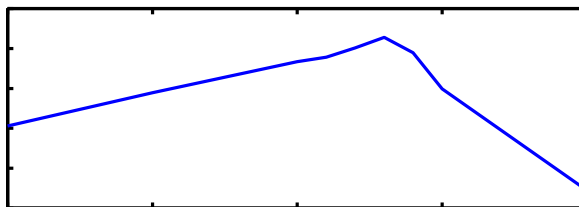


Figure 3.1: Sum of success versus (T_g)

The obtained experimental outcomes of all the considered approaches are portrayed in Table 3.2. The assessment is experimented in terms of four parameters namely, standard deviation (SD), mean error (ME), average number of function evaluations (AFE), and success rate (SR) as depicted in Table 3.2. The Table 3.2 demonstrates that the revised LABC primarily outplays to all the compared significant strategies.

Table 3.2: Result assessment of functions with LABC and other state-of-art algorithms

Test Problem	Measure	LABC	ABC	BHABC	GABC	BSFABC	LFABC	LSMO
f^{n_1}	SD	3.11E-06	1.56E-06	1.44E-06	1.81E-06	2.15E-06	1.73E-06	5.90E-07
	ME	3.92E-06	8.48E-06	8.53E-06	8.11E-06	7.49E-06	8.39E-06	9.23E-06
	AFE	12642.32	13963.77	22304.92	14347.5	30063	16733.85	18101.5
	SR	100	100	100	100	100	100	100
f^{n_2}	SD	2.43E-06	2.92E-06	2.63E-06	2.72E-06	3.12E-06	3.02E-06	8.42E-07
	ME	1.56E-06	5.46E-06	5.75E-06	5.51E-06	5.31E-06	6.62E-06	8.93E-06
	AFE	481.61	5629.43	8687.05	8388	24524.5	9556.12	2596.5
	SR	100	100	100	100	100	100	100
f^{n_3}	SD	2.79E-06	2.34E-06	2.88E-06	2.75E-06	3.11E-06	2.41E-06	1.40E+01
	ME	3.06E-06	7.58E-06	5.79E-06	6.38E-06	4.05E-06	7.18E-06	3.87E+01
	AFE	848.38	39982.67	44384.12	34805	122759.5	40644.63	2015
	SR	100	100	100	100	100	100	100
f^{n_4}	SD	2.67E-06	2.66E-06	1.60E-06	1.85E-06	6.05E-06	1.03E-05	3.03E-07
	ME	4.59E-06	7.82E-06	8.46E-06	8.32E-06	8.03E-06	9.06E-06	9.53E-06
	AFE	1030.21	75594.46	59016.04	54665.5	142277	85238.42	3046.5
	SR	100	100	100	100	96	98	100

to be cont'd on next page

Table 3.2: Result assessment of functions with LABC and other state-of-art algorithms (Cont.)

Test Problem	Measure	LABC	ABCA	BHABC	GABC	BSFABC	LFABC	LSMO
fn_5	SD	2.56E-06	2.02E-06	2.39E-06	1.91E-06	2.43E-06	2.22E-06	5.68E-02
	ME	2.79E-06	8.33E-06	7.72E-06	7.83E-06	6.97E-06	7.84E-06	2.22E-02
	AFE	11638.34	13632.1	35006.99	15420.5	32039	17862.88	13043.5
	SR	100	100	100	100	100	100	100
fn_6	SD	2.63E-03	1.61E+01	1.84E+01	1.58E+01	1.22E+01	1.57E+01	1.80E-02
	ME	5.43E-03	6.11E+01	1.01E+02	9.76E+01	8.38E+01	1.13E+02	2.20E-02
	AFE	182821.42	200025.72	200000.31	200000.01	200000	200040	186434
	SR	34	0	0	0	0	0	34
fn_7	SD	2.96E-06	1.60E-06	2.06E-06	1.97E-06	1.99E-06	1.58E-06	6.06E-07
	ME	3.42E-06	8.36E-06	8.04E-06	7.86E-06	7.73E-06	8.55E-06	9.14E-06
	AFE	11634.36	14830.27	23739.99	14076	31207.5	16111.3	15048.5
	SR	100	100	100	100	100	100	100
fn_8	SD	2.31E-01	8.66E-02	4.45E-02	3.35E-02	6.82E-02	4.45E-02	5.53E-02
	ME	6.86E-01	9.75E-01	9.33E-01	9.33E-01	9.56E-01	9.39E-01	2.88E-01
	AFE	93346.93	139235.63	94411.64	85618.12	186319.67	101452.43	91050
	SR	99	57	97	95	73	87	93
fn_9	SD	3.14E-06	2.83E-06	2.40E-06	2.60E-06	2.72E-06	3.13E-06	1.36E-06
	ME	2.85E-06	4.90E-06	6.55E-06	6.12E-06	5.84E-06	5.86E-06	8.38E-06
	AFE	11359.31	19776.19	7104.52	9392.5	14434	7523.66	9897
	SR	100	100	100	100	100	100	100
fn_{10}	SD	2.73E-06	7.33E-02	2.33E-06	2.39E-06	1.99E-01	4.27E-05	6.06E-01
	ME	2.94E-06	1.06E-02	7.20E-06	6.83E-06	6.09E-02	1.27E-05	1.40E+00
	AFE	10659.72	114061.2	70078.06	47688.66	123141.06	42442.21	19670
	SR	100	84	100	100	85	99	100
fn_{11}	SD	1.83E-06	2.38E-06	2.33E-06	1.81E-06	2.63E-06	1.97E-06	5.36E-07
	ME	7.84E-06	7.44E-06	7.71E-06	7.93E-06	7.11E-06	8.07E-06	9.23E-06
	AFE	32466.2	23156.54	23504.76	16625.5	40983.5	18653.59	34306
	SR	100	100	100	100	100	100	100
fn_{12}	SD	2.98E-06	2.73E-06	2.95E-06	2.93E-06	1.69E-05	2.84E-06	2.79E-06
	ME	5.35E-06	7.24E-06	5.49E-06	5.33E-06	1.28E-05	7.52E-06	4.86E-06
	AFE	14849.56	34002.38	7259.59	8701.35	49064.36	3746.11	2753.5
	SR	100	100	100	100	92	100	100
fn_{13}	SD	3.22E-05	2.73E-05	2.54E-05	2.29E-05	1.98E-04	2.37E-01	2.71E-01
	ME	4.58E-05	5.87E-05	6.29E-05	6.36E-05	8.56E-05	6.01E-02	8.01E-02
	AFE	7467.44	18836.4	11497.4	8726.06	6208.54	17885.73	9745.5
	SR	100	100	100	100	99	94	100
fn_{14}	SD	1.32E-03	3.02E-03	2.39E-03	7.35E-04	6.18E-03	7.35E-04	2.87E-02
	ME	2.72E-04	1.16E-03	8.61E-04	7.90E-05	4.58E-03	8.01E-05	4.05E-02
	AFE	67282.34	87111.25	91174.8	42366.85	118467.79	40382.88	117491
	SR	95	85	88	99	58	99	84
fn_{15}	SD	1.54E-06	1.85E-06	2.05E-06	1.28E-06	1.93E-06	1.34E-06	1.03E-06
	ME	8.31E-06	8.09E-06	7.68E-06	8.64E-06	8.13E-06	8.66E-06	8.83E-06
	AFE	15737.84	23391.88	71048.93	9321	31326.5	10934.63	24630
	SR	100	100	100	100	100	100	100
fn_{16}	SD	5.72E-03	5.52E-03	5.77E-03	5.37E-03	5.28E-03	5.68E-03	5.55E-03
	ME	4.98E-01	4.89E-01	4.91E-01	4.90E-01	4.91E-01	4.91E-01	4.92E-01
	AFE	885.19	3145.86	946.24	775	2800.72	687.8	5050
	SR	100	100	100	100	100	100	100
fn_{17}	SD	2.90E-06	2.97E-06	3.07E-06	2.95E-06	2.64E-06	3.10E-06	2.91E-06
	ME	2.15E-03	1.94E-03	1.95E-03	1.94E-03	1.95E-03	1.95E-03	1.95E-03
	AFE	21458.41	29064.93	4761.02	5094.92	17641.71	3418.07	3092
	SR	100	100	100	100	100	100	100
fn_{18}	SD	3.18E-03	1.75E-03	1.82E-03	2.42E-03	1.90E-03	1.67E-03	2.94E-01
	ME	5.22E-03	7.71E-03	7.78E-03	7.53E-03	7.91E-03	8.35E-03	4.39E-01
	AFE	35081.83	62307.88	42023.26	49473.76	63543.15	22030.31	71097.5
	SR	100	100	100	99	100	100	94

The boxplots investigation in form of AFE is also performed to indicate how the values in the data are spread out. The interquartile range and median of proposed variant is relatively small. This shows that proposed variant is fast enough and executes on smoother range in comparison to other consider significant algorithms.

Further, the convergence speed of the considered algorithms are also evaluated by measuring AFEs. A lesser AFEs reflect greater convergence speed. To eliminate the adverse

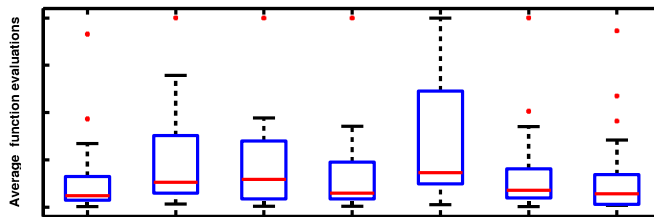


Figure 3.2: Boxplots graphs for average number of function evaluation

Table 3.3: Acceleration Rate (AR) of LABC as compare to other state of art algorithms

Test lems	Prob-	ABCA	BHABC	GABC	BSFABC	LFABC	LSMO
fn_1		1.105	1.764	1.135	2.378	1.324	1.432
fn_2		11.689	18.038	17.417	50.922	19.842	5.391
fn_3		47.128	52.316	41.025	144.699	47.909	2.375
fn_4		73.378	57.285	53.062	138.105	82.739	2.957
fn_5		1.171	3.008	1.325	2.753	1.535	1.121
fn_6		1.094	1.094	1.094	1.094	1.094	1.020
fn_7		1.275	2.041	1.210	2.682	1.385	1.293
fn_8		1.492	1.011	0.917	1.996	1.087	0.975
fn_9		1.741	0.625	0.827	1.271	0.662	0.871
fn_{10}		10.700	6.574	4.474	11.552	3.982	1.845
fn_{11}		0.713	0.724	0.512	1.262	0.575	1.057
fn_{12}		2.290	0.489	0.586	3.304	0.252	0.185
fn_{13}		2.522	1.540	1.169	0.831	2.395	1.305
fn_{14}		1.295	1.355	0.630	1.761	0.600	1.746
fn_{15}		1.486	4.515	0.592	1.991	0.695	1.565
fn_{16}		3.554	1.069	0.876	3.164	0.777	5.705
fn_{17}		1.354	0.222	0.237	0.822	0.159	0.144
fn_{18}		1.776	1.198	1.410	1.811	0.628	2.027

effect of stochastic nature of the strategies, the stated function evaluations for each test problem is averaged over 100 runs. For comparing convergence speeds, we use the acceleration rate (AR) which is computed as follows:

$$AR = \frac{AFE_{ALGO}}{AFE_{LABC}}, \quad (3.9)$$

where, $ALGO \in \{ABC, BHABC, GABC, BSFABC, LFABC, LSMO\}$ and $AR > 1$ means that the suggested LABC is speedier than other considered algorithms. To investigate the impact of AR, the outcomes of Table 3.2 are evaluated and the value of AR is calculated using equation (3.9). Table 3.3 shows a clear comparison between LABC and other considered algorithms in terms of AR. Table 3.3 clearly shows that LABC is speedy as compare to all the considered algorithms.

The algorithms are additionally verified in form of Mann-Whitney U rank sum test [5] that is applied on AFEs. The examination is performed at 5% noteworthiness level ($\alpha = 0.05$) and the upshots for 100 simulations are noted in Table 3.4. In Table 3.4 *higher* validate that our recommended LABC is better with respect to the other considered algorithm while *lower* shows that the compared strategy is superior. The statistical analysis show that the proposed variant performs well irrespective of the characteristics of the considered benchmark functions.

Table 3.4: Mann-Whitney U rank sum test on AFEs of LABC and compared strategies, TP: Test Problem.

TP	LABC Vs ABCA	LABC Vs BHABC	LABC Vs GABC	LABC Vs BS-FABC	LABC Vs LFABC	LABC Vs LSMO
fn_1	higher	higher	higher	higher	higher	higher
fn_2	higher	higher	higher	higher	higher	higher
fn_3	higher	higher	higher	higher	higher	higher
fn_4	higher	higher	higher	higher	higher	higher
fn_5	higher	higher	higher	higher	higher	higher
fn_6	higher	higher	higher	higher	higher	higher
fn_7	higher	higher	higher	higher	higher	higher
fn_8	higher	higher	lower	higher	higher	lower
fn_9	higher	lower	lower	higher	lower	lower
fn_{10}	higher	lower	higher	higher	higher	higher
fn_{11}	lower	lower	lower	higher	lower	higher
fn_{12}	higher	lower	lower	higher	lower	lower
fn_{13}	higher	higher	higher	lower	higher	higher
fn_{14}	higher	higher	lower	higher	lower	higher
fn_{15}	higher	higher	lower	higher	lower	higher
fn_{16}	higher	higher	lower	higher	lower	higher
fn_{17}	higher	lower	lower	lower	lower	lower
fn_{18}	higher	higher	higher	higher	lower	higher
Total no. of + signs	17	13	10	16	10	14

3.5 Conclusion

Local search (LS) strategies always enhance the performance of the Swarm Intelligence (SI) based algorithms. In view of this, in this article, a Limaçon arc inspired local search (LLST) strategy is hybridized with the artificial bee colony algorithm (ABCA). The designed strategy is named Limaçon inspired ABC (LABC) algorithm. The solution searchability of the LABC is tested over various unimodal, multi-modal, separable, non-separable benchmark test functions, and the performance is evaluated while comparing the results with various state-of-art algorithms. The obtained outcomes are statistically analyzed through the Mann-Whitney U rank-sum test, Boxplots, acceleration rates which validates the competitiveness of the designed strategy.

CHAPTER 4

FULLY INFORMED ABC ALGORITHM

Chapter 4

Fully Informed ABC Algorithm

The Gbest-guided Artificial Bee Colony (GABC) algorithm is a latest swarm intelligence based approach to solve optimization problem. In GABC, the individuals update their respective positions by drawing inspiration from the global best solution available in the current swarm. The GABC is a popular variant of Artificial Bee Colony (ABC) algorithm and is proved to be an efficient algorithm in terms of convergence speed. But, in this strategy, each individual is simply influenced by the global best solution, which may lead to trap in local optima. Therefore, in this chapter, a new search strategy, namely “Fully Informed Learning” is incorporated in the onlooker bee phase of ABC algorithm. The developed algorithm is named as Fully Informed Artificial Bee Colony (FABC) algorithm. To validate the performance of FABC, it is tested on 20 well known benchmark optimization problems of different complexities. The results are compared with GABC and some more recent variants of ABC. The results are very promising and show that the proposed algorithm is a competitive algorithm in the field of swarm intelligence based algorithms.

This chapter provides detailed description of fully informed ABC. The proposed FABC discussed in section 4.2. The experimental setup and result section 4.3 followed by conclusion in section 4.4.

4.1 Introduction

Swarm Intelligence is one of the recent outcome of the research in the field of Nature inspired algorithms[87, 88, 89, 90]. Collaborative trial and error method is the main concept behind the Swarm Intelligence which enables the algorithmic procedure to find the solution. D.Karaboga [4] contributed the recent addition to this category known as Artificial bee colony (ABC) algorithm. The ABC algorithm mimics the foraging behavior of honey bees while searching food for them. ABC is a simple and population based optimization algorithm. Here the population consists of possible solutions in terms of food sources for honey bees whose fitness is regulated in terms of nectar amount which the food source contains. Artificial Bee Colony is made of three groups of bees: employed bees, onlooker bees and scout bees. The number of employed and onlooker bees is equal. The employed bees searches the food source in the environment and store the information like the quality and the distance of the food source from the hive. Onlooker bees wait in the hive for employed bees and after collecting information from them, they start searching in neighborhood of that food sources which are having better nectar. If any food source is abandoned then scout bee finds new food source randomly in search space. While searching the solution of

any optimization problem, ABC algorithm first initializes ABC parameters and swarm then it requires the repetitive iterations of the three phases namely employed bee phase, onlooker bee phase and scout bee phase.

However the ABC achieves a good solution at a significantly faster rate but, like the other optimization algorithms, it is also weak in refining the already explored search space. It is shown in literature that basic ABC itself has some drawbacks like stop proceeding toward the global optimum even though the population has not converged to a local optimum [8] and it is observed that the position update equation of ABC algorithm is good at exploration but poor at exploitation [9] i.e, has not a proper balance between exploration and exploitation. Therefore these drawbacks require a modification in position update equation in ABC. To enhance the exploitation, Wei-feng Gao et al. [91] improved position update equation of ABC such that the bee searches only in neighborhood of the previous iteration's best solution. Anan Banharnsakun et al. [85] proposed the best-so-far selection in ABC algorithm and incorporated three major changes: The best-so-far method, an adjustable search radius, and an objective-value-based comparison in ABC. To solve constrained optimization problems, D. Karaboga and B. Akay [92] used Debs rules consisting of three simple heuristic rules and a probabilistic selection scheme in ABC algorithm.

In 2010, Zhu and Kwong [9] proposed an improved ABC algorithm, namely Gbest-guided ABC (GABC) algorithm by incorporating the information of global best (Gbest) solution into the solution search equation to improve the exploitation. But as all the individuals drawing inspiration from the global best solution, there is a enough chance of swarm stagnation. Therefore, in this chapter, a new position update strategy, namely "Fully Informed Learning" [93] is incorporated in the onlooker phase of GABC algorithm. The proposed algorithm is named is Fully Informed ABC (FABC). The FABC algorithm is tested on 20 benchmark problems and the results are very encouraging.

4.2 Fully Informed ABC

This section explains the proposed modified ABC algorithm. In ABC, at any instance, a solution is updated through information flow from other solutions of the swarm. This position updating process uses a linear combination of current position of the potential solution which is going to be updated and position of a randomly selected solution as step size with a random coefficient $\phi_{ij} \in [-1, 1]$. This process plays an important role to decide the quality of new solution. If the current solution is far from randomly selected solution and absolute value of ϕ_{ij} is also high then the change will be large enough to jump the true optima. On the other hand, small change will decrease the convergence rate of whole ABC process. Further, It is also suggested in literatures [8, 9] that basic ABC itself has some drawbacks, like stop proceeding toward the global optimum even though the population has not converged to a local optimum and it is observed that the position update equation of ABC algorithm is good at exploration but poor at exploitation. Therefore, to improve the exploitation capability of ABC algorithm, in 2010, Zhu and Kwong [9] proposed an improved ABC algorithm called Gbest-guided ABC (GABC) algorithm by incorporating the information of global best (Gbest) solution into the solution search equation to improve the exploitation. GABC is inspired by PSO [88], which, in order to improve the exploitation, takes advantage of the information of the global best (gbest) solution to guide the search by candidate solutions.

They made the following changes to ABC's solution search equation:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) + \psi_{ij}(y_j - x_{ij}) \quad (4.1)$$

where y_j is the j^{th} element of the global best solution, ψ_{ij} is a uniform random number in $[0, C]$, and C is a non negative constant, and the third term in the right-hand side of equation (4.1) is a new added term called gbest term, and y_j is the j^{th} element of the global best solution. The gbest term can drive the new candidate solution towards the global best solution, according to equation (4.1), therefore the modified solution search equation represented by (4.1) can enhance the exploitation of ABC method. It's worth noting that the C parameter in (4.1) is crucial for balancing the exploration and exploitation of the candidate solution search.

The previous description and equation 4.1 show that the global best solution discovered in the present population influences every individual participating in the search process. This method forces people to follow the best solution discovered in the swarm, therefore improving the algorithm's exploitation capacity. The downside of this technique is the likelihood of swarm stagnation and early convergence.

Individuals change their locations depending on the probability, which is a function of fitness (see equation 2.3) in the GABC onlooker bee phase. Furthermore, this is the period in which the swarm may converge around the optimal solution, increasing the likelihood of premature convergence. As a result, in the observer bee phase of the GABC algorithm, a novel search method called Fully Informed Learning (FIL) is included to limit such possibilities in the swarm.

To update its position in the search space, the individual collects input from the best solution in the current swarm as well as any other nearby solutions in FIL. FIL is mathematically described by the equation 4.2.

$$v_{ij} = x_{ij} + \phi_{ij} \frac{\sum_{k=1}^{SN} (x_{ij} - x_{kj})}{SN} + \psi_{ij}(y_j - x_{ij}) \quad (4.2)$$

The number of food sources is represented by SN , while the other symbols have their normal meanings, as stated in equation 4.1. The high fit solutions update their locations by drawing inspiration from the best solution in the current swarm as well as collecting input from all other people in the swarm, as shown by this equation. As a result, the current best answer cannot influence the path of the search process. The exploitation capability owing to the best solution identified so far can be balanced, as can the exploration capability due to learning phenomena from all the people in the swarm.

4.3 Results Analysis

This section illustrates how the suggested method performed in terms of accuracy, efficiency, and dependability.

4.3.1 Test problems under consideration

Twenty mathematical optimization problems (f_1 to f_{20}) of varying features and complexity are considered to validate the efficacy of the proposed algorithm FABC (mentioned in Table 4.1). All of these issues occur on a regular basis in nature.

Table 4.1: Test problems

Objective function	Search Range	Optimum Value	D	Acceptable Error
$f_1(x) = 20 + e + \exp(x) \frac{v^2}{D} \sqrt{\sum_{i=1}^D u_i^2}$ $-\exp(x) \frac{1}{D} \sum_{i=1}^D \cos(2u_i x_i)$	[-1, 1]	$f(\vec{0}) = 0$	30	1.0E-05
$f_2(x) = \sum_{i=1}^D x_i^2 \quad 0.1 \sum_{i=1}^D \cos(5\pi x_i) + 0.1 D$	[-1, 1]	$f(\vec{0}) = D \times 0.1$	30	1.0E-05
$f_3(x) = \sum_{i=1}^D x_i^2 + (\sum_{i=1}^D \frac{x_i^2}{2})^2 + (\sum_{i=1}^D \frac{x_i^2}{3})^3$	[-5, 12.5, 12]	$f(\vec{0}) = 0$	30	1.0E-02
$f_4(x) = \sum_{i=1}^D (x_i^{2^{(i+1)}} + x_i^{2^i} + 2^{x_i} + 1)$	[-1, 4]	$f(\vec{0}) = 0$	30	1.0E-05
$f_5(x) = \sum_{i=1}^D x_i + \mathbb{1}_{i=1}^D x_i $	[-10, 10]	$f(\vec{0}) = 0$	30	1.0E-05
$f_6(x) = \sum_{i=1}^D x_i ^{i+1}$	[-1, 1]	$f(\vec{0}) = 0$	30	1.0E-05
$f_7(x) = \sum_{i=1}^D i x_i^4 + \text{random}[0, 1]$	[-1, 28.1, 28]	$f(\vec{0}) = 0$	30	1.0E-05
$f_8(x) = -\sum_{i=1}^D \left(\exp\left(\frac{(x_i^2 + x_i + 1)^{1+0.3x_i x_i + 1}}{8}\right) \times 1 \right)$	[-5, 5]	$f(\vec{0}) = -D + 1$	10	1.0E-05
where, $\mathbb{1} = \cos\left(4\sqrt{\frac{x_i^2 + x_i + 1}{8}} + 0.5x_i x_i + 1\right)$ $f_9(x) = \sum_{i=1}^D (x_i - 1)^2 \sum_{j=2}^D x_i x_j$	[-1, D ² , D ²]	f_{min} $(\frac{D(D+4)}{6}, (D-1))$	10	1.0E-01
$f_{10}(x) = \frac{D}{D} (10 \sin^2(\pi g_1)) + \sum_{i=1}^D (g_i - 1)^2 \times (1 + 10 \sin^2(\pi g_{i+1})) + (g_D - 1)^2$, where $g_i = 1 + \frac{1}{4}(x_i + 1)$	[-10, 10]	$f(\vec{1}) = 0$	30	1.0E-05
$f_{11}(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 \times (1 + \sin^2(3\pi x_{i+1}))) + (x_D - 1)^2 (1 + \sin^2(2\pi x_D))$	[-5, 5]	$f(\vec{1}) = 0$	30	1.0E-05
$f_{12}(x) = 100 x_2 - x_1 ^2 + (1 - x_1)^2 + 90(x_4 - x_3)^2 + (1 - x_3)^2 + 10.1 (x_2 - 1)^2 + (x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1)$	[-10, 10]	$f(\vec{1}) = 0$	4	1.0E-05
$f_{13}(x) = \sum_{i=1}^D u_i \frac{x_1(0.2 + 0.3x_i^2)}{0.2 + 0.3x_i^2 + 4} ^2$	[-5, 5]	$f(0.192833,$ $0.190830, 0.123117,$ $0.135766) = 0.000307486$	4	1.0E-05
$f_{14}(x) = \sum_{i=1}^D (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{\text{bias}}$, $z = x - o + 1$, $x = [x_1, x_2, \dots, x_D]$, $o = [o_1, o_2, \dots, o_D]$	[-100, 100]	$f(o) = f_{\text{bias}} = 390$	10	1.0E-01
$f_{15}(x) = \sum_{i=1}^D z_i^2 + f_{\text{bias}}$, $z = x - o$, $x = [x_1, x_2, \dots, x_D]$, $o = [o_1, o_2, \dots, o_D]$	[-100, 100]	$f(o) = f_{\text{bias}} = 450$	10	1.0E-05
$f_{16}(x) = \sum_{i=1}^D \frac{x_i^2}{3000} \prod_{i=1}^D \cos(\frac{x_i}{2}) + 1 + f_{\text{bias}}$, $z = (x - o)$, $x = [x_1, x_2, \dots, x_D]$, $o = [o_1, o_2, \dots, o_D]$	[-600, 600]	$f(o) = f_{\text{bias}} = 180$	10	1.0E-05
$f_{17}(x) = \frac{20 \exp(0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2})}{\prod_{i=1}^D \cos(2\pi z_i)} + 20 + e + f_{\text{bias}}$, $z = (x - o)$, $x = (x_1, x_2, \dots, x_D)$, $o = (o_1, o_2, \dots, o_D)$	[-32, 32]	$f(o) = f_{\text{bias}} = 140$	10	1.0E-05

to be continued on next page

Table 4.1: Test problems (Cont.)

Objective function	Search Range	Optimum Value	D	Acceptable Error
$f_{18}(x) = (1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \cdot (30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	[-2, 2]	$f(0, 1) = 3$	2	1.0E-14
$f_{19}(x) = \sum_{k=1}^5 \left(\frac{x_1 x_3 x_4}{1 + x_1 x_2 + x_2 x_4} - y_k \right)^2$	[-10, 10]	$f(3.13, 1.9, 1.0, 0.78) = 0.4 \times 10^{-4}$ $f(x) = 0; x(t) = 5 \times t, t = 1, \dots, D$	3	1.0E-05
$f_{20}(x) = \sum_{i=1}^D 98 \times x_i^2$	[-5.12, 5.12]	$f(x) = 0; x(t) = 5 \times t, t = 1, \dots, D$	30	1.0E-15

Table 4.2: Comparison based on average function evaluations, TP: Test Problem.

TP	ABC	BSFABC	GABC	MABC	FABC
f_1	48726.5	72833	58057	43333	39355
f_2	22951.5	32029	28709	22763.5	20618
f_3	200000	200000	200000	200005.69	200000
f_4	20917	31416.5	25721	22992	19971
f_5	41646	52966	52650	32888.5	35643
f_6	16229	14299	16768	9389	7091
f_7	200038.73	200033.72	200038.69	200014.92	200043.25
f_8	82609.49	89613.57	65244.05	65025.88	56265.79
f_9	200004.52	200022.17	200002.78	194815.12	199579.45
f_{10}	19662	26586	24291	22549.5	18285
f_{11}	21746	28700	26762	20899.5	18952
f_{12}	200024.66	166272.97	178380.44	164408.83	46578.45
f_{13}	188361.32	144391.61	116965.09	187245.43	63687.6
f_{14}	173092.37	178745.99	155110.67	171899.54	139291.22
f_{15}	8997	18238.5	10385	8664	8370
f_{16}	93520.83	69837.25	42231.9	60493.49	73093.15
f_{17}	16673	31235	17798	14191.55	13376
f_{18}	103298.36	82223.49	115332.45	121008.11	108465.08
f_{19}	25134.26	18243.24	6521.4	8570.8	2903.48
f_{20}	62534.5	71096.5	75602	59732	54673

4.3.2 Parameter setting for experiments

The success rate, average number of function evaluations, and mean error derived from the proposed FABC are all recorded. For the sake of comparison, the results for these test issues (Table 4.1) were acquired from the basic ABC and newer versions of ABC entitled Best-So-Far ABC (BSFABC) [85], Gbest-guided ABC (GABC) [9], and Modified ABC (MABC) [94]. When implementing the proposed and other considered algorithms to solve the issues, the following parameter settings are used:

- Number of executions/run =100.
- Number of solutions $SN = NP/2$ [95, 96].
- Limit= $D \times SN$ [92, 94] and $C = 1.5$ [9].
- The algorithm stopping condition is the either 200000 number of function evaluations or the acceptable error as mentioned in Table 4.1 is achieved.
- The parameter values for the other algorithms studied, ABC, GABC, BSFABC, and MABC, are taken from their original papers.

4.3.3 Reported Results

The numerical results for the benchmark problems of Table 4.1 with the experimental settings provided in section 4.3.2 are shown in Tables 4.2, 4.3, and 4.4. In terms of average number of function evaluations (AFE), success rate (SR), and mean error, these tables demonstrate the results of the proposed and alternative methods evaluated (ME). Here, SR is the average number of function evaluations called by the algorithm in 100 runs to reach the termination condition, and AFE is the number of times the algorithm attained the function optima with acceptable error.

After examining the data, it can be concluded that FABC surpasses the other algorithms in terms of accuracy (thanks to ME), dependability (due to SR), and efficiency for the vast majority of the time (due to AFE). In order to evaluate the algorithms output more thoroughly, various statistical tests such as the Mann-Whitney U rank sum test, acceleration rate (AR) [97], boxplots, and performance indices [98] were performed.

Table 4.3: Comparison based on success rate out of 100 runs, TP: Test Problem.

TP	ABC	BSFABC	GABC	MABC	FABC
f_1	100	100	100	100	100
f_2	100	100	100	100	100
f_3	0	0	0	0	0
f_4	100	100	100	100	100
f_5	100	100	100	100	100
f_6	100	100	100	100	100
f_7	0	0	0	0	0
f_8	94	82	100	100	99
f_9	0	0	0	27	1
f_{10}	100	100	100	100	100
f_{11}	100	100	100	100	100
f_{12}	0	3	2	7	100
f_{13}	16	45	91	9	96
f_{14}	20	20	45	29	55
f_{15}	100	100	100	100	100
f_{16}	88	91	99	97	91
f_{17}	100	100	100	100	100
f_{18}	63	63	44	42	49
f_{19}	100	100	100	100	100
f_{20}	100	100	100	100	100

Table 4.4: Comparison based on mean error, TP: Test Problem.

TP	ABC	BSFABC	GABC	MABC	FABC
f_1	8.63E-06	7.71E-06	8.74E-06	9.51E-06	9.45E-06
f_2	7.47E-06	7.05E-06	8.09E-06	9.23E-06	8.99E-06
f_3	9.75E+01	8.50E+01	1.07E+02	1.47E-01	8.39E-01
f_4	7.84E-06	7.39E-06	8.11E-06	9.02E-06	9.07E-06
f_5	9.16E-06	8.99E-06	9.16E-06	9.49E-06	9.41E-06
f_6	5.16E-06	5.79E-06	6.47E-06	8.01E-06	7.55E-06
f_7	1.16E+01	1.00E+01	1.06E+01	9.85E+00	9.29E+00
f_8	3.67E-02	5.37E-02	6.86E-06	8.38E-06	2.64E-03
f_9	1.21E+00	4.39E+00	7.09E-01	5.96E-03	7.05E-01
f_{10}	6.89E-06	6.84E-06	8.32E-06	9.04E-06	9.11E-06
f_{11}	7.33E-06	7.22E-06	7.47E-06	9.18E-06	9.02E-06
f_{12}	1.39E-01	2.12E-02	2.61E-02	1.02E-02	7.83E-04
f_{13}	1.76E-04	1.46E-04	9.13E-05	2.07E-04	1.14E-04
f_{14}	1.13E+00	2.68E+00	6.77E-01	7.90E-01	1.22E+00
f_{15}	7.06E-06	6.66E-06	7.40E-06	8.21E-06	7.77E-06
f_{16}	1.09E-05	3.75E-06	5.62E-06	1.64E-05	7.45E-05
f_{17}	7.93E-06	8.07E-06	8.39E-06	8.71E-06	8.53E-06
f_{18}	9.00E-07	3.87E-14	5.61E-14	5.74E-14	5.19E-14
f_{19}	1.95E-03	1.95E-03	1.95E-03	1.95E-03	1.94E-03
f_{20}	9.20E-16	7.27E-16	9.31E-16	9.27E-16	9.24E-16

4.3.4 Statistical Analysis

On the basis of SR, AFE, and ME, the algorithms ABC, GABC, BSFABC, MABC, and FABC are compared. From the findings in Table 4.2, it is obvious that FABC is less expensive on 14 test functions ($f_1, f_2, f_4, f_6, f_8, f_{10}, f_{11} - f_{15}, f_{17}, f_{19}, f_{20}$). FABC effectively balances the exploration and exploitation capacities since these functions encompass uni-model, multimodel, separable, non separable, lower and higher dimension functions. Over the test function f_{16} , which is a multimodel non-separable test function, GABC, BSFABC, and MABC outperform FABC. On test functions f_{18} , which is likewise a multimodel non-separable test function, BSFABC outperforms FABC. It demonstrates that BSFABC is more effective at solving multimodel non-separable test problems. For f_9 test functions, MABC is less expensive than FABC. When examined independently, FABC outperforms ABC in 18 test functions, MABC and BSFABC in 16 test functions, and GABC in 17 test functions with mixed features. When the outputs of all functions are combined, the FABC method is the most cost-effective approach for the majority of functions.

Furthermore, Table 4.3 shows that FABC is more trustworthy than the examined algorithms for $f_{12} - f_{14}$ test issues when compared on the basis of success rate. Except for the test functions f_9, f_{16} , and f_{18} , the performance of the FABC method is comparable to that of the other algorithms studied. Tables 4.2 and 4.3 show that comparisons on the basis of AFE and SR are not possible for the test functions f_3 and f_7 . A fair comparison of these functions may be made by examining mean error (see Table 4.4). The MABC and FABC perform better than the ABC, GABC, and BSFABC in solving f_3 and f_7 test problems, as shown in Table 4.4. The boxplots for average number of function evaluations for all

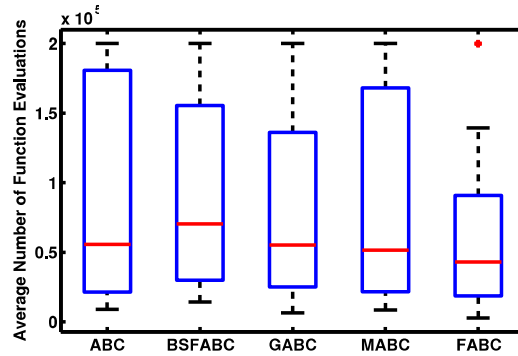


Figure 4.1: Average number of function evaluation through Boxplots graph

algorithms FABC, ABC, GABC, BSFABC, and MABC have been shown in Figure ??fig:bp feval because boxplot [99] can efficiently display the empirical distribution of findings. FABC is cost effective in terms of function evaluations, as shown by the interquartile range and median of average number of function evaluations in Figure 6.1.

Though it is obvious from box plots that FABC is more cost effective than ABC, BSFABC, GABC, and MABC, i.e., FABC's outcome varies from the others, we need to do another statistical test to see if there is a significant difference in algorithm output or if the difference is due to chance. The average number of function evaluations utilised by the investigated algorithms to solve the different problems is not normally distributed, as shown by the boxplots in Figure 6.1, therefore a non-parametric statistical test is necessary to compare the algorithms' performance. The non-parametric Mann-Whitney U rank sum

Table 4.5: The Mann-Whitney U rank sum test at a $alpha = 0.05$ significance level was used to do a comparison based on mean function evaluations ('+' means FABC is considerably better, '-' means FABC is much worse, and '=' means there is no significant difference.), TP: Test Problem.

TP	Mann-Whitney U rank sum test with FABC				TP	Mann-Whitney U rank sum test with FABC			
	ABC	BSFABC	GABC	MABC		ABC	BSFABC	GABC	MABC
f_1	+	+	+	+	f_{11}	+	+	+	+
f_2	+	+	+	+	f_{12}	+	+	+	+
f_3	=	=	=	=	f_{13}	+	+	+	+
f_4	+	+	+	+	f_{14}	+	+	+	+
f_5	+	+	+	-	f_{15}	+	+	+	+
f_6	+	+	+	+	f_{16}	+	-	-	-
f_7	=	=	=	=	f_{17}	+	+	+	+
f_8	+	+	+	+	f_{18}	+	-	+	+
f_9	+	+	+	-	f_{19}	+	+	+	+
f_{10}	+	+	+	+	f_{20}	+	+	+	+

[100] is a well-established test for comparing non-Gaussian data. This test is conducted between FABC - ABC, FABC - BSFABC, FABC - GABC, and FABC - MABC in this chapter at a 5% level of significance ($alpha = 0.05$).

The results of the Mann-Whitney U rank sum test for the average function evaluations of 100 simulations are shown in Table 4.5. First, we look for a significant difference using the Mann-Whitney U rank sum test, which determines whether or not the two data sets are substantially different. If there is no significant difference (i.e., the null hypothesis is accepted), the sign '=' appears; if there is a significant difference (i.e., the null hypothesis is rejected), compare the average number of function evaluations. We use the marks '+' and '-' to indicate whether FABC performs fewer or more average function evaluations than the other methods. As a result, in Table 4.5, a value of '+' indicates that FABC is considerably better, while a value of '-' indicates that FABC is significantly worse. Out of 80 comparisons, Table 4.5 has 67 '+' indications. As a consequence, FABC outcomes are substantially more cost efficient than ABC, BSFABC, GABC, and MABC across the test issues evaluated.

Performance indices (PIs) are also generated to compare the investigated algorithms by providing weighted importance to SR, AFE, and ME [98]. The following equations are used to compute the PI values for the FABC, ABC, BSFABC, GABC, and MABC:

$$PI = \frac{1}{N_p} \sum_{i=1}^{N_f} (k_1 \alpha_1^i + k_2 \alpha_2^i + k_3 \alpha_3^i)$$

$$\text{Where } \alpha_1^i = \frac{Sr^i}{Tr^i}; \alpha_2^i = \begin{cases} \frac{Mf^i}{Af^i}, & \text{if } Sr^i > 0. \\ 0, & \text{if } Sr^i = 0. \end{cases}; \text{ and } \alpha_3^i = \frac{Mo^i}{Ao^i}$$

$i = 1, 2, \dots, N_p$

- Sr^i shows that number of successful execution of i^{th} problem.
- Tr^i shows total number of execution of i^{th} problem.
- Mf^i shows the minimum of AFE which are used to get optimum result of i^{th} problem.
- Af^i shows AFE required to get optimum result of i^{th} problem.
- Mo^i shows least error get for the i^{th} problem.
- Ao^i shows mean error get by the applied approach for the i^{th} problem.

- N_p shows the number of benchmark functions (test problems) considered for experiments.

The weights allocated to SR, AFE, and ME are k_1, k_2 , and k_3 , respectively, where $k_1 + k_2 + k_3 = 1$ and $0 \leq k_1, k_2, k_3 \leq 1$. To compute the PI s, two variables are given equal weights, while the remaining variable's weight changes from 0 to 1, as described in [101]. The following are the outcomes:

1. $k_1 = W, k_2 = k_3 = \frac{1-W}{2}, 0 \leq W \leq 1$;
2. $k_2 = W, k_1 = k_3 = \frac{1-W}{2}, 0 \leq W \leq 1$;
3. $k_3 = W, k_1 = k_2 = \frac{1-W}{2}, 0 \leq W \leq 1$

Figures 4.2(a), 4.2(b), and 4.2(c) illustrate the graphs corresponding to each of the instances (1), (2), and (3) for the considered algorithms, respectively. The horizontal axis in these images represents the weights k_1, k_2 , and k_3 , whereas the vertical axis represents the PI .

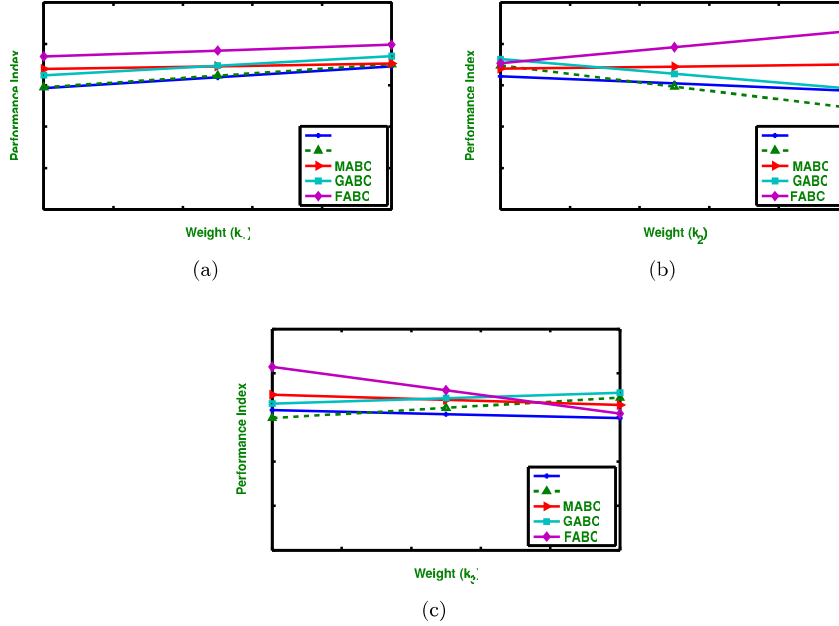


Figure 4.2: Performance index (PI) for considered benchmark functions; (a) for case (1), (b) for case (2) and (c) for case (3).

AFE and ME are equally weighted in instance (1). Figure 4.2(a) compares the performance of the different algorithms by superimposing their PI s. FABC's PI is found to be greater than the algorithms under consideration. In cases 2 and 3, SR and ME are given equal weights, while SR and AFE are given equal weights. Figures 4.2(b) and (c) show that the algorithms work in the same way in both cases (1).

We also compare the convergence speed of the algorithms under consideration by measuring the AFEs. A faster convergence speed is associated with lower AFEs. The provided function evaluations for each test problem are averaged across 100 runs to reduce the influence of the algorithms' stochastic character. We use the acceleration rate (AR) to

Table 4.6: Acceleration Rate (AR) of FABC as compared to the ABC, BSFABC, GABC, and MABC, TP: Test Problems

TP	ABC	BSFABC	GABC	MABC
f_1	1.238127303	1.850667005	1.475212807	1.101079914
f_2	1.113177806	1.553448443	1.392424095	1.10405956
f_3	1	1	1	1.00002845
f_4	1.047368685	1.573106004	1.28791748	1.151269341
f_5	1.168420167	1.486014084	1.477148388	0.922719749
f_6	2.288675786	2.016499788	2.364687632	1.324072768
f_7	0.999977405	0.99995236	0.999977205	0.999858381
f_8	1.468201015	1.59268305	1.159568718	1.155691229
f_9	1.002129828	1.002218264	1.00212111	0.976128153
f_{10}	1.075307629	1.453978671	1.328465956	1.233223954
f_{11}	1.147425074	1.514352047	1.41209371	1.102759603
f_{12}	4.29436059	3.569740298	3.829677458	3.529718786
f_{13}	2.957582324	2.267185606	1.836544162	2.940061017
f_{14}	1.242665331	1.283253819	1.113571049	1.234101762
f_{15}	1.074910394	2.179032258	1.240740741	1.035125448
f_{16}	1.279474616	0.95545547	0.577781913	0.827621877
f_{17}	1.246486244	2.335152512	1.330592105	1.060971142
f_{18}	0.95236513	0.758064162	1.06331411	1.115641181
f_{19}	8.656598289	6.283232535	2.246063345	2.951905989
f_{20}	1.143791268	1.300395076	1.382803212	1.092531963

compare convergence speeds, which is defined as follows, based on the AFEs for the two algorithms ALGO and FABC:

$$AR = \frac{AFE_{ALGO}}{AFE_{FABC}}, \quad (4.3)$$

$ALGO \in \{ABC, BSFABC, GABC, \text{ and } MABC\}$, with $AR > 1$ indicating FABC is the fastest. The findings of Table 4.2 are evaluated, and the value of AR is computed using equation (4.3) in order to study the suggested algorithm's AR in comparison to the considered methods. FABC and ABC, FABC and BSFABC, FABC and GABC, and FABC and MABC are all compared in terms of AR in Table 4.6. It is clear from the Table 4.6 that convergence speed of FABC is better than considered algorithms for most of the functions.

4.4 Conclusion

Gbest-guided Artificial Bee Colony (GABC) Algorithm has been proved to be a competitive optimization problem solver in the field of swarm intelligence based algorithm. GABC algorithm is developed to improve the convergence speed of the ABC algorithm and it has been proved its efficiency with drawbacks like premature convergence and stagnation. To reduce such possibilities in the GABC, a new solution search strategy, namely Fully Informed Learning (FIL) is incorporated with onlooker bee phase of GABC. In FIL strategy, individual updates its position through learning from the best solution in the swarm as well as gathering information from all the other individuals of the swarm. The proposed algorithm has been extensively compared with other recent variants of ABC namely, BSFABC, GABC, and MABC. Through the extensive experiments, it can be stated that the proposed algorithm is a competitive algorithm to solve the continuous optimization problems.

CHAPTER 5

FABC ALGORITHM FOR LARGE SCALE JOB SHOP SCHEDULING

Chapter 5

FABC Algorithm for Large Scale Job Shop Scheduling

The large-scale job-shop scheduling problem (LSJSSP) is among one of the complex scheduling problems. In past, although the swarm intelligence-based algorithms (SIA) have been efficiently applied to solve the LSJSSP, finding the best solution for LSJSSP instances remains a challenging task. Therefore, in this chapter, a novel SIA is applied to solve the 105 LSJSSP instances. The selected SIA is Fully Informed Artificial Bee Colony (FABC) algorithm. The FABC algorithm is developed by taking inspiration from the GABC algorithm position update process. In the FABC, the onlooker bee process of the Artificial Bee Colony (ABC) algorithm is modified and designed such that the new position of the solution search agent is obtained while learning from all the nearby agents. The results obtained by the FABC is compared with the state-of-art algorithms. The results analysis shows that the proposed approach to solving LSJSSP is competitive in the field of SIA.

This chapter provides detailed description of fully informed ABC. The proposed FABC discussed in section 5.2 with its application for job shop scheduling problem in section 5.3. The experimental setup and result section 5.5 followed by conclusion in section 5.6.

5.1 Introduction

Efficient scheduling is crucial for making the best use of available resources. In the domain of production management, the Large Scale Job-shop Scheduling Problem (LSJSSP) is a complicated combinatorial optimization problem. JSSP needs n jobs to be accomplished on m systems (machines). The system order for all jobs is fixed and varies depending on the jobs. The jobs are put in place in a non-preemptive manner, which means that while one job is running on one system, it cannot be disrupted by another. The primary goal of JSSP is to find an appropriate sequence scheme that reduces the time it takes for all jobs to be completed, which is referred to as makespan (MS). The goal is to minimize the makespan (MS) [102, 103].

The LSJSSP is one of the most important NP-hard problem. To solve LSJSSP, several deterministic conventional mathematical models and heuristic methods have been used. To small size LSJSSP cases, mathematical models have a successful solution in a reasonable amount of time. [104]. The computational time increases exponentially as the size of the instances grows. So, for a larger scale LSJSSP, Non-conventional nature inspired algorithms

(NIAs) are preferred alternatives [105]. The numerous processes found in nature are used to create NIAs. Swarm intelligence based algorithms (SIA) and evolutionary algorithms (EAs) are the two main types of NIAs. The design of SIA was influenced by the intellectual actions of creatures. Some state-of-art SIA are Artificial bee colony (ABC)[4], spider monkey optimization (SMO) [12], teaching learning based optimization (TLBO) [11] etc.,. EAs like differential evolution (DE) [106], genetic algorithm (GA) [107] etc., are based on biotic transformation like crossover, selection etc.

In recent years, NIAs are performing very well to solve physical world problems [108, 5]. In this series, many NIAs emerged well to solve LSJSSP such as genetic algorithm (GA) [109], particle swarm optimization (PSO) [110], hybrid biogeography based optimization (BBO) algorithm [111], hybrid differential evolution algorithm [112], multiple type individual enhancement PSO (MPSO) algorithm [113], classical LSJSSP [114], differential based harmony search algorithm with variable neighborhood search [115], biased random key genetic algorithm [116], new neighboured structure based algorithm [105], teaching learning based optimization (TLBO) algorithm [117], improved ABC (IABC) algorithm [118], discrete ABC (DABC) [14], best so far ABC [119], parallel ABC (pABC) algorithm [120], beer froth ABC [5] etc. In terms of computational time and solution efficiency, the obtained results are acceptable. At the same time, finding a solution for larger JSSP instances is a challenging task. These findings motivate researchers to continue their work in order to solve LSJSSP.

In light of the above, this chapter proposes a solution to the LSJSSP instances by using an efficient ABC-based algorithm called Fully Informed Artificial Bee Colony (FABC).The FABC algorithm was developed by K. Sharma et. al. [121]. The FABC algorithm is designed by taking inspiration from the Gbest-guided ABC (GABC)[9]. In 2010, Zhu and Kwong proposed an improved ABC algorithm, GABC [9], that improved exploitation by incorporating information from the global best (Gbest) solution into the solution search equation. However, since every agent (solution) is inspired by the global best approach, there is a good possibility of swarm stagnation. Therefore, the onlooker phase of the GABC algorithm is modified to improve the exploration ability of the search agents. In the modified onlooker bee phase “Fully Informed Learning” [122] strategy is incorporated. In this chapter the FABC algorithm is applied to solve 105 LSJSSP instances. The results are analysed and compared to other important methods available in the literature. The obtained findings substantiate the validity of the proposed strategy.

5.2 Fully Informed ABC

The employed bee phase, onlooker bee phase, and scout bee phase are the three key phases of the Fully Informed ABC (FABC) algorithm. In the following sections, we will go through each phase in depth. The solution agents are initially initialised as follows:

5.2.1 Initialization of the solution agents

The solution agents are randomly initialized in the give search space. If the search range of the given problem is $[B_{minj}, B_{maxj}]$ then the total number of solution agents (TSA) are initialized as follows:

$$SA_{ij} = B_{minj} + r[0,1](B_{maxj} - B_{minj}) \quad (5.1)$$

here SA_i represents the i^{th} solution agent in the swarm, B_{minj} and B_{maxj} are bounds of SA_i in j^{th} dimension whereas $r[0, 1]$ represents the uniformly distributed random number. The range of r is $[0, 1]$.

The initialization phase is same in all the SIAs. After this phase, the FABC executes its three phase namely, employed bee, onlooker bee, and scout bee Cyclically.

5.2.2 Employed bee phase

In the employed bee phase of the FABC, every solution will get chance to update its position using the following equation.

$$SAU_{ij} = SA_{ij} + r[0, 1](SA_{ij} - SA_{kj}) + \psi_{ij}(Best_j - SA_{ij}) \quad (5.2)$$

In equation 5.2, SAU_{ij} is the updated position of solution SA_{ij} . $Best_j$ is the j^{th} element of the best solution found so far. Further, ψ_{ij} is a number randomly generated in the range $[0, PC]$, where PC is a positive constant and SA_{kj} is a neighbouring solution agent. It is clear from equation 5.2 that the solution agents update their positions while learning from the nearby agents as well as attracting towards the best solution agent in the swarm. The term $\psi_{ij}(Best_j - SA_{ij})$ helps the swam to converge at the best solution location but this may lead to pre-mature convergence. Here the parameter PC helps in balancing the exploration and convergence ability of the FABC algorithm. After getting the updated position of the solution agent, a greedy selection mechanism (GSM) is applied between the update position SMU_{ij} and the old position SM_{ij} . The best one is selected for the next phase.

5.2.3 Onlooker bees phase

During this process, employed bees exchange details about their food source with onlooker bees in the comb, such as the quality, distance and direction of the food source. In terms of FABC algorithm, this phase is used to update the solutions shared by the employed bee phase on the basis of their quality. The quality of the solutions are measured using the probability $prob_i$ which is a function of fitness of the solution agent. The $prob_i$ is calculated using following equation:

$$prob_i = \frac{0.9 \times fit_i}{maxfit} + 0.1, \quad (5.3)$$

In equation 5.3, fit_i shows the fitness of i^{th} solution agent whereas $maxfit$ represents the maximum fitness in the swarm. On the basis of this $prob_i$, the quality of the solution agent is evaluated and based on that the solution agent is given chance to update its position. Therefore, we can say that in this phase the better solutions will get more chance to update the positions in compare to the less fit solutions. Further, in this phase, the fully informed learning strategy is applied in the position update process of the solution agents. The position update equation is shown below:

$$SAU_{ij} = x_{ij} + r[0, 1] \frac{\sum_{k=1}^{TSA} (SA_{ij} - SA_{kj})}{TSA} + \psi_{ij}(Best_j - SA_{ij}) \quad (5.4)$$

here TSA is the total number of solution agents while other parameters are same as mentioned in equation 5.2. Here, it can be observed that a solution agent achieve a new position while learning form all the solution agents of swarm as well as direction of the

best solution of the swarm. As the learning from all the solutions is involve in this position update process, the possibility of pre-mature convergence is reduced. the new position is achieved. So, in the fully informed learning, to update its location in the search space, the agent gathers knowledge from the best solution in the current swarm as well as all other adjacent solutions. The new position of the solution agent is compared with the old one using the GSM, and the best candidate solution will take part in the next generation of the FABC_i

5.2.4 Scout bees phase

The scout bee phase is used to reduce the possibility of stagnation of the swarm. The stagnation is the situation in which all the solution agents gathered at the same location of the search space hence the inter-agent distance becomes negligible. As the position update process depends on the inter-agent distance, the movement of the solution agents is reduced. Hence the solution agents stagnated at the same location.

In this phase, the number of update of every solution agent is checked. If any of the solution agent is not updating its position up to the pre-defined number (*limit*) of iterations then that solution agent is considered as exhausted and a new solution is randomly generated in the search space in place of that solution. Hence, the situation of stagnation can be reduced while introducing the fluctuation in the swarm through random initialization.

5.3 Job shop scheduling problem organisation

The LSJSSP can be interpreted in following manner: There are a set of n jobs to be processed using m machines. To complete the execution, each job has to be passed through all the m systems in a given predefined sequence. Each job consists of total m operations. To perform operations a job uses one of the machine. When any of the job is executing on any machine it cannot be interrupted by other jobs. The total number of operations are $m \times n$ that are scheduled on m systems [123].

The objective of the LSJSSP is to minimize the total completion time for all the jobs i.e. makespan (MS). Mathematically the problem is stated as :

$$\text{Minimize } MS_{max} \quad (5.5)$$

where, $MS_{max} = \max(MS_1, MS_2, MS_3, MS_4, \dots, MS_n)$. $MS_1, MS_2, MS_3, MS_4, \dots, MS_n$ are the completion time for all the n jobs. Followings are the constraints for LSJSSP [116]:

- Each system can process at most one operation at a time.
- The completion time of any operation must be a positive integer.
- Precedence relationships among the different jobs must be satisfied.

5.4 FABC for LSJSSP

The FABC algorithm is used to solve LSJSSP instances, and the whole method is detailed here. Since LSJSSP is a discrete optimization problem, a solution in the proposed algorithm is a discrete valued vector (representing a potential operation scheduling list). The reordering of jobs for FABC is used to estimate each solution in the search field. To generate the discrete

valued sequence from a continuous valued vector we have used random key encoding (RKE) scheme [124].

In RKE encoding scheme, first a continuous valued vector is sorted in an ascending order using an integer series from 1 to $n \times m$, where n represents the total number of jobs and m shows the total number of available systems. As each job has to go through m systems for completing its execution so further transformation from this integer sequence is performed using (Integer value mod $n + 1$). The integer series is transformed to operation order sequence using this transformation, and each job index has m occurrences. Figure 5.1 depicts the transformation of a continuous valued vector into a discrete valued vector, followed by an operation scheduling sequence. Our goal is to find an operation sequencing list (a vector of discrete values) that decreases the makespan value. The goal is to figure out a series of operations that reduces the overall time it takes to complete all of the jobs. The detailed procedure is described in the subsequent steps:

Continuous valued solution	0.9	0.6	0.8	0.2	0.5	0.3
Decoded as	6	4	5	1	3	2
Operation sequence	1	2	3	2	1	3

Figure 5.1: Random Key (RKE) Encoding Scheme

5.4.1 Initialization stage

The parameters of the proposed FABC algorithm namely, total number of solution agents, number of employed and onlooker solution agents, and total number of iterations are initialized. Each solution agent is initialized in the search space using the equation 5.1. As all the initialized sources are continuous in nature, RKE scheme is used to generate the corresponding discrete valued operation sequence. Now the MS value (objective value) for each operation sequence is calculated.

5.4.2 Employed honeybee stage

At this stage, using equation 5.2 all the continuous valued solution agents are modified. The solution agents is modified as per the information of the neighbouring agents. The updated solution agent is in continuing form, so again RKE encoding scheme is applied to alter this continuous valued solutions in to corresponding discrete operation sequence list. The MS value for this operation sequence is computed. If the corresponding MS value is better then the previous value then the solution agent corresponding to this operation sequence is selected for the next generation.

5.4.3 Onlooker honeybee stage

In onlooker honeybee stage, the probability for all the solution agents are assessed using the equation 5.3. The solution agents are chosen and updated as per the equation 5.3 and 5.4 respectively. Again the solution agent is updated based upon the information obtained from the neighbouring solution agents. To obtain the corresponding operation sequence,

RKE scheme is applied on the produced continuous valued solution agent. The MS value is computed from the generated operation sequence. GSM is used to choose new solution agent for the upcoming generation.

5.4.4 Scout honeybee stage

If a solution agent does not update its position up to the *limit*, then it is discarded and re-initialized in the search space using the equation 5.1. The produced solution agent is in continuous form, again RKE scheme is applied to obtain the corresponding operation sequence. Calculate the MS value from this operation sequence.

The pseudo-code of the designed approach for LSJSSP is shown in Algorithm 5.1.

Algorithm 5.1 FABC algorithm for LSJSSP

Parameter Initialization
 Total solution agents = TSA
 D (Dimension) = $m \times n$.
 Total generation count = MGN
 CurrentIndex=1
 Solution agents initialization in the search space using equation 5.1
 Conversion of continuous valued solution agents into an operation sequence (discrete valued solutions) to deal LSJSSP using RKE scheme
 MS value Computation for every operation sequence.
 While (CurrentIndex \leq MGN) do

- Step 1: Employed honeybee stage
 - Revise the location of every solution agent as per the equation 5.2
 - The newly generated solution agent is in continuous form only, so it is converted in to a discrete valued operation sequence using RKE scheme.
 - Compute the MS value from the generated operation sequence.
 - Apply GSM to select the new solution agent on the basis of the MS value of its respective operation sequence.
 - The former solution agent will be substituted by the new solution agent if respective operation sequence vector has a better MS value
- Step 2: Onlooker honeybee stage
 - Probability $prot_i$ computation using equation 5.3 for each solution agent
 - Revise the location of solution agent using the equation 5.4 selected as equation 5.3.
 - Obtain the new discrete operation sequence from the recently revised continuing solution agents using RKE scheme.
 - MS value computation for recently produced operation sequence
 - Apply GSM to select the new solution agent for the next generation.
 - The former solution agent will be substituted by the new solution agent if corresponding discrete solution sequence has a better MS value.
- Step 3: Scout honeybee stage
 - If a solution agent does not modify its position up to *limit*.
 - Randomly initialized that solution agent as per equation 5.1 in the search space.
 - Apply RKE scheme for producing discrete solution vector from this continuous valued solution agent.
 - MS value computation for the operation scheduling list
- Step 4: Memorize the best solution found so far.
- CurrentIndex=CurrentIndex+1.
end while

Output the best solution

5.5 Implementation and experimental results

To prove the effectiveness of FABC algorithm, it is applied on LSJSSP instances. Following 105 LSJSSP instances are considered for experimentation [125, 126, 127].

Table 5.1: Comparison in terms of best MS value for SMV instances

Instance	Size	LB	UB	FABC	BRKGA-JSP	TS/SA	TS
SWV01	20 × 10	1407	1407	1407	1407	1412	-
SWV02	20 × 10	1475	1475	1475	1475	1475	-
SWV03	20 × 10	1369	1398	1395	1398	1398	-
SWV04	20 × 10	1450	1474	1465	1470	1470	-
SWV05	20 × 10	1424	1424	1424	1425	1425	-
SWV06	20 × 15	1591	1678	1674	1675	1679	-
SWV07	20 × 15	1446	1600	1572	1594	1603	-
SWV08	20 × 15	1640	1763	1762	1755	1756	-
SWV09	20 × 15	1604	1661	1650	1656	1661	-
SWV10	20 × 15	1631	1767	1736	1743	1754	-
SWV11	50 × 10	2983	2983	2983	2983	-	2983
SWV12	50 × 10	2972	2979	2975	2979	-	2979
SWV13	50 × 10	3104	3104	3104	3104	-	3104
SWV14	50 × 10	2968	2968	2968	2968	-	2968
SWV15	50 × 10	2885	2886	2885	2901	-	2886

- 15 SWV instances
- 50 TA instances
- 40 DMU instances

To attain the least MS value for all these 105 LSJSSP instances is the main goal. The experimental setting is listed as below:

1. Number of run =10
2. Number of maximum iteration =2000
3. Number of solution agents TSA =50
4. Dimension D = Number of systems × Number of jobs
5. limit = D × TSA

Table 5.2: Comparison in terms of best MIS value for TA instances

Instance	Size	LB	UB	FABC	BRKGA-JSP	GES	AlgFix	I-TSAB	TS/SA	DHS	TLBO	NKPR
TA01	15 × 15	1231	1231	1231	1231	1231	1231	-	1231	1321	1526	1485
TA02	15 × 15	1244	1244	1244	1244	1244	1244	-	1244	1313	1538	1476
TA03	15 × 15	1218	1218	1218	1218	1218	1218	-	1218	1327	1594	1470
TA04	15 × 15	1175	1175	1175	1175	1175	1175	-	1175	1285	1550	1519
TA05	15 × 15	1224	1224	1224	1224	1224	1224	-	1224	1315	1551	1381
TA06	15 × 15	1238	1238	1238	1238	1238	1238	-	1238	1346	1538	1517
TA07	15 × 15	1227	1227	1227	1227	1228	1228	-	1228	1322	1546	1460
TA08	15 × 15	1217	1217	1217	1217	1217	1217	-	1217	1304	1534	1446
TA09	15 × 15	1274	1274	1274	1274	1274	1274	-	1274	1386	1614	1551
TA10	15 × 15	1241	1241	1241	1241	1241	1241	-	1241	1334	1542	1365
TA11	20 × 15	1323	1357	1355	1357	1357	1358	1361	1359	1520	1876	1687
TA12	20 × 15	1351	1367	1367	1367	1367	1367	-	1371	1563	1856	1770
TA13	20 × 15	1282	1342	1342	1344	1344	1342	-	1342	1513	1849	1713
TA14	20 × 15	1345	1345	1345	1345	1345	1345	-	1345	1477	1780	1748
TA15	20 × 15	1304	1339	1337	1339	1339	1339	-	1339	1537	1929	1788
TA16	20 × 15	1302	1360	1360	1360	1360	1360	-	1360	1543	1852	1716
TA17	20 × 15	1462	1462	1462	1462	1469	1473	1462	1464	1607	1941	1781
TA18	20 × 15	1369	1396	1396	1396	1401	1396	-	1399	1601	1817	1776
TA19	20 × 15	1297	1332	1331	1332	1332	1332	-	1335	1524	1842	1722
TA20	20 × 15	1318	1348	1348	1348	1348	1348	-	1351	1554	1902	1710
TA21	20 × 20	1539	1643	1642	1642	1647	1643	1644	1644	1854	2399	2165
TA22	20 × 20	1511	1600	1600	1600	1600	1600	1600	1600	1852	2241	2126
TA23	20 × 20	1472	1557	1557	1557	1558	1557	1647	1560	1765	2210	2145
TA24	20 × 20	1602	1646	1646	1646	1633	1646	1647	1646	1829	2241	2173
TA25	20 × 20	1504	1595	1594	1594	1596	1595	1595	1597	1792	2324	2117
TA26	20 × 20	1539	1645	1641	1643	1647	1647	1645	1645	1863	2299	2206
TA27	20 × 20	1616	1680	1680	1680	1685	1686	1680	1680	1905	2436	2194
TA28	20 × 20	1591	1603	1600	1603	1614	1613	1614	1603	1819	2333	2100
TA29	20 × 20	1514	1625	1625	1625	1625	1625	-	1627	1853	2280	2146
TA30	20 × 20	1473	1584	1584	1584	1584	1584	1584	1584	1812	2247	2103
TA31	30 × 15	1764	1764	1764	1764	1764	1766	-	1764	2037	2528	2382
TA32	30 × 15	1774	1790	1781	1785	1793	1790	-	1795	2106	2591	2482
TA33	30 × 15	1778	1791	1791	1791	1799	1791	1793	1796	2091	2685	2511
TA34	30 × 15	1828	1829	1832	1829	1832	1832	1829	1831	2089	2508	2480
TA35	30 × 15	2007	2007	2007	2007	2007	2007	-	2007	2139	2509	2512
TA36	30 × 15	1819	1819	1819	1819	1819	1819	-	1819	2086	2705	2395
TA37	30 × 15	1771	1771	1728	1771	1779	1784	1778	1778	2067	2512	2436
TA38	30 × 15	1673	1673	1673	1673	1673	1673	-	1673	1980	2488	2250
TA39	30 × 15	1795	1795	1795	1795	1795	1795	-	1795	2010	2439	2501
TA40	30 × 15	1631	1673	1665	1669	1680	1979	1674	1676	1986	2455	2380
TA41	30 × 20	1859	2006	2006	2008	2008	2022	-	2018	-	-	-
TA42	30 × 20	1867	1945	1934	1937	1953	1953	1956	1953	-	-	-
TA43	30 × 20	1809	1814	1836	1852	1870	1869	1859	1858	-	-	-
TA44	30 × 20	1927	1983	1983	1983	1991	1992	1984	1983	-	-	-
TA45	30 × 20	1900	1997	2000	2004	2004	2000	2000	2000	-	-	-
TA46	30 × 20	1940	2008	2000	2004	2011	2011	2021	2010	-	-	-
TA47	30 × 20	1789	1897	1892	1894	1903	1902	1903	1903	-	-	-
TA48	30 × 20	1912	1945	1940	1962	1955	1962	1953	1955	-	-	-
TA49	30 × 20	1915	1966	1962	1964	1969	1974	-	1967	-	-	-
TA50	30 × 20	1807	1925	1925	1925	1931	1927	1928	1931	-	-	-

Table 5.3: Comparison in terms of best MS value for DMU instances

Instance	Size	LB	UB	FABC	BRKGA-JSP	TS	GES	ITSAB	AlGfix
DMU01	20 × 15	2501	2563	2563	2563	2566	2566	2517	2563
DMU02	20 × 15	2651	2706	2704	2706	2711	2706	2715	2706
DMU03	20 × 15	2731	2731	2731	2731	-	2731	-	2731
DMU04	20 × 15	2601	2669	2669	2669	-	2669	-	2669
DMU05	20 × 15	2749	2749	2749	2749	-	2749	-	2749
DMU06	20 × 20	2834	3244	3242	3244	3254	3250	3265	3244
DMU07	20 × 20	2677	3046	3045	3046	-	3053	-	3046
DMU08	20 × 20	2901	3188	3188	3188	3191	3197	3199	3188
DMU09	20 × 20	2739	3092	3091	3092	-	3092	3094	3096
DMU10	20 × 20	2716	2984	2982	2984	-	2984	2985	2984
DMU11	30 × 15	3395	3453	3454	3445	3455	3453	3470	3455
DMU12	30 × 15	3481	3516	3512	3513	3516	3518	3519	3522
DMU13	30 × 15	3681	3681	3681	3681	3681	3697	3698	3687
DMU14	30 × 15	3394	3394	3394	3394	-	3394	3394	3394
DMU15	30 × 15	3332	3343	3343	3343	-	3343	-	3343
DMU16	30 × 20	3726	3759	3748	3751	3739	3781	3787	3772
DMU17	30 × 20	3697	3836	3828	3830	3842	3848	3854	3836
DMU18	30 × 20	3844	3846	3844	3844	3846	3849	3854	3852
DMU19	30 × 20	3650	3775	3769	3770	3784	3807	3823	3775
DMU20	30 × 20	3604	3712	3712	3712	3716	3739	3740	3712
DMU21	40 × 15	4380	4380	4380	4380	-	4380	-	4380
DMU22	40 × 15	4325	4725	4725	4725	-	4725	-	4725
DMU23	40 × 15	4668	4668	4668	4668	-	4668	-	4668
DMU24	40 × 15	4648	4648	4648	4648	-	4648	-	4648
DMU25	40 × 15	4164	4164	4164	4164	-	4164	-	4164
DMU26	40 × 20	4647	4647	4647	4647	4647	4667	4679	4688
DMU27	40 × 20	4848	4848	4848	4848	-	4848	4848	4848
DMU28	40 × 20	4692	4692	4692	4692	-	4692	-	4692
DMU29	40 × 20	4691	4691	4691	4691	-	4691	4691	4691
DMU30	40 × 20	4732	4732	4732	4732	-	4732	4732	4749
DMU31	50 × 15	5640	5640	5640	5640	-	5640	-	5640
DMU32	50 × 15	5927	5927	5927	5927	-	5927	-	5927
DMU33	50 × 15	5728	5728	5728	5728	-	5728	-	5728
DMU34	50 × 15	5385	5385	5385	5385	-	5385	-	5385
DMU35	50 × 15	5635	5635	5635	5635	-	5635	-	5635
DMU36	50 × 20	5621	5621	5621	5621	-	5621	-	5621
DMU37	50 × 20	5851	5851	5851	5851	-	5851	5851	5851
DMU38	50 × 20	5713	5713	5713	5713	-	5713	-	5713
DMU39	50 × 20	5747	5747	5747	5747	-	5747	-	5747
DMU40	50 × 20	5577	5577	5577	5577	-	5577	-	5577

The parametric ambience for the FABC approach and the other considered approaches are kept same in terms of swarm size and maximum number of iterations to carry out an equitable comparison.

The reported results of FABC are compared with the following state-of-art algorithms available in the literature:

- Biased random key genetic algorithm (BRKGA-JSP) [116]
- A guided tabu search for LSJSSP (NKPR) [128]
- Teaching learning based optimization method (TLBO) [129]
- Differential based harmony search (DHS) algorithm [115]
- A tabu search to solve LSJSSP (TS) [130]
- An advanced tabu search algorithm for LSJSSP (i-TSAB) [131]
- AlgFix [132]
- A tabu search/simulated annealing algorithm for LSJSSP (TS/SA) [133]
- Global equilibrium search technique (GES) [134]

The obtained results for the above three instances are represented in Tables 5.1 to 5.3. These tables list the name of the instance, its size, the lower bound (LB), the upper bound (UB) for the best known solution (BKS), the BKS obtained by FABC approach, and BKS value obtained from the compared algorithms. The obtained results for all the instances demonstrate that the proposed FABC is superior approach in reference to *MS* value during assessment with other considered approaches.

Further to analyse the outcomes, average relative percentage error (RPE) is also calculated and compared as tabulated in Table 5.4. The value of RPE is computed (with respect to the UB value of an instance) as per demonstrated in equation 5.6.

$$RPE = 100 \times (BKS_{algo} - UB)/UB \quad (5.6)$$

Here, BKS_{algo} represents the *MS* value obtained using the considered approaches. The attained outcomes of Table 5.4 demonstrate the significant improvement in the average RPE which assures the authenticity of the introduced approach.

5.6 Conclusion

This article proposed a solution to solve 105 large scale instances of job shop scheduling problem (LSJSSP) using an efficient fully informed artificial bee colony (FABC). In FABC algorithm, to improve the exploration ability during the solution search process, a fully informed learning strategy is incorporated in the onlooker bee phase of the Gbest ABC algorithm. The *MS* time is used as an evaluation criterion in the LSJSSP. The results are analysed and compared to cutting-edge techniques proposed by a number of researchers. According to the results of the experiments, the proposed solution gives better solution. In Future, some more performance metrics may be considered for experimentation.

Table 5.4: Comparison based upon Average RPE

Approach	Instances Solved	ARPE (%)	FABC(%)	Improvement (%)
TA instances				
BRKGA-JSP [116]	50	0.015	-0.093	0.108
GES [134]	50	0.218	-0.093	0.311
AlgFix [132]	50	0.556	-0.093	0.649
i-TSAB [131]	25	0.269	-0.15	0.419
TS/SA [133]	50	0.156	-0.093	0.249
DHS [115]	40	12.412	-0.104	12.516
TLBO [129]	40	36.883	-0.104	36.987
NKPR [128]	40	28.951	-0.104	29.055
DMU Instances				
BRKGA-JSP [116]	40	-0.021	-0.027	0.006
TS [130]	13	0.097	-0.072	0.169
GES [134]	40	0.107	-0.027	0.134
i-TSAB [131]	20	0.24	-0.06	0.3
AlgFix [132]	40	0.056	-0.027	0.083
SWU Instances				
BRKGA-JSP [116]	15	-0.156	-0.364	0.52
TS/SA [133]	10	-0.073	-0.529	0.602
TS [130]	5	0	-0.0338	0.0338

CHAPTER 6

FITNESS BASED PARTICLE SWARM OPTIMIZATION

Chapter 6

Fitness based Particle Swarm Optimization

Particle Swarm Optimization Algorithm (PSOA) is a popular population-based approach for addressing nonlinear and difficult optimization problems. It's a simple to create swarm-based probabilistic algorithm, but it has drawbacks including easy local optima and sluggish convergence in the latter stages. PSO includes a unique position update phase, dubbed fitness based position updating in PSO, to reduce the risk of stagnation while increasing convergence speed. The recommended phase is based on the spectator bee phase of the Artificial Bee Colony (ABC) algorithm. In the proposed position update phase, solutions alter their positions depending on probability, which is a function of fitness. This method allows better solutions in the solution search process to update their places more often. The recommended approach is called Fitness Based Particle Swarm Optimization (FitPSO). FitPSO's efficiency is demonstrated by comparing it to the traditional PSO 2011 and ABC methods on 15 well-known benchmark problems and three real-world engineering optimization difficulties.

This chapter goes into the fitness-based PSO in detail (FitPSO). In section 6.2, standard PSO is discussed. Fitness based Particle Swarm Optimization (FitPSO) is proposed in section 6.3. The proposed strategy's performance is examined in Section 6.4. The section 6.5 explains how FitPSO may be used to solve engineering optimization issues. Finally, the chapter is finished in section 6.6.

6.1 Introduction

After being inspired by the social behaviour of fish schooling and birds flocking while seeking for food, Kennedy and Eberhart [88], [135] developed a swarm intelligence based optimization technique called Particle swarm optimization (PSO) in 1995. PSO is an easy-to-understand and use population-based meta heuristic optimization approach. PSO is a preferable alternative for multi-model, non-convex, non-linear, and complicated optimization problems, but it, like any other evolutionary method, has limitations including entrapment in local optima ([136]) and computational inefficiency ([137]). PSO's applicability is limited by these factors [138]. Researchers are always attempting to achieve these goals while investigating the use of PSO, such as enhancing convergence speed and ignoring local optima. As a result, many PSOA modifications have been proposed to solve these flaws [139], [136], [140], [141], [142],

and [137]. However, achieving both objectives at the same time is difficult. Liang et al., for example, proposed the comprehensive-learning PSO (CLPSO) [136], which tries to ignore local optima but has sluggish convergence. Ratnaweera et al. [139] proposed time-varying acceleration parameters to balance cognitive and social components in the early and late stages of development. Zhan et al. [140] also attempted to use acceleration parameters that increased or reduced depending on the phase of searching or using search space. Zhang et al. [141] looked examined the impact of these variables on position expectation and variance, and found that setting the cognitive acceleration factor to 1.85 and the social acceleration factor to 2 improves system stability. Gai-yun et al. [142] also researched on cognitive and social component self-adaptation. To balance the exploration and exploitation capabilities, a fitness-based position update approach is presented in this chapter of PSO. In addition, the PSO velocity update equation has been tweaked to increase convergence.

6.2 Standard PSO Strategy

PSO is an optimization approach that mimics the behaviour of flocking birds. PSO is a living, breathing community of active, interacting individuals with relatively little innate intelligence. In PSO, the whole group is referred to as a *swarm*, while each individual is referred to as a *particle*, which represents a potential candidate's answer. By watching the behaviour of neighbouring birds who looked to be near the food source, the swarm seeks food for itself through social learning. Initially, each particle is randomly started inside the search area and remembers information about its personal best position, *pbest*, swarm best position, *gbest*, and current velocity, V , with which it is travelling. Each particle adjusts its location based on these three variables. As a result, the entire swarm advances in a better direction while using a collaborative trial and error technique, eventually settling on a single best known answer.

The i^{th} particle of the swarm is represented by a D-dimensional vector, $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$, in a D-dimensional search space. Another D-dimensional vector $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ represents the velocity of this particle. $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$ denotes the previously best visited position of the i^{th} particle. The best particle in the swarm's index is g . For movement, the PSO swarm employs two equations: the *velocity* update equation and the *position* update equation. The velocity of the i^{th} particle is updated using equation (6.1) as the velocity update equation. Equation (6.2) is used to update the position..

$$v_{ij} = v_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (p_{gj} - x_{ij}) \quad (6.1)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (6.2)$$

The dimension is represented by $j = 1, 2, \dots, D$, while the particle index is represented by $i = 1, 2, \dots, S$. S is the swarm's size, and c_1 and c_2 are constants (typically $c_1 = c_2$), often known as cognitive and social scaling parameters or simply acceleration coefficients, respectively. r_1 and r_2 are uniformly distributed random numbers in the range $[0, 1]$.

The right hand side of the velocity update equation (6.1) has three terms. The first term v_{ij} represents the memory of the prior direction of movement, which may be regarded of as a momentum term that prevents the particle from changing direction dramatically. The second phrase, $c_1 r_1 (p_{ij} - x_{ij})$, refers to the cognitive component or persistence, which pulls particles back to their prior optimal state and allows swarms to search locally. The

last term, $c_2r_2(p_{gj} - x_{ij})$, is known as the social component and is responsible for global search. It allows individuals to compare themselves to others in their group. The following is a description of the Pseudo-code for Particle Swarm Optimization:

Algorithm 6.1 PSOA Pseudo-code:

```

Parameters  $w$ ,  $c_1$ , and  $c_2$  are initialized;
Initial particle locations and respective velocities are initialized;
Calculate the objective value of the initial particles;
Save gbest and pbest positions;
while termination criteria do
  for each particle,  $X_i$  do
    for each  $j$ ,  $x_{ij}$  do
      (i) Calculate the velocity  $v_{ij}$  using (6.1);
      (ii) Calculate the new location  $x_{ij}$  using (6.2);
    end for
  end for
  Calculate the objective value of the updated solution;
  Through greedy selection get the new value of gbest and pbest;
end while
Get the particle having best objective value;

```

Initially, two variants of PSOA were published in the literature based on neighbourhood size: the global version of PSO (PSO-G), which is the original PSO, and the local version of PSO (PSO-L), [39]. The term p_g in the social component of the velocity update equation (6.1) is the only difference between PSO-G and PSO-L. It refers to the best particle of the whole swarm in PSO-G, and the best particle of the individual's vicinity in PSO-L. The PSO-social G's network is based on the star topology, which allows for faster convergence but is more prone to converge prematurely. PSO-L, on the other hand, employs a ring social network architecture, with smaller regions specified for each particle. It is clear that because PSO-L has less particle interconnectivity, it is less prone to being caught in local minima, albeit at the cost of delayed convergence. PSO-G is better for unimodal situations whereas PSO-L is better for multimodal ones.

The velocity update equation determines the balance between PSO's exploration and exploitation capabilities. Because there are no velocity constraints in Basic PSO, particles far from gbest will take massive steps in early iterations and are very likely to exit the search space. The velocity clamping notion was presented to balance particle update step size by regulating velocity. Velocity clamping is used to keep velocity within its bounds when it surpasses them. To prevent velocity clamping and strike a balance between exploration and exploitation, a new parameter called inertia weight [40] was added to the velocity update equation, as follows:

$$v_{ij} = w * v_{ij} + c_1r_1(p_{ij} - x_{ij}) + c_2r_2(p_{gj} - x_{ij}) \quad (6.3)$$

where w stands for inertia weight. The proposed PSOA is detailed in the next section.

6.3 Fitness Based PSOA

However, while the conventional PSO may provide a decent solution at a much faster rate, it struggles to refine the optimal solution when compared to other optimization approaches,

owing to less variety in later search [?]. On the other hand, in PSO, problem-based parameter tuning is also crucial in order to obtain the best solution correctly and efficiently [143]. There are three terms in the typical PSO velocity update equation (6.1). The first phrase refers to the capacity to search globally, while the second and third terms refer to the ability to share knowledge socially and cognitively. More cognitive capability forces particles to move quickly toward their personal best position, whereas more social knowledge forces particles to move quickly toward their global best position. The acceleration factors c_1 and c_2 direct the movement of the swarm towards the optimal solution, as shown in (6.1). As a result, the acceleration coefficients c_1 and c_2 need be fine-tuned to get the desired result.

More value of the cognitive component compared to the social component results in excessive wandering of individuals through the search space, while more value of the social component may result in particles converge prematurely toward a local optimum, according to Kennedy and Eberhart ([88]). These two components are crucial for PSO's exploration and exploitation capabilities to be balanced. As a result, two changes are recommended in this chapter to improve PSO's solution search efficiency.

1. The modified Velocity update equation (refer eq. 6.3) is as follows:

$$v_{ij} = w \times v_{ij} + c \times r(p_{gj} - x_{ij}) \quad (6.4)$$

The *Pbest* component (cognitive component) is eliminated from the velocity update equation of PSO ($c_1 r_1 (p_{ij} - x_{ij})$), as shown in equation (6.4). The magnitude of each individual's velocity will now be determined by their distance from the current global best solution. As a result, this technique will enhance PSO's exploitation capabilities.

2. PSO now includes a new position updating mechanism inspired by the Artificial Bee Colony (ABC) algorithm's spectator bee phase [144]. All employed bees hunt for food sources and compute their fitness using equation (6.5) during the employed bee phase of ABC:

$$fitness_i = \begin{cases} 1/(1 + f_i), & \text{if } f_i \geq 0 \\ 1 + abs(f_i), & \text{if } f_i < 0. \end{cases} \quad (6.5)$$

The observer bees then assess the given information and choose a solution with a probability, $prob_i$, that is connected to its fitness. Equation (6.6) may be used to determine the probability $prob_i$:

$$prob_i(G) = \frac{0.9 \times fitness_i(G)}{maxfit(G)} + 0.1, \quad (6.6)$$

where G represents the iteration counter, $fitness_i(G)$ represents the fitness value of the i^{th} solution, and $maxfit(G)$ represents the maximum fitness of the solutions in the G^{th} iteration. The ABC position update equation (6.7) is as follows:

$$y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (6.7)$$

where $k \in \{1, 2, \dots, S\}$ and $j \in \{1, 2, \dots, D\}$ are randomly chosen indices, k must be distinct from i , phi_{ij} is a random integer between $[-1, 1]$, and x_{kj} is a random solution in the current population. In the basic ABC, only one dimension is updated at a time in the employed or onlooker bee phase. This update occurs during the onlooker bee phase and is based on a likelihood that is a function of fitness.

The suggested position update approach is combined with PSO, resulting in a novel algorithm known as Fitness-based PSO (FitPSO). Algorithm 6.2 is used after basic PSO operators in FitPSO. FitPSO is now more capable of exploitation in improved search areas because to the addition of Algorithm 6.2. It's reasonable to assume this since, after applying fundamental PSO operators in FitPSO, better candidate solutions are given more opportunities to update themselves than poorer candidates. Algorithm 6.2 shows the pseudo-code for the suggested position update method that works after PSO operators.

Algorithm 6.2 Position update based on Fitness of the solution:

```

for every solution,  $x_i$  do
  if  $prob_i > random(0,1)$  then
     $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$ ,
    evaluate the fitness of  $\vec{y}_i$ ,
    Select the best one between  $\vec{y}_i$  and  $\vec{x}_i$ ,
  end if
end for

```

Algorithm 6.3 shows the pseudo-code for the proposed FitPSO algorithm.

Algorithm 6.3 FitPSO Algorithm(FitPSOA):

```

The algorithm variables  $w$ , and  $c$  and  $S$  are initialized;
The solutions and their respective velocities are initialized.
Calculate the objective value (fitness) of every individual.
Identify the global best solution (gbest).
while termination condition do
  for each solution,  $X_i$  do
    for each dimension  $j$  of  $x_{ij}$  do
      (i) Calculate the new velocity  $v_{ij}$  using (6.4);
      (ii) Calculate the new Location  $x_{ij}$  using (6.2);
    end for
  end for
  Calculate the objective value (fitness) of the new solutions.
  Through greedy select get the best solution in the swarm as gbest.
  /*****FitPSO Phase *****/
   $t = 1, i = 1$ 
  while  $t \leq S$  do
    if  $prob_i > rand(0,1)$  then
      /*****  $prob_i$  (refer 6.6) *****/
       $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$ ,
      { $k, j$  is the solution selected randomly}
      Evaluate the velocity  $\vec{y}_i$ ;
      Get the best solution between  $\vec{y}_i$  and  $\vec{x}_i$ ;
       $t = t + 1$ 
    end if
     $i = i + 1$ 
    if  $i > S$  then
       $i = 1$ 
    end if
  end while
end while
Return the best solution from the swarm.

```

Table 6.1: Test problems; AE: Acceptable Error

Objective function	Search Range	Optimum Value	D	AE
$f_1(x) = \sum_{i=1}^L (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	[-30, 30]	$f(\bar{1}) = 0$	30	$1.0E - 02$
$f_2(x) = 10L + \sum_{i=1}^L [x_i^2 - 10 \cos(2\pi x_i)]$	[-5.12, 5.12]	$f(\bar{0}) = 0$	30	$1.0E - 05$
$f_3(x) = -\sum_{i=1}^D \sin x_i (\sin(\frac{ix_i^2}{\pi}))^{20}$	[C, π]	$f_{min} = -9.66015$	10	$1.0E - 05$
$f_4(x) = \sum_{i=1}^L x_i^2 + (\sum_{i=1}^D \frac{ix_i}{z})^2 + (\sum_{i=1}^L \frac{ix_i}{z})^4$	[-5.12, 5.12]	$f(\bar{0}) = 0$	30	$1.0E - 02$
$f_5(x) = \sum_{i=1}^L ix_i^4 + \text{random}[0, 1]$	[-1.28, 1.28]	$f(\bar{0}) = 0$	30	$1.0E - 05$
$f_6(x) = -\sum_{i=1}^{D-1} \left(\exp\left(\frac{-(x_i^2 + x_{i+1}^2 + c \cdot 5x_i x_{i+1})}{8}\right) \times 1 \right)$ where, $I = \cos(4\sqrt{x_i^2 + x_{i+1}^2 + c \cdot 5x_i x_{i+1}})$	[-5, 5]	$f(\bar{0}) = -L + 1$	10	$1.0E - 05$
$f_7(x) = \sum_{i=1}^L (x_i - 1)^2 - \sum_{i=2}^D x_i x_{i-1}$	$[-D^2, D^2]$	$f_{min} = -\frac{(D(D+4)(L-1))}{6}$	10	$1.0E - 01$
$f_8(x) = 100[x_2 - x_1^2]^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$	[-10, 10]	$f(\bar{1}) = 0$	4	$1.0E - 05$
$f_9(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(t_i^2 + b_i x_2)}{t_i^2 + b_i x_3 + x_4}]^2$	[-5, 5]	$f(0.192833 \quad 0.190836 \quad 0.123117 \quad 0.135766) = 3.075E - 04$	4	$1.0E - 05$
$f_{10}(x) = \sum_{i=1}^{L-1} (100(x_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$, $z = x - c + 1$, $x = [x_1, x_2, \dots, x_D]$, $c = [c_1, c_2, \dots, c_L]$	[-100, 100]	$f(c) = f_{bias} = 390$	10	$1.0E - 01$
$f_{11}(x) = (1 + (x_1 + x_2 + 1)^2) \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \cdot (30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	[-2, 2]	$f(0, -1) = 3$	2	$1.0E - 14$
$f_{12}(x) = -\cos x_1 \cos x_2 e^{((x_1 - \pi)^2 - (x_2 - \pi)^2)}$	[-10, 10]	$f(\pi, \pi) = -1$	2	$1.0E - 13$
$f_{13}(x) = \sum_{i=1}^L \left(\frac{x_1 x_2^2 t_i}{1 + x_1 t_i + x_2 t_i} - y_i \right)^2$	[-10, 10]	$f(3.13, 15.16, 0.78) = 0.4E - 04$	3	$1.0E - 03$
$f_{14}(x) = -\sum_{i=1}^L i \cos((i+1)x_1 + 1) \cdot \sum_{i=1}^5 i \cos((i+1)x_2 + 1)$	[-10, 10]	$f(7.083E-4, 8.580) = -186.730E$	2	$1.0E - 05$
$f_{15}(x) = - A \prod_{i=1}^D \sin(x_i - z) + \prod_{i=1}^D \sin(B(x_i - z))$, $A = 2.5$, $B = 5$, $z = 3C$	[0, 180]	$f(90, z) = -(A + 1)$	10	$1.0E - 02$

6.4 Experiments and Results

This section evaluates the suggested algorithm's performance in terms of accuracy, efficiency, and dependability.

6.4.1 Benchmark functions for testing

15 mathematical optimization problems (f_1 to f_{15}) of various features and difficulties are considered to validate the efficacy of FitPSO (listed in Table 6.1). All of these issues occur on a regular basis in nature.

6.4.2 Experimental setting

The success rate, average number of function evaluations, and mean error derived from the proposed FitPSO are all recorded. For the sake of comparison, the results for these test issues (Table 6.1) are likewise acquired from ABC and PSO. When implementing the proposed and other considered algorithms to solve the issues, the following parameter settings are used:

Parameter setting for PSO (Standard PSO 2011) and FitPSO:

- Swarm size $S = 50$,
- Inertia weight $w = 0.8$,
- Acceleration coefficients $c = c_1 = c_2 = 0.5 + \log 2$ (for PSO)[145],
- The number of run =100.
- The terminating criteria: Either acceptable error (Table 6.1) meets or maximum number of function evaluations (which is set to be 200000) is reached,

Parameter setting for ABC:

Table 6.2: Comparison based on average number of function evaluations, TP: Test Problem.

TP	ABC	PSO	FitPSO
f_1	196024	199407	193681
f_2	49984	200050	102911
f_3	29192	198326	69919
f_4	200000	196434	125547
f_5	200000	200000	200000
f_6	89944	195748	67107
f_7	200000	67527	44941
f_8	200000	52689	21049
f_9	172839	35314	15399
f_{10}	174699	187126	67504
f_{11}	112961	102884	87733
f_{12}	187003	9818	10599
f_{13}	28346	3397	2926
f_{14}	4861	80823	9836
f_{15}	55076	177156	104254

Table 6.3: Comparison based on success rate out of 100 runs, TP: Test Problem.

TP	ABC	PSO	FitPSO
f_1	5	1	23
f_2	100	0	100
f_3	100	3	100
f_4	0	31	100
f_5	0	0	0
f_6	97	6	100
f_7	0	100	100
f_8	0	100	100
f_9	21	100	100
f_{10}	23	58	98
f_{11}	60	51	59
f_{12}	13	100	100
f_{13}	100	100	100
f_{14}	100	73	100
f_{15}	99	27	99

- Colony size $S = 50$ [95, 96],
- $\phi_{ij} = rand[-1, 1]$,
- Number of food sources $SN = S/2$,
- The number of run =100,
- The terminating criteria: Either acceptable error (Table 6.1) meets or maximum number of function evaluations (which is set to be 200000) is reached,

6.4.3 Results Analysis of Experiments

Tables 6.2, 6.3 and 6.4 present the numerical results for the benchmark problems of Table 6.1 with the experimental settings shown in section 6.4.2. These tables show the results of the proposed and other considered algorithms in terms of average number of function evaluations (AFE), success rate (SR) and mean error (ME).

After analyzing the results, it can be said that FitPSO outperforms the considered algorithms most of the time in terms of accuracy (due to ME), reliability (due to SR) and efficiency (due to AFE). Some other statistical tests like the Mann-Whitney U rank sum test, acceleration rate (AR) [97] and boxplots have also been done in order to analyze the algorithms output more intensively.

Table 6.4: Comparison based on mean error, TP: Test Problem.

TP	ABC	PSO	FitPSO
f_1	1.34E+00	4.44E+01	6.80E+00
f_2	5.66E-06	3.87E+01	7.65E-06
f_3	3.84E-06	3.12E-01	5.04E-06
f_4	9.75E+01	2.20E-02	9.72E-03
f_5	1.18E+01	9.94E+00	8.95E+00
f_6	1.05E-02	1.48E+00	8.64E-06
f_7	8.89E-01	9.53E-06	8.98E-06
f_8	1.52E-01	8.17E-04	7.65E-04
f_9	1.69E-04	8.96E-05	8.21E-05
f_{10}	6.26E-01	1.69E+00	1.66E-01
f_{11}	1.19E-06	4.96E-14	4.22E-14
f_{12}	2.44E-05	5.34E-14	5.10E-14
f_{13}	1.95E-03	1.95E-03	1.88E-03
f_{14}	4.58E-06	9.54E-05	5.42E-06
f_{15}	7.89E-03	3.74E-01	7.81E-03

6.4.4 Statistical Analysis

Algorithms ABC, PSO and FitPSO are compared based on SR, AFE, and ME. From the results shown in Table 6.2, it is clear that FitPSO costs less on 9 test functions ($f_1, f_4, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{13}$) among all the considered algorithms. As these functions include unimodel, multimodel, separable, non separable, lower and higher dimension functions, it can be stated that FitPSO balances the exploration and exploitation capabilities efficiently for all kind of functions compared to the other considered algorithms. ABC outperforms FitPSO over test functions f_2, f_3, f_{14}, f_{15} , while PSO is outperforms over FitPSO on test functions f_{12} .

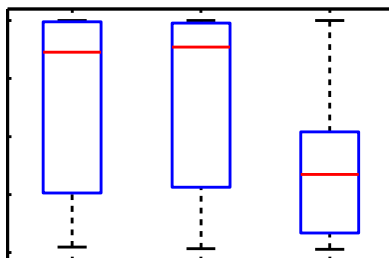


Figure 6.1: Boxplots graph for average number of function evaluation

Since boxplots [99] can efficiently represent the empirical distribution of results, the boxplots for average number of function evaluations for PSO, ABC and FitPSO have been represented in Figure 6.1. Figure 6.1 shows that FitPSO is cost effective in terms of function evaluations as interquartile range and median of average number of function evaluations are very low for FitPSO.

Though, it is clear from box plots that FitPSO is cost effective than ABC and PSO i.e., FitPSO's result differs from the other, now to check, whether there exists any significant difference between algorithm's output or this difference is due to some randomness, we require another statistical test. It can be observed from boxplots of Figure 6.1 that average number of function evaluations used by the considered algorithms to solve the different problems are not normally distributed, so a non-parametric statistical test is required to compare the performance of the algorithms. The Mann-Whitney U rank sum [146], a non-parametric test, is well established test for comparison among non-Gaussian data. In this chapter, this test is performed at 5% level of significance ($\alpha = 0.05$) between FitPSO - ABC and FitPSO - PSO.

Table 6.5: Comparison based on mean function evaluations and the Mann-Whitney U rank sum test at a $\alpha = 0.05$ significance level ('+' indicates FitPSO is significantly better, '-' indicates FitPSO is worse and '=' indicates that there is no significant difference), TP: Test Problem.

TP	Mann-Whitney U rank sum test with FitPSO		TP	Mann-Whitney U rank sum test with FitPSO	
	ABC	PSO		ABC	PSO
f_1	+	+	f_{ϵ}	+	+
f_2	-	+	f_{1c}	+	+
f_3	-	+	f_{11}	+	+
f_4	+	+	f_{12}	+	-
f_5	=	=	f_{13}	+	+
f_6	+	+	f_{14}	-	+
f_7	+	+	$f_{1\epsilon}$	-	+
f_8	+	+			

Table 6.6: Acceleration Rate (AR) of FitPSO as compared to ABC and PSO, TP: Test Problems

TP	ABC	PSO
f_1	1.01209718	1.029561419
f_2	0.485698877	1.943903257
f_3	0.417512568	2.836490536
f_4	1.593022561	1.564618969
f_5	1.00019985	1.00025
f_6	1.340312037	2.916931789
f_7	4.450313185	1.502564445
f_8	9.502562531	2.50312359
f_9	11.22408988	2.293298266
f_{1c}	2.587992119	2.772072766
f_{11}	1.287559869	1.172694425
f_{12}	17.64270956	0.926317279
f_{13}	9.68765892	1.160970608
f_{14}	0.494260878	8.217110614
$f_{1\epsilon}$	0.528288507	1.69927293

Table 6.5 shows the results of the Mann-Whitney U rank sum test for the average number of function evaluations of 100 simulations. First we observe the significant difference by Mann-Whitney U rank sum test i.e., whether the two data sets are significantly different or not. If significant difference is not seen (i.e., the null hypothesis is accepted) then sign '=' appears and when significant difference is observed i.e., the null hypothesis is rejected then compare the average number of function evaluations. And we use signs '+' and '-' for the case where FitPSO takes less or more average number of function evaluations than the other algorithms, respectively. Therefore in Table 6.5, '+' shows that FitPSO is significantly better and '-' shows that FitPSO is significantly worse. As Table 6.5 includes 23 '+' signs out of 30 comparisons. Therefore, it can be concluded that the results of FitPSO is significantly cost effective than ABC and PSO over considered test problems.

Further, we compare the convergence speed of the considered algorithms by measuring the AFEs. A smaller AFEs means higher convergence speed. In order to minimize the effect of the stochastic nature of the algorithms, the reported function evaluations for each test problem is averaged over 100 runs. In order to compare convergence speeds, we use the acceleration rate (AR) which is defined as follows, based on the AFEs for the two algorithms ALGO and FitPSO:

$$AR = \frac{AFE_{ALGO}}{AFE_{FitPSO}}, \quad (6.8)$$

where, $ALGO \in \{ABC, PSO\}$ and $AR > 1$ means FitPSO is faster. In order to investigate

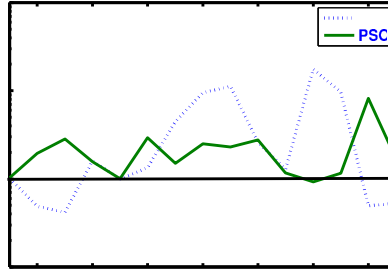


Figure 6.2: Acceleration Rate of FitPSO as compared to ABC and PSO

the AR of the proposed algorithm as compare to the considered algorithms, results of Table 6.2 are analyzed and the value of AR is calculated using equation (6.8). Table 6.6 shows a comparison between FitPSO and ABC, FitPSO and PSO in terms of AR . It is clear from the Table 6.6 that convergence speed of FitPSO is better than considered algorithms for most of the functions. The same claim can also be justified by visualizing the Figure 6.2. The convergence speed of FitPSO can also be judged through convergence figures 6.3(a)-6.3(d) that show the fitness movement through the iterations in a single run for selective functions f_6 , f_9 , f_{10} and f_{11} 3 respectively.

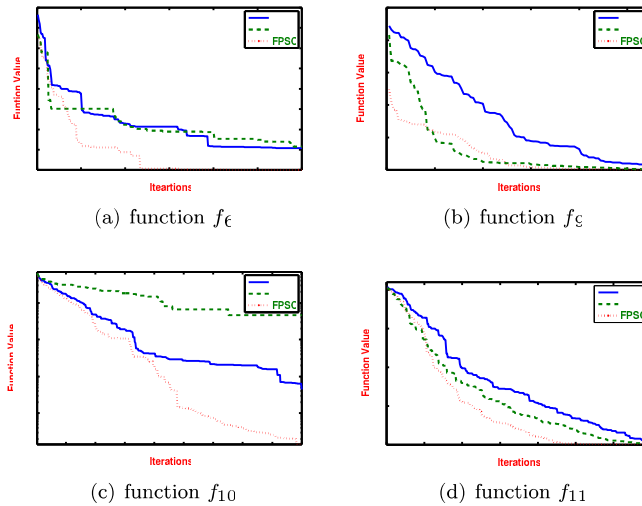


Figure 6.3: Convergence graph for ABC, PSO and FitPSO on test problems f_6 , f_9 , f_{10} , f_{11} .

6.5 Applications of FitPSO to Engineering Optimization Problems

Further, the proposed FitPSO algorithm is applied to solve three real world engineering optimization problems, namely Compression Spring [147, 148], Lennard-Jones [149], and Welded beam design optimization [150, 151]. Each of the engineering optimization problem is described as follows:

6.5.1 Compression Spring:

This problem minimizes the weight of a compression spring, subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. There are three design variables: the wire diameter x_1 , the mean coil diameter x_2 , and the number of active coils x_3 . This is a simplified version of a more difficult problem. The mathematical formulation of this problem is:

$$\begin{aligned} x_1 &\in \{1, \dots, 70\} \text{ granularity } 1 \\ x_2 &\in [0.6; 3] \\ x_3 &\in [0.207; 0.5] \text{ granularity } 0.001 \end{aligned}$$

and four constraints

$$\begin{aligned} g_1 &:= \frac{8C_f F_{max} x_2}{\pi x_1^3} - S \leq 0 \\ g_2 &:= l_f - l_{max} \leq 0 \\ g_3 &:= \sigma_p - \sigma_{pm} \leq 0 \\ g_4 &:= \sigma_w - \frac{F_{max} - F_p}{K} \leq 0 \end{aligned}$$

with

$$\begin{aligned} C_f &= 1 + 0.75 \frac{x_3}{x_2 - x_3} + 0.615 \frac{x_3}{x_2} \\ F_{max} &= 1000 \\ S &= 189000 \\ l_f &= \frac{F_{max}}{K} + 1.05(x_1 + 2)x_3 \\ l_{max} &= 14 \\ \sigma_p &= \frac{F_p}{K} \\ \sigma_{pm} &= 6 \\ F_p &= 300 \\ K &= 11.5 \times 10^6 \frac{x_3^4}{8x_1 x_2^3} \\ \sigma_w &= 1.25 \end{aligned}$$

and the function to be minimized is

$$E_1(\vec{X}) = \pi^2 \frac{x_2 x_3^2 (x_1 + 2)}{4}$$

The best known solution is $(7, 1.386599591, 0.292)$, which gives the fitness value $f^* = 2.6254$. Acceptable error for this problem is $1.0E - 04$.

6.5.2 Lennard-Jones :

The function to minimize is a kind of potential energy of a set of N atoms. The position X_i of the atom i has three coordinates, and therefore the dimension of the search space is $3N$. In practice, the coordinates of a point X are the concatenation of the ones of the X_i . In short, we can write $X = (X_1, X_2, \dots, X_N)$, and we have then

$$E_2(\vec{X}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N \left(\frac{1}{\|X_i - X_j\|^{2\alpha}} - \frac{1}{\|X_i - X_j\|^\alpha} \right)$$

In this study $N = 5$, $\alpha = 6$, and the search space is $[-2, 2]$ [149].

6.5.3 Welded beam design optimization problem:

The problem is to design a welded beam for minimum cost, subject to some constraints [150, 151]. The objective is to find the minimum fabricating cost of the welded beam subject to constraints on shear stress τ , bending stress σ , buckling load P_c , end deflection δ , and side constraint. There are four design variables: x_1 , x_2 , x_3 and x_4 . The mathematical formulation of the objective function is described as follows:

$$E_3(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

subject to:

$$\begin{aligned} g_1(\vec{x}) &= \tau(\vec{x}) - \tau_{max} \leq 0 \\ g_2(\vec{x}) &= \sigma(\vec{x}) - \sigma_{max} \leq 0 \\ g_3(\vec{x}) &= x_1 - x_4 \leq 0 \\ g_4(\vec{x}) &= \delta(\vec{x}) - \delta_{max} \leq 0 \\ g_5(\vec{x}) &= P - P_c(\vec{x}) \leq 0 \\ 0.125 &\leq x_1 \leq 5, \quad 0.1 \leq x_2, x_3 \leq 10 \text{ and } 0.1 \leq x_4 \leq 5 \end{aligned}$$

where

$$\begin{aligned} \tau(\vec{x}) &= \sqrt{\tau'^2 - \tau'\tau''\frac{x_2}{R} + \tau''^2}, \\ \tau' &= \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P(L + \frac{x_2}{2}), \\ R &= \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \\ J &= 2/\left(\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right), \\ \sigma(\vec{x}) &= \frac{6PL}{x_4x_3^2}, \delta(\vec{x}) = \frac{6PL^3}{Ex_4x_3^2}, \\ P_c(\vec{x}) &= \frac{4.013Ex_3x_4^3}{6L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \end{aligned}$$

$$P = 6000 \text{ lb}, L = 14 \text{ in.}, \delta_{max} = 0.25 \text{ in.}, \sigma_{max} = 30,000 \text{ psi},$$

$$\tau_{max} = 13600 \text{ psi}, E = 30 \times 10^6 \text{ psi}, G = 12 \times 10^6 \text{ psi}.$$

The best known solution is (0.205730, 3.470489, 9.036624, 0.205729), which gives the function value 1.724852. Acceptable error for this problem is $1.0E - 01$.

6.5.4 Experimental Results

To solve the constraint optimization problems (E_1 and E_3), a penalty function approach is used in the experiments. In this approach the search is modified by converting the original problem into an unconstrained optimization problem by adding a penalty term in case of constraints violation as shown below:

$$f(x) = f(x) + \beta$$

where, $f(x)$ is the original function value and β is the penalty term which is set to 10^3 .

Table 6.7 shows the experimental results of the considered algorithms on the engineering optimization problems. It is clear from Table 6.7 that the inclusion of new position update strategy in the PSO 2011, it performs better than the considered algorithms.

Table 6.7: Comparison of the results of test problems; TP: Test Problems

TP	Algorithm	SD	ME	AFE	SR
E_1	ABC	1.17E-02	1.36E-02	187602.32	10
	PSO	2.99E-04	4.86E-04	23789.5	100
	FitPSO	9.09E-04	5.30E-04	18159	100
E_2	ABC	1.16E-04	8.68E-04	71931.02	100
	PSO	3.38E-01	1.95E-01	139263.5	53
	FitPSO	1.48E-04	8.41E-04	62669.5	100
E_3	ABC	7.65E-02	2.40E-01	196524	3
	PSO	4.29E-03	9.43E-02	4012.5	100
	FitPSO	5.61E-03	9.36E-02	4826	100

Further, the algorithms are compared through SR , ME and AFE . On the basis of results shown in Table 6.7, it can be stated that the FitPSO performs better than the PSO 2011 and ABC algorithms for the considered engineering problems.

Further, the Mann-Whitney U rank sum test as mentioned in section 6.4.4 is applied to the considered algorithms for the average number of function evaluations of 100 simulations. The result of the test is shown in Table 6.8. It is clear from Table 6.8 that FitPSO is an efficient algorithms in the same category.

Table 6.8: Comparison based on mean function evaluations and the Mann-Whitney U rank sum test at a $\alpha = 0.05$ significance level ('+' indicates FitPSO is significantly better), TP: Test Problem.

Function	FitPSO Vs PSO	FitPSO Vs ABC
E_1	+	+
E_2	+	+
E_3	+	+

6.6 Conclusion

In this chapter, inspired from onlooker bee phase of ABC algorithm, a new solution search phase is introduced in PSO. In the proposed phase, solutions get chance to update themselves on the basis of probability which is a function of fitness. In this way, the high fit solutions get more chance to update themselves as compared to low fit solutions. Further, velocity update equation of PSO is also modified and the previous best learning component (cognitive component) is deleted from the velocity update equation. This modification helps the solutions to converge fast to the global optima. Through the experiments on 15 complex well known benchmark functions and three real world engineering optimization problems, it is shown that the proposed strategy is a competitive candidate in the field of swarm intelligence based algorithms.

CHAPTER 7

FitPSO FOR LARGE SCALE JOB SHOP SCHEDULING

Chapter 7

FitPSO for Large Scale Job Shop Scheduling

The large-scale job-shop scheduling problem (LSJSSP) is among one of the complex scheduling problems. Researchers are continuously working to deal with the LSJSSP through applying the various probabilistic algorithms which includes swarm intelligence based as well as the evolutionary algorithms even though not able to get the optimum results and it is still an interesting area. Therefore, in this chapter a recently developed non-deterministic algorithm namely fitness based particle swarm optimization (FitPSO) is applied to solve the LSJSSP problem instances. In the proposed solution, fitness based solution update strategy is incorporated with the PSO strategy to get the desired results. The obtained outcome is motivating and through results analysis, a confidence is achieved that the proposed FitPSO can be recommendation to solve the existing and the new LSJSSP instance. A fair comparative analysis is also presented which also supports the proposed recommendation.

The remainder of the chapter is structured as follows: FitPSO is explained in Section 7.2. Formulation of LSJSSP is discussed during the section 7.3. The entire process for solving LSJSSP using the proposed strategy is discussed in section 7.4. The implementation and experimental results are shown in section 7.5. Finally, the section 7.6 summarises the proposed work and suggests future research directions.

7.1 Introduction

Efficient scheduling is crucial for making the best use of available resources. In the domain of production management, the Large Scale Job-shop Scheduling Problem (LSJSSP) is a complicated combinatorial optimization problem. JSSP needs n jobs to be accomplished on m systems (machines). The system order for all jobs is fixed and varies depending on the jobs. The jobs are put in place in a non-preemptive manner, which means that while one job is running on one system, it cannot be disrupted by another. The primary goal of JSSP is to find an appropriate sequence scheme that reduces the time it takes for all jobs to be completed, which is referred to as makespan (MS). The goal is to minimize the makespan (MS) [102, 103].

The LSJSSP is one of the most important NP-hard problem. To solve LSJSSP, several deterministic conventional mathematical models and heuristic methods have been used. To small size LSJSSP cases, mathematical models have a successful solution in a reasonable

amount of time. [104]. The computational time increases exponentially as the size of the instances grows. So, for a larger scale LSJSSP, Non-conventional nature inspired algorithms (NIAs) are preferred alternatives [105]. The numerous processes found in nature are used to create NIAs. Swarm intelligence based algorithms (SIA) and evolutionary algorithms (EAs) are the two main types of NIAs. The design of SIA was influenced by the intellectual actions of creatures. Some state-of-art SIA are Artificial bee colony (ABC)[4], spider monkey optimization (SMO) [12], teaching learning based optimization (TLBO) [11] etc., EAs like differential evolution (DE) [106], genetic algorithm (GA) [107] etc., are based on biotic transformation like crossover, selection etc.

In recent years, NIAs are performing very well to solve physical world problems [108, 5]. In this series, many NIAs emerged well to solve LSJSSP such as genetic algorithm (GA) [109], particle swarm optimization (PSO) [110], hybrid biogeography based optimization (BBO) algorithm [111], hybrid differential evolution algorithm [112], multiple type individual enhancement PSO (MPSO) algorithm [113], classical LSJSSP [114], differential based harmony search algorithm with variable neighborhood search [115], biased random key genetic algorithm [116], new neighbored structure based algorithm [105], teaching learning based optimization (TLBO) algorithm [117], improved ABC (IABC) algorithm [118], discrete ABC (DABC) [14], best so far ABC [119], parallel ABC (pABC) algorithm [120], beer froth ABC [5] etc. In terms of computational time and solution efficiency, the obtained results are acceptable. At the same time, finding a solution for larger JSSP instances is a challenging task. These findings motivate researchers to continue their work in order to solve LSJSSP.

So in this chapter, a novel solution is proposed to solve the LSJSSP instances through the recent variant of PSO algorithm, namely fitness based particle swarm optimization algorithm (FitPSO). The FitPSO algorithm was developed by K. Sharma et. al. [152]. In the FitPSO algorithm, a fitness based solution search mechanism is incorporated in the standard PSO. As the FitPSO algorithm efficiently balances the diversification of the population during the solution search process [152]. In this chapter the FitPSO algorithm is applied to solve 105 LSJSSP instances. The results are analysed and compared to other important methods available in the literature. The obtained findings substantiate the validity of the proposed strategy.

7.2 Fitness Based PSO

PSO is an optimization technique which simulates the birds flocking behavior. PSO is a dynamic population of active, interactive agents with very little in the way of inherent intelligence. In PSO, whole group is called *swarm* and each individual is called *particle* which represents possible candidate's solution. The swarm finds food for its self through social learning by observing the behavior of nearby birds who appeared to be near the food source. Initially each particle is initialized within the search space randomly and keeps the information about its personal best position known as *pbest*, swarm best position known as *gbest* and current velocity V with which it is moving, in her memory. Based on these three values, each particle updates its position. In this manner, whole swarm moves in better direction while following collaborative trail and error method and converges to single best known solution.

For an D dimensional search space, the i^{th} particle of the swarm is represented by a D - dimensional vector, $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$. The velocity of this particle is represented

by another D-dimensional vector $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$. The previously best visited position of the i^{th} particle is denoted as $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$. g is the index of the best particle in the swarm. PSO swarm uses two equations for movement called *velocity update equation* and *position update equation*. The velocity of the i^{th} particle is updated using the velocity update equation given by equation (7.1) and the position is updated using equation (7.2).

$$v_{ij} = v_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (p_{gj} - x_{ij}) \quad (7.1)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (7.2)$$

where $j = 1, 2, \dots, D$ represents the dimension and $i = 1, 2, \dots, S$ represents the particle index. S is the size of the swarm and c_1 and c_2 are constants (usually $c_1 = c_2$), called cognitive and social scaling parameters respectively or simply acceleration coefficients. r_1 and r_2 are random numbers in the range $[0, 1]$ drawn from a uniform distribution.

The right hand side of velocity update equation (7.1) consists of three terms, the first term v_{ij} is the memory of the previous direction of movement which can be thought of as a momentum term and prevents the particle from drastically changing direction. The second term $c_1 r_1 (p_{ij} - x_{ij})$ is called cognitive component or persistence which draws particle back to their previous best situation and enables the local search in swarm. The last term $c_2 r_2 (p_{gj} - x_{ij})$ is known as social component which allows individuals to compare themselves to others in it's group and is responsible for global search. The Pseudo-code for Particle Swarm Optimization, is described as follows :

Algorithm 7.1 Particle Swarm Optimization Algorithm:

```

Initialize the parameters,  $w$ ,  $c_1$  and  $c_2$ ;
Initialize the particle positions and their velocities in the search space;
Evaluate fitness of individual particles;
Store gbest and pbest;
while stopping condition(s) not true do
  for each individual,  $X_i$  do
    for each dimension  $j$ ,  $x_{ij}$  do
      (i) Evaluate the velocity  $v_{ij}$  using (7.1);
      (ii) Evaluate the position  $x_{ij}$  using (7.2);
    end for
  end for
  Evaluate fitness of updated particles;
  Update gbest and pbest;
end while
Return the individual with the best fitness as the solution;

```

Based on the neighborhood size, initially two versions of PSO algorithm were presented in literature namely, global version of PSO which is the original PSO (PSO-G) and the local version of PSO (PSO-L)[39]. The only difference between PSO-G and PSO-L is that the term p_g in social component in velocity update equation (7.1). For PSO-G, it refers the best particle of whole swarm while for PSO-L it represents the best particle of the individual's neighborhood. The social network employed by the PSO-G reflects the star topology which offers a faster convergence but it is very likely to converge prematurely. While PSO-L uses a ring social network topology where smaller neighborhoods are defined for each particle. It can be easily observed that due to the less particle inter connectivity in PSO-L, it is less susceptible to be trapped in local minima but at the cost of slow conver-

gence. In general, PSO-G performs better for unimodal problems and PSO-L for multimodal problems.

However the standard PSO has the capability to get a good solution at a significantly faster rate but, when it is compared to other optimization techniques, it is weak to refine the optimum solution, mainly due to less diversity in later search [153]. On the different side, problem-based tuning of parameters is also important in PSO, to get optimum solution accurately and efficiently[143]. In standard PSO velocity update equation (7.1) contains three terms. The first term has the global search capability, the second and third terms are the particles cognitive and social information sharing capability respectively. More cognitive capability force particle to move towards personal best position fast and more social information force particle to move towards global best position fast. It can be seen from (7.1), the movement of swarm towards optimum solution is guided by the acceleration factor c_1 and c_2 . Therefore, acceleration coefficient c_1 and c_2 should be tuned carefully to get the desired solution.

Kennedy and Eberhart [88] explained that more value of the cognitive component compared to the social component, results in excessive wandering of individuals through the search space while on the other hand more value of the social component may results that particles will converge prematurely toward a local optimum. These two component play important role for balancing the exploration and exploitation capabilities of PSO. Therefore, in this chapter two modifications are proposed for improving the solution search efficiency of PSO.

1. Velocity update equation of PSO is modified as follows, here w is the inertia weight:

$$v_{ij} = w \times v_{ij} + c \times r(p_{gj} - x_{ij}) \quad (7.3)$$

It is clear from equation (7.3) that the $Pbest$ component (cognitive component) is removed $\{c_1 r_1(p_{ij} - x_{ij})\}$ from the velocity update equation of PSO. Now the magnitude of velocity of each individual will depend on its distance from the current global best solution. Therefore, this strategy will improve the exploitation capability of PSO.

2. A new position update process, which is inspired from the Artificial Bee Colony (ABC) algorithm's onlooker bee phase [144] is incorporated with PSO. In employed bee phase of ABC, all the employed bees search the food source and calculate their fitness using equation (7.4):

$$\text{if } f_i \geq 0, \text{ then } fitness_i = 1/(1 + f_i), \text{ else } fitness_i = 1 + abs(f_i). \quad (7.4)$$

and then in the onlooker bee phase, onlooker bees analyze the available information and select a solution with a probability, $prob_i$, related to its fitness. The probability $prob_i$ may be calculated using equation (7.5):

$$prob_i(G) = \frac{0.9 \times fitness_i(G)}{maxfit(G)} + 0.1, \quad (7.5)$$

where G is the iteration counter, $fitness_i(G)$ is the fitness value of i^{th} solution and $maxfit(G)$ is the maximum fitness of the solutions in G^{th} iteration. Position update equation of ABC is shown in equation (7.6):

$$y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (7.6)$$

where $k \in \{1, 2, \dots, S\}$ and $j \in \{1, 2, \dots, D\}$ are randomly chosen indices, k must be different from i , ϕ_{ij} is a random number between $[-1, 1]$ and x_{kj} is a random individual in the current population. In the basic ABC, at any given time, only one dimension is updated in employed or onlooker bee phase. In onlooker bee phase this update takes place based on a probability which is a function of fitness.

The proposed position update strategy is incorporated with PSO and the newly developed algorithm is named as Fitness based PSO (FitPSO). In FitPSO, Algorithm 7.2 is applied after basic PSO operators. The insertion of Algorithm 7.2 makes FitPSO more capable of exploitation in the better search regions. It is expected because in FitPSO after applying basic PSO operators, better candidate solutions are offered more chances to update themselves than worse candidates. The pseudo-code of the proposed position update strategy which works after PSO operators is shown in Algorithm 7.2.

Algorithm 7.2 Fitness based Position Update Phase:

```

for each individual,  $x_i$  do
  if  $prob_i > rand(0, 1)$  then
     $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$ ,
    Calculate fitness of  $\vec{y}_i$ ,
    Apply greedy selection between  $\vec{y}_i$  and  $\vec{x}_i$ ,
  end if
end for

```

The Pseudo-code for the proposed FitPSO algorithm is shown in Algorithm 7.3.

7.3 Job shop scheduling problem organisation

The LSJSSP can be interpreted in following manner: There are a set of n jobs to be processed using m machines. To complete the execution, each job has to be passed through all the m systems in a given predefined sequence. Each job consists of total m operations. To perform operations a job uses one of the machine. When any of the job is executing on any machine it cannot be interrupted by other jobs. The total number of operations are $m \times n$ that are scheduled on m systems [123].

The objective of the LSJSSP is to minimize the total completion time for all the jobs i.e. makespan (MS). Mathematically the problem is stated as :

$$\text{Minimize } MS_{max} \quad (7.7)$$

where, $MS_{max} = \max(MS_1, MS_2, MS_3, MS_4, \dots, MS_n)$. $MS_1, MS_2, MS_3, MS_4, \dots, MS_n$ are the completion time for all the n jobs. Followings are the constraints for LSJSSP [116]:

- Each system can process at most one operation at a time.
- The completion time of any operation must be a positive integer.
- Precedence relationships among the different jobs must be satisfied.

Algorithm 7.3 Fitness based Particle Swarm Optimization(FitPSO):

```
Initialize the parameters,  $w$ , and  $c$  and  $S$ ;  
Initialize the particle positions and their velocities in the search space;  
Evaluate fitness of individual particles;  
Store the gbest solution;  
while stopping condition(s) not true do  
  for each individual,  $X_i$  do  
    for each dimension  $j$  of  $x_{ij}$  do  
      (i) Evaluate the velocity  $v_{ij}$  using (7.3);  
      (ii) Evaluate the position  $x_{ij}$  using (7.2);  
    end for  
  end for  
  Evaluate fitness of updated particles;  
  Update gbest solution;  
  /*****Fitness based position update phase in FitPSO *****/  
   $t = 1, i = 1$  /***  $t$  counts number of updates ***/  
  while  $t \leq S$  do  
    if  $prob_i > rand(0, 1)$  then  
      /*****  $prob_i$  is the probability of an individual  $x_i$  described by equation (7.5)*****/  
  
       $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}),$   
       $\{k, j \text{ is randomly selected index}\}$   
      Calculate fitness of  $\vec{y}_i$ ;  
      Apply greedy selection between  $\vec{y}_i$  and  $\vec{x}_i$ .;  
       $t = t + 1$   
    end if  
     $i = i + 1$   
    if  $i > S$  then  
       $i = 1$   
    end if  
  end while  
end while  
Return the individual with the best fitness as the solution;
```

7.4 FitPSO for LSJSSP

The FitPSO algorithm is used to solve LSJSSP instances, and the whole method is detailed here. Since LSJSSP is a discrete optimization problem, a solution in the proposed algorithm is a discrete valued vector (representing a potential operation scheduling list). The reordering of jobs for FitPSO is used to estimate each solution in the search field. To generate the discrete valued sequence from a continuous valued vector we have used random key encoding (RKE) scheme [124].

In RKE encoding scheme, first a continuous valued vector is sorted in an ascending order using an integer series from 1 to $n \times m$, where n represents the total number of jobs and m shows the total number of available systems. As each job has to go through m systems for completing its execution so further transformation from this integer sequence is performed using (Integer value mod $n + 1$). The integer series is transformed to operation order sequence using this transformation, and each job index has m occurrences. Figure 7.1 depicts the transformation of a continuous valued vector into a discrete valued vector, followed by an operation scheduling sequence. Our goal is to find an operation sequencing list (a vector of discrete values) that decreases the makespan value. The goal is to figure out a series of operations that reduces the overall time it takes to complete all of the jobs. The detailed procedure is described in the subsequent steps:

Continuous valued solution	0.9	0.6	0.8	0.2	0.5	0.3
Decoded as	6	4	5	1	3	2
Operation sequence	1	2	3	2	1	3

Figure 7.1: Random Key (RKE) Encoding Scheme

Step 1:

The parameters of the proposed FitPSO algorithm namely, cognitive, social scaling parameters (c_1 , c_2), Inertia Weight (w), total members of population (solution agents), and total number of iterations are initialized. Each solution agent is initialized in the search space in a uniformly distributed way. As all the initialized sources are continuous in nature so RKE scheme is used to generate the corresponding discrete valued operation sequence. Now the MS value (objective value) for each operation sequence is calculated.

Step 2:

As the step 2 and 3 are iterative steps, the solutions refined themselves in these steps to get the optimum solution. In step2, all the solution agents update themselves using the standard PSO algorithm. The updated solution agent is in continuing form, so again RKE encoding scheme is applied to alter this continuous valued solutions in to corresponding discrete operation sequence list. The MS value for this operation sequence is computed. The *pbest* and *gbest* solutions are updated on the basis of the MS value.

Step 3:

In this step, the probability for all the solution agents are assessed using the equation 7.5. This probability will help to decide that which solution is high fit than the other solutions in the swarm. The solution agents are chosen and updated as per the equation 7.5. Again the solution agent is updated based upon the information obtained from the neighbouring solution agents. To obtain the corresponding operation sequence, RKE scheme is applied on the produced continuous valued solution agent. The *MS* value is computed from the generated operation sequence and the *pbest* and *gbest* solutions are updated.

Step 4:

In this step, the best solution found so far is memorized (*gbest* solution). Thus obtained solution is termed is the optimum solution generated by the FitPSO.

The pseudo-code of the designed approach for LSJSSP is shown in Algorithm 7.4.

Algorithm 7.4 FitPSO algorithm for LSJSSP

```
Parameter Initialization
Total solution agents = TSA
D (Dimension) =  $m \times n$ 
Total generation count = MGN
CurrentIndex=1
Step 1: Random initialization of the Solution members in the search space
Conversion of continuous valued solution agents into an operation sequence (discrete valued solutions) to deal LSJSSP
using RKE scheme
MS value Computation for every operation sequence.
While (CurrentIndex  $\leq$  MGN) dc
    • Step 2: PSO stage:
    • for each individual,  $X_i$  dc
      for each dimension  $j$  of  $x_{ij}$  do
        (i) Evaluate the velocity  $v_{ij}$  using (7.3);
        (ii) Evaluate the position  $x_{ij}$  using (7.2);
      end for
    • end for
    • Obtain the new discrete operation sequence from the recently revised continuing solution agents using RKE scheme.
    • MS value computation for recently produced operation sequence.
    • Evaluate fitness of updated particles;
    • Update the respective pbest solutions;
    • Update gbest solution;
    • Step 3: FitPSO stage:
    • Probability  $prob_i$  computation using equation 7.5 for each solution agent.
    •  $t = 1, i = 1$  /***  $t$  counts number of updates ***/
    • while  $t \leq S$  do
      if  $prob_i > rand(0, 1)$  then
         $y_{ij} = x_{ij} + \phi_{ij}(x_{i5} - x_{kj})$ 
        { $k, j$  is randomly selected index}
        Calculate fitness of  $y_i$ ;
        Apply greedy selection between  $y_i$  and  $x_i$ .
        Obtain the new discrete operation sequence from the recently revised continuing solution agents using RKE
        scheme
        MS value computation for recently produced operation sequence.
        Evaluate fitness of updated particles
        Update the respective pbest solutions
         $t = t + 1$ 
      end if
       $i = i + 1$ 
      if  $i > S$  then
         $i = 1$ 
      end if
    • end while
    • Update gbest solution;
    • Step 4: Memorize the best solution found so far.
    • CurrentIndex=CurrentIndex+1.
  end while
```

Output the best solution

7.5 Implementation and experimental results

To prove the effectiveness of FitPSO algorithm, it is applied on LSJSSP instances. Following 105 LSJSSP instances are considered for experimentation [125, 126, 127].

- 15 SWV instances
- 50 TA instances
- 40 DMU instances

To attain the least MS value for all these 105 LSJSSP instances is the main goal. The experimental setting is listed as below:

1. Number of run =10
2. Number of maximum iteration =2000
3. Number of solution agents TSA =50
4. Dimension $D = \text{Number of systems} \times \text{Number of jobs}$
5. Inertia weight $w = 0.8$,
6. Acceleration coefficients $c = c_1 = c_2 = 0.5 + \log 2$ (for PSO)[145],

The parametric ambience for the FitPSO approach and the other considered approaches are kept same in terms of swarm size and maximum number of iterations to carry out an equitable comparison.

The reported results of FitPSO are compared with the following state-of-art algorithms available in the literature:

- Biased random key genetic algorithm (BRKGA-JSP) [116]
- A guided tabu search for LSJSSP (NKPR) [128]
- Teaching learning based optimization method (TLBO) [129]
- Differential based harmony search (DHS) algorithm [115]
- A tabu search to solve LSJSSP (TS) [130]
- An advanced tabu search algorithm for LSJSSP (i-TSAB) [131]
- AlgFix [132]
- A tabu search/simulated annealing algorithm for LSJSSP (TS/SA) [133]
- Global equilibrium search technique (GES) [134]

The obtained results for the above three instances are represented in Tables 7.1 to 7.3. These tables list the name of the instance, its size, the lower bound (LB), the upper bound (UB) for the best known solution (BKS), the BKS obtained by FitPSO approach, and BKS value obtained from the compared algorithms. The obtained results for all the instances demonstrate that the proposed FitPSO is superior approach in reference to MS value during assessment with other considered approaches.

Table 7.1: Comparison in terms of best MS value for SMV instances

Instance	Size	LB	UB	FitPSO	FABC	BRKGA-JSP	TS/SA	TS
SWV01	20 × 10	1407	1407	1407	1407	1407	1412	-
SWV02	20 × 10	1475	1475	1475	1475	1475	1475	-
SWV03	20 × 10	1369	1398	1392	1395	1398	1398	-
SWV04	20 × 10	1450	1474	1468	1465	1470	1470	-
SWV05	20 × 10	1424	1424	1424	1424	1425	1425	-
SWV06	20 × 15	1591	1678	1662	1674	1675	1679	-
SWV07	20 × 15	1446	1600	1548	1572	1594	1603	-
SWV08	20 × 15	1640	1763	1756	1762	1755	1756	-
SWV09	20 × 15	1604	1661	1654	1650	1656	1661	-
SWV10	20 × 15	1631	1767	1729	1736	1743	1754	-
SWV11	50 × 10	2983	2983	2983	2983	2983	-	2983
SWV12	50 × 10	2972	2979	2974	2975	2979	-	2979
SWV13	50 × 10	3104	3104	3104	3104	3104	-	3104
SWV14	50 × 10	2968	2968	2968	2968	2968	-	2968
SWV15	50 × 10	2885	2886	2885	2885	2901	-	2886

Further to analyse the outcomes, average relative percentage error (RPE) is also calculated and compared as tabulated in Table 7.4. The value of RPE is computed (with respect to the UB value of an instance) as per demonstrated in equation 7.8.

$$RPE = 100 \times (BKS_{algo} - UB)/UB \quad (7.8)$$

Here, BKS_{algo} represents the MS value obtained using the considered approaches. The attained outcomes of Table 7.4 demonstrate the significant improvement in the average RPE which assures the authenticity of the introduced approach.

Table 7.2: Comparison in terms of best MS value for TA instances

Instance	Size	LB	UB	FitPSO	FABC	BRKGA-JSP	GES	ALPfix	I-TSAB	TS//SA	DHS	TLBO	NKPR
TA01	15 × 15	1231	1231	1231	1231	1231	1231	1231	-	1231	1321	1526	1485
TA02	15 × 15	1244	1244	1244	1244	1244	1244	1244	-	1244	1313	1538	1476
TA03	15 × 15	1218	1218	1218	1218	1218	1218	1218	-	1218	1327	1594	1470
TA04	15 × 15	1175	1175	1175	1175	1175	1175	1175	-	1175	1285	1550	1519
TA05	15 × 15	1224	1224	1224	1224	1224	1224	1224	-	1224	1315	1551	1381
TA06	15 × 15	1238	1238	1238	1238	1238	1238	1238	-	1238	1346	1538	1517
TA07	15 × 15	1227	1227	1227	1227	1227	1228	1228	-	1228	1322	1546	1460
TA08	15 × 15	1217	1217	1217	1217	1217	1217	1217	-	1217	1304	1534	1446
TA09	15 × 15	1274	1274	1274	1274	1274	1274	1274	-	1274	1386	1614	1551
TA10	15 × 15	1241	1241	1241	1241	1241	1241	1241	-	1241	1334	1542	1365
TA11	20 × 15	1323	1357	1348	1355	1357	1357	1358	1361	1359	1520	1876	1687
TA12	20 × 15	1351	1367	1367	1367	1367	1367	1367	-	1371	1563	1856	1770
TA13	20 × 15	1282	1342	1342	1342	1342	1344	1342	-	1342	1513	1849	1713
TA14	20 × 15	1345	1345	1345	1345	1345	1345	1345	-	1345	1477	1780	1748
TA15	20 × 15	1304	1339	1334	1337	1339	1339	1339	-	1339	1557	1929	1788
TA16	20 × 15	1302	1360	1360	1360	1360	1360	1360	-	1360	1543	1852	1716
TA17	20 × 15	1462	1462	1462	1462	1462	1469	1473	1462	1464	1607	1941	1781
TA18	20 × 15	1369	1396	1396	1396	1396	1401	1396	-	1399	1601	1817	1776
TA19	20 × 15	1297	1332	1330	1331	1332	1332	1332	1335	1335	1524	1842	1722
TA20	20 × 15	1318	1348	1348	1348	1348	1348	1348	1351	1350	1554	1902	1710
TA21	20 × 20	1539	1643	1637	1642	1642	1647	1643	1644	1644	1854	2399	2165
TA22	20 × 20	1511	1600	1600	1600	1600	1602	1600	1600	1600	1852	2241	2126
TA23	20 × 20	1472	1557	1557	1557	1557	1558	1557	1557	1560	1755	2210	2145
TA24	20 × 20	1602	1646	1646	1646	1646	1653	1646	1647	1646	1829	2241	2173
TA25	20 × 20	1504	1595	1592	1594	1595	1596	1595	1595	1597	1792	2324	2117
TA26	20 × 20	1539	1645	1639	1641	1643	1647	1645	1647	1647	1863	2299	2206
TA27	20 × 20	1616	1680	1680	1680	1680	1685	1686	1680	1680	1905	2436	2194
TA28	20 × 20	1591	1603	1599	1600	1603	1614	1613	1614	1603	1819	2333	2100
TA29	20 × 20	1514	1625	1625	1625	1625	1625	1625	1627	1627	1853	2280	2146
TA30	20 × 20	1473	1584	1584	1584	1584	1584	1584	1584	1584	1812	2247	2103
TA31	30 × 15	1764	1764	1764	1764	1764	1764	1766	-	1764	2037	2528	2382
TA32	30 × 15	1774	1790	1779	1781	1785	1793	1790	-	1795	2106	2591	2482
TA33	30 × 15	1778	1791	1791	1791	1791	1791	1791	-	1796	2091	2685	2511
TA34	30 × 15	1828	1829	1830	1832	1829	1832	1832	1829	1831	2089	2508	2480
TA35	30 × 15	2007	2007	2007	2007	2007	2007	2007	-	2007	2139	2509	2512
TA36	30 × 15	1819	1819	1819	1819	1819	1819	1819	-	1819	2086	2705	2395
TA37	30 × 15	1771	1771	1738	1728	1771	1779	1784	1778	1778	2067	2512	2436
TA38	30 × 15	1673	1673	1673	1673	1673	1673	1673	-	1778	1980	2488	2250
TA39	30 × 15	1795	1795	1795	1795	1795	1795	1795	-	1795	2010	2439	2501
TA40	30 × 15	1631	1673	1668	1665	1669	1680	1679	1674	1676	1986	2455	2380
TA41	30 × 20	1859	2006	2006	2006	2008	2008	2022	-	2018	-	-	-
TA42	30 × 20	1867	1945	1931	1867	1937	1956	1953	1956	1953	-	-	-
TA43	30 × 20	1809	1814	1828	1836	1852	1870	1869	1859	1858	-	-	-
TA44	30 × 20	1927	1983	1983	1983	1983	1991	1992	1984	1983	-	-	-
TA45	30 × 20	1940	2000	2000	2000	2000	2004	2011	2000	2000	-	-	-
TA46	30 × 20	1940	2008	2002	2000	2004	2011	2011	2021	2010	-	-	-
TA47	30 × 20	1789	1897	1892	1892	1894	1903	1902	1903	1903	-	-	-
TA48	30 × 20	1912	1945	1935	1940	1943	1962	1962	1953	1903	-	-	-
TA49	30 × 20	1915	1966	1960	1962	1964	1969	1974	-	1967	-	-	-
TA50	30 × 20	1807	1925	1925	1925	1925	1931	1927	1928	1931	-	-	-

Table 7.3: Comparison in terms of best MS value for DMU instances

Instance	Size	LB	UB	FitPSO	FABC	BRKGA-JSP	TS	GES	i-TSAB	AlgFix
DMU01	20 × 15	2501	2563	2563	2563	2563	2566	2566	2517	2563
DMU02	20 × 15	2651	2706	2701	2704	2706	2711	2706	2715	2706
DMU03	20 × 15	2731	2731	2731	2731	2731	-	2731	-	2731
DMU04	20 × 15	2601	2669	2669	2669	2669	-	2669	-	2669
DMU05	20 × 15	2749	2749	2749	2749	2749	-	2749	-	2749
DMU06	20 × 20	2834	3244	3241	3242	3244	3254	3250	3265	3244
DMU07	20 × 20	2677	3046	3045	3045	3046	-	3053	-	3046
DMU08	20 × 20	2901	3188	3188	3188	3188	3191	3197	3199	3188
DMU09	20 × 20	2739	3092	3090	3091	3092	-	3092	3094	3096
DMU10	20 × 20	2716	2984	2982	2982	2984	-	2984	2985	2984
DMU11	30 × 15	3395	3453	3453	3454	3445	3455	3453	3470	3455
DMU12	30 × 15	3481	3516	3512	3512	3513	3516	3518	3519	3522
DMU13	30 × 15	3681	3681	3681	3681	3681	3681	3697	3698	3687
DMU14	30 × 15	3394	3394	3394	3394	3394	-	3394	3394	3394
DMU15	30 × 15	3332	3343	3343	3343	3343	-	3343	-	3343
DMU16	30 × 20	3726	3759	3748	3748	3751	3759	3781	3787	3772
DMU17	30 × 20	3697	3836	3830	3828	3830	3842	3848	3854	3836
DMU18	30 × 20	3844	3846	3844	3844	3844	3846	3849	3854	3852
DMU19	30 × 20	3650	3775	3768	3769	3770	3784	3807	3823	3775
DMU20	30 × 20	3604	3712	3712	3712	3716	3716	3739	3740	3712
DMU21	40 × 15	4380	4380	4380	4380	4380	-	4380	-	4380
DMU22	40 × 15	4325	4725	4725	4725	4725	-	4725	-	4725
DMU23	40 × 15	4668	4668	4668	4668	4668	-	4668	-	4668
DMU24	40 × 15	4648	4648	4648	4648	4648	-	4648	-	4648
DMU25	40 × 15	4164	4164	4164	4164	4164	-	4164	-	4164
DMU26	40 × 20	4647	4647	4647	4647	4647	4647	4667	4679	4688
DMU27	40 × 20	4848	4848	4848	4848	4848	-	4848	4848	4848
DMU28	40 × 20	4692	4692	4692	4692	4692	-	4692	-	4692
DMU29	40 × 20	4691	4691	4691	4691	4691	-	4691	4691	4691
DMU30	40 × 20	4732	4732	4732	4732	4732	-	4732	4732	4749
DMU31	50 × 15	5640	5640	5640	5640	5640	-	5640	-	5640
DMU32	50 × 15	5927	5927	5927	5927	5927	-	5927	-	5927
DMU33	50 × 15	5728	5728	5728	5728	5728	-	5728	-	5728
DMU34	50 × 15	5385	5385	5385	5385	5385	-	5385	-	5385
DMU35	50 × 15	5635	5635	5635	5635	5635	-	5635	-	5635
DMU36	50 × 20	5621	5621	5621	5621	5621	-	5621	-	5621
DMU37	50 × 20	5851	5851	5851	5851	5851	-	5851	5851	5851
DMU38	50 × 20	5713	5713	5713	5713	5713	-	5713	-	5713
DMU39	50 × 20	5747	5747	5747	5747	5747	-	5747	-	5747
DMU40	50 × 20	5577	5577	5577	5577	5577	-	5577	-	5577

Table 7.4: Comparison based upon Average RPE

Approach	Instances Solved	ARPE (%)	FitPSO(%)	Improvement (%)
TA instances				
BRKGA-JSP [116]	50	0.015	-0.131	0.146
GES [134]	50	0.218	-0.131	0.349
AlgFix [132]	50	0.556	-0.131	0.687
i-TSAB [131]	25	0.269	-0.21	0.479
TS/SA [133]	50	0.156	-0.131	0.287
DHS [115]	40	12.412	-0.13	12.542
TLBO [129]	40	36.883	-0.13	37.013
NKPR [128]	40	28.951	-0.13	29.081
FABC	50	-0.093	-0.131	0.038
DMU Instances				
BRKGA-JSP [116]	40	-0.021	-0.031	0.01
TS [130]	13	0.097	-0.083	0.18
GES [134]	40	0.107	-0.031	0.138
i-TSAB [131]	20	0.24	-0.06	0.3
AlgFix [132]	40	0.056	-0.031	0.087
FABC	40	-0.027	-0.031	0.004
SWU Instances				
BRKGA-JSP [116]	15	-0.156	-0.559	0.403
TS/SA [133]	10	-0.073	-0.818	0.745
TS [130]	5	0	-0.04	0.04
FABC	15	-0.364	-0.559	0.195

7.6 Conclusion

This article proposed a solution to solve 105 large scale instances of job shop scheduling problem (LSJSSP) using an efficient fully informed artificial bee colony (FitPSO). In FitPSO algorithm, to balances the diversification in the swarm during the solution search process, a fitness based strategy is incorporated in the standard PSO algorithm. The MS time is used as an evaluation criterion in the LSJSSP. The results are analysed and compared to cutting-edge techniques proposed by a number of researchers. According to the results of the experiments, the proposed solution gives better solution. In Future, some more performance metrics may be considered for experimentation.

CHAPTER 8

CONCLUSION AND FUTURE WORK

Chapter 8

Conclusion and Future Work

This chapter forms the concluding part of the Thesis and also proposes some suggestions to extend the present work.. This chapter provides the conclusion and future work for the research work carried out in this thesis. Section 8.1 discusses about conclusion for proposed approaches. Section 8.2 lists some future direction to extend this work.

8.1 Conclusion

The primary goals of this work are to produce some improvements in Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO) algorithms for the numerical optimization . For improvements in ABC and PSO, the main focus is on creating diversity in the search process and balancing the exploration and exploitation capabilities of both the algorithms. The significant research contributions of this thesis are three new variant of nature inspired algorithms. Here, two new variant of ABC and one unique variant of PSO proposed and deployed for solving real world problem. First variant is limaçon-based local search in ABCA, and the designed strategy is named Limaçon inspired ABC (LABC) algorithm. Then, the exploitation capability of the LABC strategy is tested over 18 complex benchmark optimization problems. Finally, the test results are compared with similar state-of-art algorithms, and statistical analysis shows the LABC can be considered a practical variant of the ABC algorithms to solve the complex optimization problems.

The second variant is Fully Informed Artificial Bee Colony (FABC) algorithm. FABC is applied to solve the 105 LSJSSP instances. The FABC algorithm is developed by taking inspiration from the GABC algorithm position update process. In the FABC, the onlooker bee process of the Artificial Bee Colony (ABC) algorithm is modified and designed such that the new position of the solution search agent is obtained while learning from all the nearby agents. The results obtained by the FABC are compared with the state-of-art algorithms. The results analysis shows that the proposed approach to solving LSJSSP is competitive in the field of SIA.

Third variant is fitness-based particle swarm optimization (FitPSO). FitPSO is applied to solve the LSJSSP problem instances. Here, a fitness-based solution update strategy is incorporated with the PSO strategy to get the desired results in the proposed solution. The obtained outcome is motivating, and through results analysis, confidence is achieved that the proposed FitPSO can be a recommendation to solve the existing and the new LSJSSP instance. A fair comparative analysis is also presented, which also supports the proposed recommendation.

8.2 Future Work

In the future, the proposed ABC variants like FABC, LABC and PSO variant (FitPSO) may be applied to solve engineering optimization problems. The applications of variants of ABC and PSO can be continued to unexplored areas of other scheduling problems like dynamic job shop scheduling, ow shop scheduling, batch scheduling, and more complex model of real-world problems. Further, more performance measure parameters may also be included in the engineering optimization problems. The problem-specific efficiency of the proposed variants of ABC and PSO may also be evaluated in the future. The developed variants may also be improved for better accuracy, reliability, and efficiency for benchmark functions as well as for real-world optimization problems.

SUMMARY

Summary

Swarm intelligence (SI) techniques are based on the aggregate cognitive behaviour of numerous species found in nature. These SI-based methods perform well in determining the answer to real-life complicated optimization issues. The artificial bee colony (ABC) method, for example, is one of the most efficient SI-based theorems. The ABC is based on a mathematical modelling of honey bee food gathering behaviour. However, ABC and PSO, like other SI-based methods, suffer from early convergence, stagnation, and the inability to unfold the real results for optimization problems.

The basic ABC and PSO algorithm is revised in this thesis utilising many techniques to solve current limitations and improve the execution ability of the ABC and PSO algorithm. Two novel ABC variants, Limaçon-inspired ABC and Fully Informed ABC, are created in this study, as well as one new PSO variation, Fitness-based PSO. Furthermore, these variations are used to address a real-world difficult optimization issue, namely the scheduling problem for a work shop (JSSP).

To begin, ABCA incorporates an influential local search (LS) approach inspired by the limaçon curve, and the resulting strategy is known as the Limaçon influenced ABC (LABC) algorithm. The LABC strategy's exploitation capacity is next put to the test using 18 difficult benchmark optimization tasks. Finally, the test results are compared to similar state-of-the-art algorithms, and statistical analysis demonstrates that the LABC may be used to tackle complicated optimization problems as a realistic variation of the ABC algorithms.

In addition, the Fully Informed Artificial Bee Colony (FABC) method is created using the position update technique of the fully informed particle swarm optimization algorithm as inspiration. The Artificial Bee Colony (ABC) algorithm's observer bee process is updated and developed in the FABC so that the new position of the solution search agent is acquired while learning from all surrounding agents. The ABC incorporates a unique learning process in which people change their locations by learning from all neighbouring solutions as well as the best solution in the swarm. In the following experiment, the FABC is used to solve 105 LSJSSP cases. The FABC's findings are compared to those of state-of-the-art algorithms. The findings demonstrate that the proposed LSJSSP solution is competitive in the field of swarm intelligence based algorithms (SIA).

In addition, to address the LSJSSP problem cases, a fitness-based particle swarm optimization (FitPSO) is created and deployed. To achieve the desired outcomes in the suggested approach, a fitness-based solution update method is used with the PSO technique. The acquired result is encouraging, and via examination of the data, it is clear that the suggested FitPSO may be used to address both existing and new LSJSSP instances. In addition, a fair comparative study is given, which supports the recommended recommendation.

BIBLIOGRAPHY

Bibliography

- [1] R. Storn and K. Price. Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. *International computer science institute - Publications - TR*, 1995.
- [2] D.E. Goldberg. Genetic algorithms in search, optimization, and machine learning. 1989.
- [3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford University Press, USA, 1999.
- [4] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.
- [5] Nirmala Sharma, Harish Sharma, and Ajay Sharma. Beer froth artificial bee colony algorithm for job-shop scheduling problem. *Applied Soft Computing*, 68:507–524, 2018.
- [6] Dervis Karaboga, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, 42(1):21–57, 2014.
- [7] Jagdish Chand Bansal, Harish Sharma, and Shimpi Singh Jadon. Artificial bee colony algorithm: a survey. *International Journal of Advanced Intelligence Paradigms*, 5(1-2):123–159, 2013.
- [8] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108–132, 2009.
- [9] Guopu Zhu and Sam Kwong. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation*, 217(7):3166–3173, 2010.
- [10] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.
- [11] Ravipudi V Rao, Vimal J Savsani, and DP Vakharia. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43(3):303–315, 2011.
- [12] Jagdish Chand Bansal, Harish Sharma, Shimpi Singh Jadon, and Maurice Clerc. Spider monkey optimization algorithm for numerical optimization. *Memetic computing*, 6(1):31–47, 2014.
- [13] Bahriye Akay and Dervis Karaboga. A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, 192:120–142, 2012.
- [14] Minghao Yin, Xiangtao Li, and Junping Zhou. An efficient job shop scheduling algorithm based on artificial bee colony. *Scientific Research and Essays*, 6(12):2578–2596, 2011.

- [15] Hao Liu, Guiyan Ding, and Huafei Sun. An improved opposition-based disruption operator in gravitational search algorithm. In *Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on*, volume 2, pages 123–126. IEEE, 2012.
- [16] Yuko Aratsu, Kazunori Mizuno, Hitoshi Sasaki, and Seiichi Nishihara. Experimental evaluation of artificial bee colony with greedy scouts for constraint satisfaction problems. In *Technologies and Applications of Artificial Intelligence (TAAI), 2013 Conference on*, pages 134–139. IEEE, 2013.
- [17] Jagdish Chand Bansal, Harish Sharma, KV Arya, Kusum Deep, and Millie Pant. Self-adaptive artificial bee colony. *Optimization*, 63(10):1513–1532, 2014.
- [18] Alkin Yurtkuran and Erdal Emel. An enhanced artificial bee colony algorithm with solution acceptance rule and probabilistic multisearch. *Computational intelligence and neuroscience*, 2016:41, 2016.
- [19] Xianneng Li and Guangfei Yang. Artificial bee colony algorithm with memory. *Applied Soft Computing*, 41:362–372, 2016.
- [20] Wan-li Xiang, Xue-lei Meng, Yin-zhen Li, Rui-chun He, and Mei-qing An. An improved artificial bee colony algorithm based on the gravity model. *Information Sciences*, 429:49–71, 2018.
- [21] Dražen Bajer and Bruno Zorić. An effective refined artificial bee colony algorithm for numerical optimisation. *Information Sciences*, 504:221–275, 2019.
- [22] Clodomir J Santana Jr, Mariana Macedo, Hugo Siqueira, Anu Gokhale, and Carmelo JA Bastos-Filho. A novel binary artificial bee colony algorithm. *Future Generation Computer Systems*, 98:180–196, 2019.
- [23] Ali R Yildiz. A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing. *Applied Soft Computing*, 13(5):2906–2912, 2013.
- [24] Reza Zanjirani Farahani, Ashkan Hassani, Seyyed Mostafa Mousavi, and Mohammad Bakhshayeshi Baygi. A hybrid artificial bee colony for disruption in a hierarchical maximal covering location problem. *Computers & Industrial Engineering*, 75:129–141, 2014.
- [25] Jun-qing Li and Quan-ke Pan. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Information Sciences*, 316:487–502, 2015.
- [26] Wei-feng Gao, Ling-ling Huang, Jue Wang, San-yang Liu, and Chuan-dong Qin. Enhanced artificial bee colony algorithm through differential evolution. *Applied Soft Computing*, 48:137–150, 2016.
- [27] Shimpi Singh Jadon, Ritu Tiwari, Harish Sharma, and Jagdish Chand Bansal. Hybrid artificial bee colony algorithm with differential evolution. *Applied Soft Computing*, 58:11–24, 2017.
- [28] Tao Meng, Quan-Ke Pan, and Hong-Yan Sang. A hybrid artificial bee colony algorithm for a flexible job shop scheduling problem with overlapping in operations. *International Journal of Production Research*, 56(16):5278–5292, 2018.
- [29] Shuwei Wang, Xiuping Guo, and Jia Liu. An efficient hybrid artificial bee colony algorithm for disassembly line balancing problem with sequence-dependent part removal times. *Engineering Optimization*, 51(11):1920–1937, 2019.
- [30] Dervis Karaboga and Ebubekir Kaya. Training anfis by using an adaptive and hybrid artificial bee colony algorithm (aabc) for the identification of nonlinear static systems. *Arabian Journal for Science and Engineering*, 44(4):3531–3547, 2019.

- [31] Jun-Qing Li, Mei-Xian Song, Ling Wang, Pei-Yong Duan, Yu-Yan Han, Hong-Yan Sang, and Quan-Ke Pan. Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs. *IEEE transactions on cybernetics*, 2019.
- [32] Fei Kang, Junjie Li, Zhenyue Ma, and Haojin Li. Artificial bee colony algorithm with local search for numerical optimization. *Journal of Software*, 6(3):490–497, 2011.
- [33] Jagdish Chand Bansal, Harish Sharma, KV Arya, and Atulya Nagar. Memetic search in artificial bee colony algorithm. *Soft Computing*, 17(10):1911–1928, 2013.
- [34] Harish Sharma, Jagdish Chand Bansal, and KV Arya. Power law-based local search in artificial bee colony. *International Journal of Artificial Intelligence and Soft Computing*, 4(2-3):164–194, 2014.
- [35] Harish Sharma, Jagdish Chand Bansal, KV Arya, and Xin-She Yang. Lévy flight artificial bee colony algorithm. *International Journal of Systems Science*, 47(11):2652–2670, 2016.
- [36] Gürcan Yavuz and Doğan Aydın. Improved self-adaptive search equation-based artificial bee colony algorithm with competitive local search strategy. *Swarm and Evolutionary Computation*, 51:100582, 2019.
- [37] Dervis Karaboga and Beyza Gorkemli. A quick artificial bee colony (qabc) algorithm and its performance on optimization problems. *Applied Soft Computing*, 23:227–238, 2014.
- [38] Neha Pathak, Manuj Mishra, and Shiv Pratap Singh Kushwah. Improved local search based modified abc algorithm for tsp problem. In *2017 4th International Conference on Electronics and Communication Systems (ICECS)*, pages 173–178. IEEE, 2017.
- [39] J. Kennedy R.C. Eberhart. A new optimizer using particle swarm theory, proceedings of 6th symp. micro machine and human science. In *Proceedings of 6th symp. Micro Machine and Human Science*, pages 39–43. Nagoya, Japan, 1995.
- [40] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73. IEEE, 1998.
- [41] You-Min Jau, Kuo-Lan Su, Chia-Ju Wu, and Jin-Tsong Jeng. Modified quantum-behaved particle swarm optimization for parameters estimation of generalized nonlinear multi-regressions model based on choquet integral with outliers. *Applied Mathematics and Computation*, 221:282–295, 2013.
- [42] Mostafa Jamalipour, Reza Sayareh, Morteza Gharib, Farrokh Khoshahval, and Mahmood Reza Karimi. Quantum behaved particle swarm optimization with differential mutation operator applied to wwer-1000 in-core fuel management optimization. *Annals of Nuclear Energy*, 54:134–140, 2013.
- [43] Ahmad Bagheri, Hamed Mohammadi Peyhani, and Mohsen Akbari. Financial forecasting using anfis networks with quantum-behaved particle swarm optimization. *Expert Systems with Applications*, 41(14):6235–6250, 2014.
- [44] Deyu Tang, Yongming Cai, Jie Zhao, and Yun Xue. A quantum-behaved particle swarm optimization with memetic algorithm and memory for continuous non-linear large scale problems. *Information Sciences*, 289:162–189, 2014.
- [45] Elnaz Davoodi, Mehrdad Tarafdar Hagh, and Saeid Ghassem Zadeh. A hybrid improved quantum-behaved particle swarm optimization–simplex method (iqpsos) to solve power system load flow problems. *Applied Soft Computing*, 21:171–179, 2014.

- [46] James Kennedy. Bare bones particle swarms. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 80–87. IEEE, 2003.
- [47] Haibo Zhang, Devid Desfrees Kennedy, Gade Pandu Rangaiah, and Adrián Bonilla-Petriciolet. Novel bare-bones particle swarm optimization and its performance for modeling vapor–liquid equilibrium data. *Fluid Phase Equilibria*, 301(1):33–45, 2011.
- [48] Junqi Zhang, NI Lina, YAO Jing, WANG Wei, and TANG Zheng. Adaptive bare bones particle swarm inspired by cloud model. *IEICE TRANSACTIONS on Information and Systems*, 94(8):1527–1538, 2011.
- [49] Haibo Zhang, Jorge Adán Fernández-Vargas, Gade Pandu Rangaiah, Adrián Bonilla-Petriciolet, and Juan Gabriel Segovia-Hernández. Evaluation of integrated differential evolution and unified bare-bones particle swarm optimization for phase equilibrium and stability problems. *Fluid Phase Equilibria*, 310(1):129–141, 2011.
- [50] Li-Yeh Chuang, Sheng-Wei Tsai, and Cheng-Hong Yang. Chaotic catfish particle swarm optimization for solving global numerical optimization problems. *Applied Mathematics and Computation*, 217(16):6900–6916, 2011.
- [51] Yudong Zhang and Lenan Wu. Crop classification by forward neural network with adaptive chaotic particle swarm optimization. *Sensors*, 11(5):4721–4743, 2011.
- [52] Yongshou Dai, Yuqin Wei, Jian Chen, Yanan Zhang, and Jinjie Ding. Seismic wavelet estimation based on adaptive chaotic embedded particle swarm optimization algorithm. In *Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on*, volume 2, pages 57–60. IEEE, 2012.
- [53] Chaoshun Li, Jianzhong Zhou, Pangao Kou, and Jian Xiao. A novel chaotic particle swarm optimization based fuzzy clustering algorithm. *Neurocomputing*, 83:98–109, 2012.
- [54] Yau-Tarng Juang, Shen-Lung Tung, and Hung-Chih Chiu. Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions. *Information Sciences*, 181(20):4539–4549, 2011.
- [55] Alireza Alfi and Mohammad-Mehdi Fateh. Intelligent identification and control using improved fuzzy particle swarm optimization. *Expert Systems with Applications*, 38(10):12312–12317, 2011.
- [56] Wen-an Yang, Yu Guo, and Wen-he Liao. Optimization of multi-pass face milling using a fuzzy particle swarm optimization algorithm. *The International Journal of Advanced Manufacturing Technology*, 54(1):45–57, 2011.
- [57] Habib Dhahri and Adel M Alimi. Opposition-based particle swarm optimization for the design of beta basis function neural network. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.
- [58] Hui Wang, Zhijian Wu, Shahryar Rahnamayan, Yong Liu, and Mario Ventresca. Enhancing particle swarm optimization using generalized opposition-based learning. *Information Sciences*, 181(20):4699–4714, 2011.
- [59] Na Dong, Chun-Ho Wu, Wai-Hung Ip, Zeng-Qiang Chen, Ching-Yuen Chan, and Kai-Leung Yung. An opposition-based chaotic ga/pso hybrid algorithm and its application in circle detection. *Computers & Mathematics with Applications*, 64(6):1886–1902, 2012.
- [60] Wei-feng Gao, San-yang Liu, and Ling-ling Huang. Particle swarm optimization with chaotic opposition-based population initialization and stochastic search technique. *Communications in Nonlinear Science and Numerical Simulation*, 17(11):4316–4327, 2012.

- [61] Li-Yeh Chuang, Sheng-Wei Tsai, and Cheng-Hong Yang. Improved binary particle swarm optimization using catfish effect for feature selection. *Expert Systems with Applications*, 38(10):12699–12707, 2011.
- [62] Ruifeng Shi and Xiangjie Liu. A hybrid improved particle swarm optimization based on dynamic parameters control and metropolis accept rule strategy. In *Genetic and Evolutionary Computing, 2009. WGECC'09. 3rd International Conference on*, pages 649–653. IEEE, 2009.
- [63] Jianhuan Zhang, Yingxin Wang, Rui Wang, and Guolian Hou. Bidding strategy based on adaptive particle swarm optimization for electricity market. In *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pages 3207–3210. IEEE, 2010.
- [64] RJ Kuo and CW Hong. Integration of genetic algorithm and particle swarm optimization for investment portfolio optimization. *Appl. Math*, 7(6):2397–2408, 2013.
- [65] Wen-Chin Chen and Denni Kurniawan. Process parameters optimization for multiple quality characteristics in plastic injection molding using taguchi method, bpnn, ga, and hybrid pso-ga. *International journal of precision engineering and manufacturing*, 15(8):1583–1593, 2014.
- [66] M Nazir, A Majid-Mirza, and S Ali-Khan. Pso-ga based optimized feature selection using facial and clothing information for gender classification. *Journal of applied research and technology*, 12(1):145–152, 2014.
- [67] Hongzhong Tang, Yewei Xiao, Huixian Huang, and Xuefeng Guo. A novel dynamic particle swarm optimization algorithm based on improved artificial immune network. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 103–106. IEEE, 2010.
- [68] Yudong Zhang, Yan Jun, Geng Wei, and Lenan Wu. Find multi-objective paths in stochastic networks via chaotic immune pso. *Expert Systems with Applications*, 37(3):1911–1919, 2010.
- [69] Shyi-Ming Chen and Chih-Yao Chien. Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications*, 38(12):14439–14450, 2011.
- [70] Renbin Xiao, Weiming Chen, and Tinggui Chen. Modeling of ant colony’s labor division for the multi-project scheduling problem and its solution by pso. *Journal of Computational and Theoretical Nanoscience*, 9(2):223–232, 2012.
- [71] Mohammed El-Abd. Testing a particle swarm optimization and artificial bee colony hybrid algorithm on the cec13 benchmarks. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2215–2220. IEEE, 2013.
- [72] Tarun Kumar Sharma, Millie Pant, and Ajith Abraham. Blend of local and global variant of pso in abc. In *Nature and Biologically Inspired Computing (NaBIC), 2013 World Congress on*, pages 113–119. IEEE, 2013.
- [73] Mustafa Servet KIRAN and Mesut GÜNDÜZ. A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems. *Applied Soft Computing*, 13(4):2188–2203, 2013.
- [74] Guido Maione and Antonio Punzi. Combining differential evolution and particle swarm optimization to tune and realize fractional-order controllers. *Mathematical and computer modelling of dynamical systems*, 19(3):277–299, 2013.
- [75] Yangguang Fu, Mingyue Ding, Chengping Zhou, and Hanping Hu. Route planning for unmanned aerial vehicle (uav) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(6):1451–1465, 2013.

- [76] Bahriye Akay and Dervis Karaboga. Artificial bee colony algorithm variants on constrained optimization. *An International Journal of Optimization and Control*, 7(1):98, 2017.
- [77] Xianpeng Wang and Lixin Tang. A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking. *Applied Soft Computing*, 12(2):652–662, 2012.
- [78] Alberto García-Villoria and Rafael Pastor. Introducing dynamic diversity into a discrete particle swarm optimization. *Computers & Operations Research*, 36(3):951–966, 2009.
- [79] Dervis Karaboga and Bahriye Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied soft computing*, 8(1):687–697, 2008.
- [80] Harish Sharma, Jagdish Chand Bansal, and KV Arya. Opposition based lévy flight artificial bee colony. *Memetic Computing*, 5(3):213–227, 2013.
- [81] Shimpi Singh Jadon, Jagdish Chand Bansal, Ritu Tiwari, and Harish Sharma. Expedited artificial bee colony algorithm. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving*, pages 787–800. Springer, 2014.
- [82] Ajay Sharma, Harish Sharma, Annapurna Bhargava, and Nirmala Sharma. Fibonacci series based local search in spider monkey optimisation for transmission expansion planning. *Int. J. of Swarm Intelligence*, pages Accepted, in press.
- [83] Ajay Sharma, Harish Sharma, Annapurna Bhargava, Nirmala Sharma, and Jagdish Chand Bansal. Optimal placement and sizing of capacitor using limaçon inspired spider monkey optimization algorithm. *Memetic Computing*, 9(4):311–331, 2017.
- [84] Geerat J Vermeij. *A natural history of shells*. Princeton University Press, 1995.
- [85] Anan Banharnsakun, Tiranee Achalakul, and Booncharoen Sirinaovakul. The best-so-far selection in artificial bee colony algorithm. *Applied Soft Computing*, 11(2):2888–2901, 2011.
- [86] Nirmala Sharma, Harish Sharma, Ajay Sharma, and Jagdish Chand Bansal. Black hole artificial bee colony algorithm. In *International Conference on Swarm, Evolutionary, and Memetic Computing*, pages 214–221. Springer, 2015.
- [87] M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2. IEEE, 1999.
- [88] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.
- [89] K.V. Price, R.M. Storn, and J.A. Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Verlag, 2005.
- [90] J. Vesterstrom and R. Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1980–1987. IEEE, 2004.
- [91] W. Gao and S. Liu. A modified artificial bee colony algorithm. *Computers & Operations Research*, 2011.
- [92] D. Karaboga and B. Akay. A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied Soft Computing*, 2010.

- [93] R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: simpler, maybe better. *Evolutionary Computation, IEEE Transactions on*, 8(3):204–210, 2004.
- [94] B. Akay and D. Karaboga. A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, doi:10.1016/j.ins.2010.07.015, 2010.
- [95] K. Diwold, A. Aderhold, A. Scheidler, and M. Middendorf. Performance evaluation of artificial bee colony optimization and new selection schemes. *Memetic Computing*, pages 1–14, 2011.
- [96] M. El-Abd. Performance assessment of foraging algorithms vs. evolutionary algorithms. *Information Sciences*, 182(1):243–263, 2011.
- [97] S. Rahnamayan, H.R. Tizhoosh, and M.M.A. Salama. Opposition-based differential evolution. *Evolutionary Computation, IEEE Transactions on*, 12(1):64–79, 2008.
- [98] J.C. Bansal and H. Sharma. Cognitive learning in differential evolution and its application to model order reduction problem for single-input single-output systems. *Memetic Computing*, pages 1–21, 2012.
- [99] D.F. Williamson, R.A. Parker, and J.S. Kendrick. The box plot: a simple visual method to interpret data. *Annals of internal medicine*, 110(11):916, 1989.
- [100] H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 18(1):50–60, 1947.
- [101] Thakur M. Deep, K. A new crossover operator for real coded genetic algorithms. *Applied Mathematics and Computation*, 188(1):895–911, 2007.
- [102] Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.
- [103] Shyam Sundar, Ponnuthurai N Suganthan, Chua Tay Jin, Cai Tian Xiang, and Chong Chin Soon. A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint. *Soft Computing*, 21(5):1193–1202, 2017.
- [104] Joseph Adams, Egon Balas, and Daniel Zawack. The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3):391–401, 1988.
- [105] Liang Gao, Xinyu Li, Xiaoyu Wen, Chao Lu, and Feng Wen. A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem. *Computers & Industrial Engineering*, 88:417–429, 2015.
- [106] Rainer Storn and Kenneth Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.
- [107] David E Goldberg and John Henry Holland. Genetic algorithms and machine learning. 1988.
- [108] Nirmala Sharma, Harish Sharma, and Ajay Sharma. An effective solution for large scale single machine total weighted tardiness problem using lunar cycle inspired artificial bee colony algorithm. *IEEE/ACM transactions on computational biology and bioinformatics*, 2019.
- [109] José Fernando Gonçalves, Jorge José de Magalhães Mendes, and Maurício GC Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, 167(1):77–95, 2005.
- [110] M Clerc and J Kennedy. Standard pso 2011. *Particle Swarm Central Site [online]* <http://www.particleswarm.info>, 2011.

- [111] Xiaohua Wang and Haibin Duan. A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Computers & Industrial Engineering*, 73:96–114, 2014.
- [112] Bin Qian, Ling Wang, Rong Hu, Wan-Liang Wang, De-Xian Huang, and Xiong Wang. A hybrid differential evolution method for permutation flow-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 38(7-8):757–777, 2008.
- [113] Tsung-Lieh Lin, Shi-Jinn Horng, Tzong-Wann Kao, Yuan-Hsin Chen, Ray-Shine Run, Rong-Jian Chen, Jui-Lin Lai, and I-Hong Kuo. An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, 37(3):2629–2636, 2010.
- [114] Scott R Schultz, Thom J Hodgson, and Russell E King. On solving the classic job shop makespan problem by minimizing lmax. *Raleigh, NC: Department of Industrial Engineering, North Carolina State University*, 2004.
- [115] Fuqing Zhao, Shuo Qin, Guoqiang Yang, Weimin Ma, Chuck Zhang, and Houbin Song. A differential-based harmony search algorithm with variable neighborhood search for job shop scheduling problem and its runtime analysis. *IEEE Access*, 6:76313–76330, 2018.
- [116] JOSÉ FERNANDO Gonçalves and MAURICIO GC Resende. A biased random-key genetic algorithm for job-shop scheduling. *AT&T Labs Research Technical Report*, 46:253–271, 2011.
- [117] HS Keesari and RV Rao. Optimization of job shop scheduling problems using teaching-learning-based optimization algorithm. *Opsearch*, 51(4):545–561, 2014.
- [118] Bao Zhen Yao, Cheng Yong Yang, Juan Juan Hu, Guo Dong Yin, and Bo Yu. An improved artificial bee colony algorithm for job shop problem. In *Applied Mechanics and Materials*, volume 26, pages 657–660. Trans Tech Publ, 2010.
- [119] Anan Banharnsakun, Booncharoen Sirinaovakul, and Tiranee Achalakul. Job shop scheduling with the best-so-far abc. *Engineering Applications of Artificial Intelligence*, 25(3):583–593, 2012.
- [120] Leila Asadzadeh. A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy. *Computers & Industrial Engineering*, 2016.
- [121] Ajay Sharma, Harish Sharma, Annapurna Bhargava, and Nirmala Sharma. Optimal power flow analysis using lvy flight spider monkey optimisation algorithm. *International Journal of Artificial Intelligence and Soft Computing*, 5(4):320–352, 2016.
- [122] R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: simpler, maybe better. *Evolutionary Computation, IEEE Transactions on*, 8(3):204–210, 2004.
- [123] Renato EN Moraes. Job shop scheduling. *Universidade Federal Fluminense*.
- [124] A Tamilarasi and T Anantha Kumar. Job-shop scheduling using random key encoding scheme particle swarm optimization. *International Journal of Computational Intelligence Research*, 6(1):33–43, 2010.
- [125] Eric Taillard. Benchmarks for basic scheduling problems. *european journal of operational research*, 64(2):278–285, 1993.
- [126] Ebru Demirkol, Sanjay Mehta, and Reha Uzsoy. Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109(1):137–141, 1998.
- [127] Robert H Storer, S David Wu, and Renzo Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Management science*, 38(10):1495–1509, 1992.

- [128] Mohammad Mahdi Nasiri and Farhad Kianfar. A guided tabu search/path relinking algorithm for the job shop problem. *The International Journal of Advanced Manufacturing Technology*, 58(9-12):1105–1113, 2012.
- [129] R Venkata Rao, Vimal J Savsani, and DP Vakharia. Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. *Information Sciences*, 183(1):1–15, 2012.
- [130] ChaoYong Zhang, PeiGen Li, ZaiLin Guan, and YunQing Rao. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, 34(11):3229–3242, 2007.
- [131] Eugeniusz Nowicki and Czesław Smutnicki. An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*, 8(2):145–159, 2005.
- [132] Panos M Pardalos, Oleg V Shylo, and Alkis Vazacopoulos. Solving job shop scheduling problems utilizing the properties of backbone and big valley. *Computational Optimization and Applications*, 47(1):61–76, 2010.
- [133] Chao Yong Zhang, PeiGen Li, YunQing Rao, and ZaiLin Guan. A very fast ts/sa algorithm for the job shop scheduling problem. *Computers & Operations Research*, 35(1):282–294, 2008.
- [134] Panos M Pardalos and Oleg V Shylo. An algorithm for the job shop scheduling problem based on global equilibrium search techniques. *Computational Management Science*, 3(4):331–348, 2006.
- [135] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. Ieee, 1995.
- [136] J.J. Liang, AK Qin, P.N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evolutionary Computation, IEEE Transactions on*, 10(3):281–295, 2006.
- [137] G. Ciuprina, D. Ioan, and I. Munteanu. Use of intelligent-particle swarm optimization in electromagnetics. *Magnetics, IEEE Transactions on*, 38(2):1037–1040, 2002.
- [138] X. D. Li and A. P. Engelbrecht. Particle swarm optimization: An introduction and its recent developments. *Genetic Evol. Comput. Conf.*, pages 3391–3414, 2007.
- [139] Asanga Ratnaweera, Saman K Halgamuge, and Harry C Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *Evolutionary Computation, IEEE Transactions on*, 8(3):240–255, 2004.
- [140] Z.H. Zhan, J. Zhang, Y. Li, and H.S.H. Chung. Adaptive particle swarm optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(6):1362–1381, 2009.
- [141] W. Zhang, H. Li, Z. Zhang, and H. Wang. The selection of acceleration factors for improving stability of particle swarm optimization. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 1, pages 376–380. IEEE, 2008.
- [142] W. Gai-yun and H. Dong-xue. Particle swarm optimization based on self-adaptive acceleration factors. In *Genetic and Evolutionary Computing, 2009. WGECC'09. 3rd International Conference on*, pages 637–640. IEEE, 2009.
- [143] Y. Shi and R. Eberhart. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, pages 591–600. Springer, 1998.
- [144] D. Karaboga and B. Basturk. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. *Foundations of Fuzzy Logic and Soft Computing*, 4529, no. 1:789–798, 2007.

- [145] J.J. Kim, S.Y. Park, and J.J. Lee. Experience repository based particle swarm optimization for evolutionary robotics. In *ICCAS-SICE, 2009*, pages 2540–2544. IEEE, 2009.
- [146] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.
- [147] G.C. Onwubolu and BV Babu. *New optimization techniques in engineering*. Springer Verlag, 2004.
- [148] E. Sandgren. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, 112:223, 1990.
- [149] M. Clerc. List based pso for real problems. [http://clerc.maurice.free.fr /pso /ListBasedPSO /ListBasedPSO28PSOsite29.pdf](http://clerc.maurice.free.fr/pso/ListBasedPSO/ListBasedPSO28PSOsite29.pdf), 16 July 2012.
- [150] KM Ragsdell and DT Phillips. Optimal design of a class of welded structures using geometric programming. *ASME Journal of Engineering for Industries*, 98(3):1021–1025, 1976.
- [151] M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2):1567–1579, 2007.
- [152] Kavita Sharma, Varsha Chhamunya, PC Gupta, Harish Sharma, and Jagdish Chand Bansal. Fitness based particle swarm optimization. *International Journal of System Assurance Engineering and Management*, 6(3):319–329, 2015.
- [153] P. Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In *Evolutionary Programming VII*, pages 601–610. Springer, 1998.

PUBLICATIONS

List of Publications

Research Publications

1. **Kavita Sharma**, P. C. Gupta, and Nirmala Sharma, “Limaon inspired artificial bee colony algorithm for numerical optimization”, *Evolutionary Intelligence* (2020): 1-9. <https://doi.org/10.1007/s12065-020-00430-8>. Springer Nature Switzerland, **Scopus indexed**.
2. **Kavita Sharma**, P. C. Gupta, “Fully Informed ABC Algorithm for Large Scale Job Shop Scheduling problem” (Submission code: IJIEI-327309) for the *International Journal of Intelligent Engineering Informatics (IJIEI)*. (2021) (Accepted)
3. **Kavita Sharma**, P. C. Gupta, “Fitness based PSO for Large Scale Job Shop Scheduling problem”, 2nd Congress on Intelligent Systems (CIS 2021) proceedings in Scopus Indexed Springer Book Series, “Lecture Notes on Data Engineering and Communications Technologies”. (Accepted)
4. **Kavita Sharma**, P C Gupta, Harish Sharma, “Fully Informed Artificial Bee Colony Algorithm, *Journal of Experimental and Theoretical Artificial Intelligence*: 1-14 Taylor and Francis. (**Published**)
5. **Kavita Sharma**, Gupta, P. C., Sharma, H., “Fitness based particle swarm optimization. *International Journal of System Assurance Engineering and Management*”, 6(3), 319-329. (**Published**)

Special Issue | [Published: 18 June 2020](#)

Limaçon inspired artificial bee colony algorithm for numerical optimization

[Kavita Sharma](#), [P. C. Gupta](#) & [Nirmala Sharma](#) 

[Evolutionary Intelligence](#) (2020)

51 Accesses | **1** Citations | [Metrics](#)

Abstract

The artificial bee colony algorithm (ABCA) has established itself as a signature algorithm in the area of swarm intelligence based algorithms. The hybridization of the local search technique enhances the exploitation capability in the search process of the global optimization strategies. In this article, an effective local search technique that is designed by taking inspiration by Limaçon curve, is incorporated in ABCA and the designed strategy is named Limaçon inspired ABC (LABC) algorithm. The exploitation capability of the LABC strategy is tested over 18 complex benchmark optimization problems. The test results are compared with similar state-of-art algorithms and statistical analysis shows the LABC can be considered an effective variant of the ABC algorithms to solve the complex optimization problems.

This is a preview of subscription content, [access via your institution](#).

Limaçon inspired artificial bee colony algorithm for numerical optimization

Kavita Sharma^a, P C Gupta^b, and Nirmala Sharma^{c*}

^{a,b}*University of Kota, Kota, India*, ^c*Rajasthan Technical University, Kota, India*

(Received 00 Month 20XX; final version received 00 Month 20XX)

The artificial bee colony algorithm (ABCA) has established itself as a signature algorithm in the area of swarm intelligence based algorithms. The hybridization of the local search technique enhances the exploitation capability in the search process of the global optimization strategies. In this article, an effective local search (LS) technique that is designed by taking inspiration by limaçon curve, is incorporated in ABCA and the designed strategy is named Limaçon inspired ABC (LABC) algorithm. The exploitation capability of the LABC strategy is tested over 18 complex benchmark optimization problems. The test results are compared with similar state-of-art algorithms and statistical analysis shows the LABC can be considered an effective variant of the ABC algorithms to solve the complex optimization problems.

Keywords: Artificial bee colony algorithm; Swarm intelligence; Limaçon curve; Local search, Optimization

1. Introduction

The swarm intelligence (SI) derived techniques are impressive methods to deal with complex optimization problems. The SI based strategies do rely upon the social intellectual conduct of natural species. Artificial bee colony algorithm (ABCA) is a prominent SI technique, developed by taking inspiration from the intelligent communication of honey bees (Karaboga & Basturk, 2008; Bansal, Sharma, & Jadon, 2013). In the past, the ABCA has been applied to many real-world complex optimization problems but it also suffers the common problems of SI based optimization techniques like stopping to move at the global optima and skipping the true solution due to high explorative nature of the solution search process (Karaboga & Akay, 2009; H. Sharma, Bansal, & Arya, 2013; Jadon, Bansal, Tiwari, & Sharma, 2014; Bansal, Sharma, Arya, Deep, & Pant, 2014; H. Sharma, Bansal, & Arya, 2014).

The local search (LS) hybridization with the global search optimization algorithms boosts the exploitation capability of the global search algorithms which further decreases the chance of skipping the true solution. So, to enhance the local searchability in the ABCA solution search process, in this article a limaçon arc inspired LS (LLS) hybridized with ABCA. The hybridized algorithm is titled as Limaçon inspired artificial bee colony (LABC) algorithm. The solution searchability of the proposed LABC is evaluated through numerous experimentations in form of accurateness, reliability, and consistency.

*Corresponding author. Email: nsharma@rtu.ac.in,

2. Artificial bee colony algorithm

Artificial bee colony algorithm (ABCA) is a significant strategy in the field of SI centered strategies. ABCA was proposed by D. Karaboga in the year 2005 (Karaboga, 2005). It is motivated by the food foraging activities of the honey bees. There are three types of honey bees in the colony of bees that are employed honey bees, onlooker honey bees, and scout honey bees. At the initial stage, employed honey bees go for searching the food sources. They collect the nectar with all the associated information and return to the hive. They transfer knowledge associated with the food sources with the onlooker honey bees staying at the hive. Scout honey bees search the food sources randomly depending upon the internal motivation. Like other meta heuristic approaches, ABCA is also an iterative process that consists of following cycles of four stages:

2.1. Initialization stage:

The initialization of all the N solutions take place during this stage in the D dimensional space as per the lower bound and upper bound of all the decision parameters of the optimization problem. The initialization for w_i (i^{th} candidate solution where $i = 1, \dots, N$) is as per the following equation 1.

$$w_{ij} = w_{lowj} + rand[0, 1](w_{upperj} - w_{lowj}) \quad (1)$$

where, w_{lowj} and w_{upperj} respectively represent bounds of w_i in j^{th} direction further, $rand[0, 1]$ is an evenly scattered arbitrary number in the bound 0 to 1.

2.2. Employed honey bee stage:

During the employed honey bee stage, each solution of the search space is updated as per the equation 2. The solution is modified based upon the information obtained from any arbitrary solution of the search region. The fitness value of the newly produced solution (nectar amount) is calculated (Akay & Karaboga, 2012). If the fitness value of the newly produced solution is greater than that of the previous solution, the new solution is selected for the next generation and the old one is discarded. Equation 2 represent position update equation for the i^{th} candidate solution.

$$w_{ij} = w_{ij} + \phi_{ij}(w_{ij} - w_{neighj}) \quad (2)$$

Where, $neigh \in \{1, 2, \dots, N\}$ and $j \in \{1, 2, \dots, D\}$ are arbitrarily selected indices, $neigh$ must be distinct from i , and ϕ_{ij} is an arbitrary number between $[-1, 1]$.

2.3. Onlooker honey bee stage:

When all the employed honey bees complete their task, they share all the information regarding the nectar with the onlooker honey bees. Onlooker honey bees analyze the information received from employed honey bees and select a food source based on a probability value $Prob_i$. The probability $Prob_i$ is a function of fitness and computed

using the equation 3.

$$Prob_i = \frac{Fitness_i}{\sum_{i=1}^N Fitness_i} \quad (3)$$

where, $Fitness_i$ is representing the fitness value for the i^{th} solution. Same as the previous phase the position of a solution is modified based on its probability value. The fitness value of the newly produced solution is computed. A greedy selection mechanism is applied to select a solution for the next generation. The higher fit solution is selected for the next generation.

2.4. Scout honey bee stage:

Scout honey bee stage is initiated when a food source is not modifying its position up to a threshold limit. In this case, the discarded food source and the associated bee to the discarded food source becomes scout honey bee. Now that food source is re-initialized in the search space. Let the discarded food source is w_i and $j \in \{1, 2, \dots, D\}$ then food source is re-initialized as per the equation 4:

$$w_{ij} = w_{lowj} + rand[0, 1](w_{upperj} - w_{lowj}) \quad (4)$$

3. Limaçon inspired local search strategy and it's incorporation to artificial bee colony algorithm

As per the reported literature, the ABCA experiences the issue of premature convergence, stagnation, and sometimes unable to discover the true solution of an optimization problem (Zhu & Kwong, 2010). In the past, the solution searchability of ABCA has been enhanced by introducing LS techniques, by introducing control parameters, or by hybridization with other search strategies (A. Sharma, Sharma, Bhargava, & Sharma, n.d.; N. Sharma, Sharma, & Sharma, 2018).

Recently, an LS approach derived from the Limaçon arc equation namely, Limaçon inspired local search (LLS) is reported in literature (A. Sharma, Sharma, Bhargava, Sharma, & Bansal, 2017). In LLS, a limaçon is presented as a roulette traced by the locus of a point located on the periphery of a circle when this circle rolls around the periphery of another circle of equal radius. It can also be represented as the roulette created when a circle rolls around another circle with half its radius so that the tiny circle is inside the bigger circle. This mathematical curve is inspired by the natural species limaçon more commonly known as a snail. The vertical and horizontal axis contour equations of limaçon are demonstrated in equations 5 and 6 respectively (Vermeij, 1995).

$$z = p \pm q \sin \phi \quad (5)$$

$$z = p \pm q \cos \phi \quad (6)$$

Here, z represents the distance of the limaçon from the starting point, two constants are represented as p and q , the angle of revolution is denoted by ϕ . The value of q is responsible for transient phases of the curve, $q = 0, 1, \text{ and } > p$ represents a circular, cardioid, and a noose curve respectively.

! On **Wednesday 28 July 07:00 – 15:00 GMT**, we'll be carrying out some essential maintenance on Taylor & Francis Online. You'll still be able to search, browse and read our articles, where access rights already apply, but registration, purchasing, activation of tokens, eprints and other features of Your Account will be unavailable during this scheduled work.

Home ▶ All Journals ▶ Journal of Experimental & Theoretical Artificial Intelligence ▶ List of Issues
▶ Volume 28, Issue 1-2 ▶ Fully informed artificial bee colony alg ...

Journal of Experimental & Theoretical Artificial Intelligence >
Volume 28, 2016 - Issue 1-2: Advances and Applications of Swarm Intelligence



2027 Views | 0 CrossRef citations to date | Altmetric



Articles

Fully informed artificial bee colony algorithm

Kavita Sharma , P.C. Gupta & Harish Sharma

Pages 403-416 | Received 16 Mar 2015, Accepted 25 May 2015, Published online: 25 Jul 2015

 Download citation  <https://doi.org/10.1080/0952813X.2015.1056238>




 Full Article  Figures & data  References  Citations  Metrics

 Reprints & Permissions  PDF

Abstract

The Gbest-guided artificial bee colony (GABC) algorithm is a latest swarm intelligence-based approach to solve optimisation problem. In GABC, the individuals update their respective positions by drawing inspiration from the global best solution available in the current swarm. The GABC is a popular variant of the artificial bee colony (ABC) algorithm and is proved to be an efficient algorithm in terms of convergence speed. But, in this strategy, each individual is simply

In this article  cal best solution, which may lead to trap in local optima.
er. a new search strategy, namely “Fully Informed Learning” is

Research Article

Fully Informed Artificial Bee Colony Algorithm

Kavita Sharma,^a P C Gupta,^b Harish Sharma,^{c*}

^a *Government Polytechnic College, Kota;*

^b *University of Kota, Kota;*

^c *Vardhaman Mahaveer Open University, Kota;*

(Received 00 Month 20xx; final version received 00 Month 20xx)

The Gbest-guided Artificial Bee Colony (GABC) algorithm is a latest swarm intelligence based approach to solve optimization problem. In GABC, the individuals update their respective positions by drawing inspiration from the global best solution available in the current swarm. The GABC is a popular variant of Artificial Bee Colony (ABC) algorithm and is proved to be an efficient algorithm in terms of convergence speed. But, in this strategy, each individual is simply influenced by the global best solution, which may lead to trap in local optima. Therefore, in this paper, a new search strategy, namely “Fully Informed Learning” is incorporated in the onlooker bee phase of ABC algorithm. The developed algorithm is named as Fully Informed Artificial Bee Colony (FABC) algorithm. To validate the performance of FABC, it is tested on 20 well known benchmark optimization problems of different complexities. The results are compared with GABC and some more recent variants of ABC. The results are very promising and show that the proposed algorithm is a competitive algorithm in the field of swarm intelligence based algorithms.

Keywords: Numerical optimization ; Swarm intelligence ; Fully Informed Learning ; Artificial Bee Colony

1. Introduction

Swarm Intelligence is one of the recent outcome of the research in the field of Nature inspired algorithms(6, 12, 15, 17). Collaborative trial and error method is the main concept behind the Swarm Intelligence which enables the algorithmic procedure to find the solution. D.Karaboga (9) contributed the recent addition to this category known as Artificial bee colony (ABC) algorithm. The ABC algorithm mimics the foraging behavior of honey bees while searching food for them. ABC is a simple and population based optimization algorithm. Here the population consists of possible solutions in terms of food sources for honey bees whose fitness is regulated in terms of nectar amount which the food source contains. Artificial Bee Colony is made of three groups of bees: employed bees, onlooker bees and scout bees. The number of employed and onlooker bees is equal. The employed bees searches the food source in the environment and store the information like the quality and the distance of the food source from the hive. Onlooker bees wait in the hive for employed bees and after collecting information from them, they start searching in neighborhood of that food sources which are having better nectar. If any food source is abandoned then scout bee finds new food source randomly in search space. While searching the solution of any optimization problem, ABC algorithm first initializes ABC parameters and swarm then it requires the repetitive iterations of the three phases namely employed bee phase, onlooker bee phase and scout bee phase.

*Corresponding author. Email: hsharma@vmou.ac.in

However the ABC achieves a good solution at a significantly faster rate but, like the other optimization algorithms, it is also weak in refining the already explored search space. It is shown in literature that basic ABC itself has some drawbacks like stop proceeding toward the global optimum even though the population has not converged to a local optimum (10) and it is observed that the position update equation of ABC algorithm is good at exploration but poor at exploitation (19) i.e, has not a proper balance between exploration and exploitation. Therefore these drawbacks require a modification in position update equation in ABC. To enhance the exploitation, Wei-feng Gao et al. (8) improved position update equation of ABC such that the bee searches only in neighborhood of the previous iteration's best solution. Anan Banharnsakun et al. (2) proposed the best-so-far selection in ABC algorithm and incorporated three major changes: The best-so-far method, an adjustable search radius, and an objective-value-based comparison in ABC. To solve constrained optimization problems, D. Karaboga and B. Akay (11) used Debs rules consisting of three simple heuristic rules and a probabilistic selection scheme in ABC algorithm.

In 2010, Zhu and Kwong (19) proposed an improved ABC algorithm, namely Gbest-guided ABC (GABC) algorithm by incorporating the information of global best (Gbest) solution into the solution search equation to improve the exploitation. But as all the individuals drawing inspiration from the global best solution, there is a enough chance of swarm stagnation. Therefore, in this paper, a new position update strategy, namely "Fully Informed Learning" (14) is incorporated in the onlooker phase of GABC algorithm. The proposed algorithm is named is Fully Informed ABC (FABC). The FABC algorithm is tested on 20 benchmark problems and the results are very encouraging.

Rest of the paper is organized as follows: ABC is explained in Section 2. In Section 3, FABC algorithm is proposed and explained. In Section 4, performance of the proposed algorithm is analyzed via numerical experiment. Finally, in Section 5, paper is concluded.

2. Artificial Bee Colony(ABC) algorithm

The ABC algorithm is divided importantly into three phases, namely employed bee phase, onlooker bee phase, and scout bee phase. Each of the phase is explained in details in subsequent sections. First initialization of the solutions is done as:

2.1. Initialization of the swarm

If D is the number of variables in the optimization problem then each food source $x_i (i = 1, 2, \dots, SN)$ is a D -dimensional vector among the SN food sources and is generated using a uniform distribution as:

$$x_{ij} = x_{minj} + rand[0, 1](x_{maxj} - x_{minj}) \quad (1)$$

here x_i represents the i^{th} food source in the swarm, x_{minj} and x_{maxj} are bounds of x_i in j^{th} dimension and $rand[0, 1]$ is a uniformly distributed random number in the range $[0, 1]$. After initialization phase ABC requires the cycle of the three phases namely employed bee phase, onlooker bee phase and scout bee phase to be executed.

2.2. Employed bee phase

In this phase, i^{th} candidate's position is updated using following equation:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (2)$$

here $k \in \{1, 2, \dots, SN\}$ and $j \in \{1, 2, \dots, D\}$ are randomly chosen indices and $k \neq i$. ϕ_{ij} is a random number in the range $[-1, 1]$. After generating new position, the position with better fitness between the newly generated and old one is selected.

2.3. Onlooker bees phase

In this phase, employed bees share the information associated with its food source like quality (nectar) and position of the food source with the onlooker bees in the hive. Onlooker bees evaluate the available information about the food source and based on its fitness it selects a solution with a probability $prob_i$. Here $prob_i$ can be calculated as function of fitness (there may be some other):

$$prob_i(G) = \frac{0.9 \times fitness_i}{maxfit} + 0.1, \quad (3)$$

here $fitness_i$ is the fitness value of the i^{th} solution and $maxfit$ is the maximum fitness amongst all the solutions. Based on this probability, onlooker selects a solution and modifies it using the same equation (2) as in employed bee phase. Again by applying greedy selection, if the fitness is higher than the previous one, the onlooker bee stores the new position in its memory and forgets the old one.

2.4. Scout bees phase

If for a predetermined number of cycles, any bee's position is not getting updated then that food source is taken to be abandoned and this bee becomes scout bee. In this phase, the abandoned food source is replaced by a randomly chosen food source within the search space. In ABC, the number of cycles after which a particular food source becomes abandoned is known as *limit* and is a crucial control parameter. In this phase the abandoned food source x_i is replaced by a randomly chosen food source within the search space using the equation (1) as in initialization phase.

3. Fully Informed Artificial Bee Colony Algorithm

This section explains the proposed modified ABC algorithm. In ABC, at any instance, a solution is updated through information flow from other solutions of the swarm. This position updating process uses a linear combination of current position of the potential solution which is going to be updated and position of a randomly selected solution as step size with a random coefficient $\phi_{ij} \in [-1, 1]$. This process plays an important role to decide the quality of new solution. If the current solution is far from randomly selected solution and absolute value of ϕ_{ij} is also high then the change will be large enough to jump the true optima. On the other hand, small change will decrease the convergence rate of whole ABC process. Further, It is also suggested in literatures (10, 19) that basic ABC itself has some drawbacks, like stop proceeding toward the global optimum even though the population has not converged to a local optimum and it is observed that the position update equation of ABC algorithm is good at exploration but poor at exploitation. Therefore, to improve the exploitation capability of ABC algorithm, in 2010, Zhu and Kwong (19) proposed an improved ABC algorithm called Gbest-guided ABC (GABC) algorithm by incorporating the information of global best (Gbest) solution into the solution search equation to improve the exploitation. GABC is inspired by PSO (12), which, in order to improve the exploitation, takes advantage of the information of the global best (gbest) solution to guide the search by candidate solutions.



Kavita Sharma <erkavita28@gmail.com>

Refereeing Process: Editor comments IJIEI-327309

1 message

Inderscience Publishers <noreply@indersciencemail.com>

Sat, Jul 24, 2021 at 2:09 AM

Reply-To: Inderscience Publishers <noreply@indersciencemail.com>

To: "Ms. Kavita Sharma" <erkavita28@gmail.com>, "Dr. P C Gupta" <dr.pcgupta@uok.ac.in>, "Prof. Ahmad Taher Azar" <ahmad_t_azar@ieee.org>

Dear Authors

Additional 5 references from International Journal of Intelligent Engineering Informatics (IJIEI) from recent years (2019-2021) should be cited inside the text.

References must be cited in Harvard format NOT IEEE format.

The in-text citation must be written in Harvard format.

The references must be also written in Harvard format.

Read the reference guidelines carefully:

<https://www.inderscience.com/www/dl.php?filename=refguide.pdf>

All references must be revised and upload the new revised version

Prof. Ahmad Taher Azar

Int. J. of Intelligent Engineering Informatics (IJIEI)

submissions@inderscience.com

Fully Informed ABC Algorithm for Large Scale Job Shop Scheduling problem

the date of receipt and acceptance should be inserted later

Abstract The large-scale job-shop scheduling problem (LSJSSP) is a complex scheduling problems. Previously, although the nature-inspired algorithm, specially the swarm intelligence (SIA) based algorithms have been efficiently applied to solve the LSJSSP, finding the best solution for LSJSSP instances remains a challenging task. Therefore, in this paper, a novel SIA is applied to solve the 105 LSJSSP instances. The selected SIA is Fully Informed Artificial Bee Colony (FABC) algorithm. The FABC algorithm is a variant of the ABC algorithm in which position update process is inspired from the GABC. In the FABC, onlooker bee process of the ABC strategy is modified and designed such that the new position of the solution search agent is obtained while learning from all the nearby agents. The results obtained by the FABC is compared with the strategies available in the literature. The results analysis shows that the proposed approach to solving LSJSSP is competitive in the field of SIA.

Keywords Fully Informed Learning · Swarm Intelligence · Artificial bee colony · large-scale JSS Problem

1 Introduction

Efficient scheduling is crucial for making the best use of available resources. In the domain of production management, the Large Scale Job-shop Scheduling Problem (LSJSSP) is a complicated combinatorial optimization problem. JSSP needs n jobs to be accomplished on m systems (machines). The system order for all jobs is fixed and varies depending on the jobs. The jobs are put in place in a non-preemptive manner, which means that while one job is running on one system, it cannot be disrupted by another. The primary goal of JSSP is to find an appropriate sequence scheme that reduces the time it takes for all jobs to be completed, which is referred to as makespan (MS). The goal is to minimize the makespan (MS) [8,33].

The LSJSSP is one of the most important NP-hard problem. To solve LSJSSP, several deterministic conventional mathematical models and heuristic methods have been used. To small size LSJSSP cases, mathematical models have a successful solution in a reasonable amount of time. [1]. The computational time increases exponentially as the size of the instances grows. So, for a larger scale LSJSSP, Non-conventional nature inspired algorithms (NIAs) are preferred alternatives [7]. The numerous processes found in nature are used to create NIAs. Swarm intelligence based algorithms (SIA) and evolutionary algorithms (EAs) are the two main types of NIAs. The design of SIA was influenced by the intellectual actions of creatures. Some state-of-art SIA are Artificial bee colony (ABC)[12], spider monkey optimization (SMO) [4], teaching learning based optimization (TLBO) [23] etc.,. EAs like differential evolution (DE) [32], genetic algorithm (GA) [9] etc., are based on biotic transformation like crossover, selection etc.

In recent years, NIAs are performing very well to solve physical world problems [28,27]. In 2020, the ABC algorithm is applied to solved the colour image watermarking problem in hybrid DDS (DWT-DCT-SVD) transform domain [30]. Further, N. Sharma et. al proposed a new variant of ABC algorithm

for numerical optimisation problems[29]. In this series, many NIAs emerged well to solve LSJSSP such as particle swarm optimization (PSO) [5], hybrid biogeography based optimization (BBO) algorithm [36], genetic algorithm (GA) [10], hybrid differential evolution algorithm [21], multiple type individual enhancement PSO (MPSO) algorithm [14], classical LSJSSP [25], differential based harmony search algorithm with variable neighborhood search [41], biased random key genetic algorithm [11], new neighbourhood structure based algorithm [7], teaching learning based optimization (TLBO) algorithm [13], improved ABC (IABC) algorithm [37], discrete ABC (DABC) [38], best so far ABC [3], parallel ABC (pABC) algorithm [2], beer froth ABC [27] etc. Further, N. S. Rathore et. al. presented a novel WOA based controller design in 2019 [24]. In terms of computational time and solution efficiency, the obtained results are acceptable. At the same time, finding a solution for larger JSSP instances is a challenging task. These findings motivate researchers to continue their work in order to solve LSJSSP.

In light of the above, this paper proposes a solution to the LSJSSP instances by using an efficient ABC-based algorithm called Fully Informed Artificial Bee Colony (FABC). The FABC algorithm was developed by K. Sharma et. al. [26]. The FABC algorithm is designed by taking inspiration from the Gbest-guided ABC (GABC)[42]. Zhu and Kwong developed an efficient variant of ABC algorithm named GABC [42], that improved exploitation capability of the solution search process while learning from the global best (Gbest) solution. However, since every agent (solution) is inspired by the global best approach, there is a good possibility of swarm stagnation. Therefore, the onlooker stage of the GABC algorithm is modified to improve the exploration ability of the search agents. In the modified onlooker bee stage “Fully Informed Learning” [15] strategy is incorporated. In this paper the FABC algorithm is applied to solve 105 LSJSSP instances. The results are analysed and compared to other important methods available in the literature. The reported findings substantiate the authenticity of the FABC strategy.

The remainder of the paper is structured as follows: FABC is explained in Section 2. Formulation of LSJSSP is discussed during the section 3. The entire process for solving LSJSSP using the proposed strategy is discussed in section 4. The section 5 shows the parameter setting and results analysis. Finally, the section 6 summarises the carried out work and suggests future work in the field.

2 Fully Informed ABC

The employed bee stage, onlooker bee stage, and scout bee stage are the three key phases of the Fully Informed ABC (FABC) algorithm. In the following sections, we will go through each stage in depth. The solution agents are initially initialised as follows:

2.1 Initialization of the solution agents

The solution agents are randomly initialized in the given search space. If the search range of the given problem is $[B_{minj}, B_{maxj}]$ then the total number of solution agents (*TSA*) are initialized as follows:

$$SA_{ij} = B_{minj} + r[0,1](B_{maxj} - B_{minj}) \quad (1)$$

here SA_i represents the i^{th} solution agent in the swarm, B_{minj} and B_{maxj} are bounds of SA_i in j^{th} dimension whereas $r[0,1]$ represents the uniformly distributed random number. The range of r is $[0, 1]$.

The initialization stage is same in all the SIAs. After this stage, the FABC executes its three stage namely, employed bee, onlooker bee, and scout bee cyclically.

2.2 Employed bee stage

In the employed bee stage of the FABC, every solution will get chance to update its position using the following equation.

$$SAU_{ij} = SA_{ij} + r[0,1](SA_{ij} - SA_{kj}) + \psi_{ij}(Best_j - SA_{ij}) \quad (2)$$

In equation 2, SAU_{ij} is the updated position of solution SA_{ij} . $Best_j$ is the j^{th} element of the best solution found so far. Further, ψ_{ij} is a number randomly generated in the range $[0, PC]$, where PC

is a positive constant and SA_{kj} is a neighbouring solution agent. It is clear from equation 2 that the solution agents update their positions while learning from the nearby agents as well as attracting towards the best solution agent in the swarm. The term $\psi_{ij}(Best_j - SA_{ij})$ helps the swarm to converge at the best solution location but this may lead to pre-mature convergence. Here the parameter PC helps in balancing the exploration and convergence ability of the FABC algorithm. After getting the updated position of the solution agent, a greedy selection mechanism (GSM) is applied between the update position SMU_{ij} and the old position SM_{ij} . The best one is selected for the next stage.

2.3 Onlooker bees stage

During this process, employed bees exchange details about their food source with onlooker bees in the comb, such as the quality, distance and direction of the food source. In terms of FABC algorithm, this stage is used to update the solutions shared by the employed bee stage on the basis of their quality. The quality of the solutions are measured using the probability $prob_i$ which is a function of fitness of the solution agent. The $prob_i$ is calculated using following equation:

$$prob_i = \frac{0.9 \times fit_i}{maxfit} + 0.1, \quad (3)$$

In equation 3, fit_i shows the fitness of i^{th} solution agent whereas $maxfit$ represents the maximum fitness in the swarm. On the basis of this $prob_i$, the quality of the solution agent is evaluated and based on that the solution agent is given chance to update its position. Therefore, we can say that in this stage the better solutions will get more chance to update the positions in compare to the less fit solutions. Further, in this stage, the fully informed learning strategy is applied in the position update process of the solution agents. The position update equation is shown below:

$$SAU_{ij} = x_{ij} + r[0, 1] \frac{\sum_{k=1}^{TSA} (SA_{ij} - SA_{kj})}{TSA} + \psi_{ij}(Best_j - SA_{ij}) \quad (4)$$

here TSA is the total number of solution agents while other parameters are same as mentioned in equation 2. Here, it can be observed that a solution agent achieve a new position while learning from all the solution agents of swarm as well as direction of the best solution of the swarm. As the learning from all the solutions is involve in this position update process, the possibility of pre-mature convergence is reduced. the new position is achieved. So, in the fully informed learning, to update its location in the search space, the agent gathers knowledge from all the neighbouring solutions as well as the best solution in the swarm. The new position of the solution agent is compared with the old one using the GSM, and the best candidate solution will take part in the next generation of the FABC.

2.4 Scout bees stage

The scout bee stage is used to reduce the possibility of stagnation of the swarm. The stagnation is the situation in which all the solution agents gathered at the same location of the search space hence the inter-agent distance becomes negligible. As the position update process depends on the inter-agent distance, the movement of the solution agents is reduced. Hence the solution agents stagnated at the same location.

In this stage, the number of update of every solution agent is checked. If any of the solution agent is not updating its position up to the pre-defined number (*limit*) of iterations then that solution agent is considered as exhausted and a new solution is randomly generated in the search space in place of that solution. Hence, the situation of stagnation can be reduced while introducing the fluctuation in the swarm through random initialization.

3 Job shop scheduling problem organisation

The LSJSSP can be interpreted in following manner: There are a set of n jobs to be processed using m machines. To complete the execution, each job has to be passed through all the m systems in a given predefined sequence. Each job consists of total m operations. To perform operations a job uses one of the machine. When any of the job is executing on any machine it cannot be interrupted by other jobs. The total number of operations are $m \times n$ that are scheduled on m systems [16].

A solution of the LSJSSP will be the minimum completion time for the considered jobs that is called the makespan (MS). Mathematically the problem is stated as :

$$\text{Minimize } MS_{max} \quad (5)$$

where, $MS_{max} = \max(MS_1, MS_2, MS_3, MS_4, \dots, MS_n)$. $MS_1, MS_2, MS_3, MS_4, \dots, MS_n$ are the completion time for all the n jobs. Followings are the constraints for LSJSSP [11]:

- Each system can process at most one operation at a time.
- The completion time of any operation must be a positive integer.
- Precedence relationships among the different jobs must be satisfied.

4 FABC for LSJSSP

The FABC algorithm is used to solve LSJSSP instances, and the whole method is detailed here. Since LSJSSP is a discrete optimization problem, a solution in the proposed algorithm is a discrete valued vector (representing a potential operation scheduling list). The reordering of jobs for FABC is used to estimate each solution in the search field. To generate the discrete valued sequence from a continuous valued vector we have used random key encoding (RKE) scheme [35].

In RKE encoding scheme, first a continuous valued vector is arranged in increasing order using the integer numbers from 1 to $n \times m$ for the n jobs and m available systems (machines). As each job has to go through m systems for completing its execution so further transformation from this integer sequence is performed using (Integer value mod $n + 1$). The integer series is transformed to operation order sequence using this transformation, and each job index has m occurrences. Figure 1 depicts the transformation of a continuous valued vector into a discrete valued vector, followed by an operation scheduling sequence. Through EKE, we have to get the discrete value sequence list which can decrease the MS value. The goal is to figure out a series of operations that reduces the overall time it takes to complete all of the jobs. The detailed procedure is described in the subsequent steps:

Continuous valued solution	0.9	0.6	0.8	0.2	0.5	0.3
Decoded as	6	4	5	1	3	2
Operation sequence	1	2	3	2	1	3

Fig. 1: Random Key (RKE) Encoding Scheme

4.1 Initialization stage

The parameters of the proposed FABC algorithm namely, total number of solution agents, number of employed and onlooker solution agents, and total number of iterations are initialized. Each solution agent is initialized in the search space using the equation 1. As all the initialized sources are continuous in nature, RKE scheme is used to generate the corresponding discrete valued operation sequence. Now the MS value (objective value) for each operation sequence is calculated.

4.2 Employed honeybee stage

At this stage, using equation 2 all the continuous valued solution agents are modified. The solution agents is modified as per the information of the neighbouring agents. The updated solution agent is in continuing form, so again RKE encoding scheme is applied to alter this continuous valued solutions in to corresponding discrete operation sequence list. The MS value for this operation sequence is computed. If the corresponding MS value is better then the previous value then the solution agent corresponding to this operation sequence is selected for the next generation.

4.3 Onlooker honeybee stage

In onlooker honeybee stage, the probability for all the solution agents are assessed using the equation 3. The solution agents are chosen and updated as per the equation 3 and 4 respectively. Again the solution agent is updated based upon the information obtained from the neighbouring solution agents. To obtain the corresponding operation sequence, RKE scheme is applied on the produced continuous valued solution agent. The MS value is computed from the generated operation sequence. GSM is used to choose new solution agent for the upcoming generation.

4.4 Scout honeybee stage

If a solution agent does not update it's position up to the *limit*, then it is discarded and re-initialized in the search space using the equation 1. The produced solution agent is in continuous form, again RKE scheme is applied to obtain the corresponding operation sequence. Calculate the MS value from this operation sequence.

The pseudo-code of the designed approach for LSJSSP is shown in Algorithm 1.

Algorithm 1 FABC algorithm for LSJSSP

```
Parameter Initialization
Total solution agents =  $TSA$ .
D (Dimension) =  $m \times n$ 
Total generation count =  $MGN$ 
CurrentIndex=1.
Solution agents initialization in the search space using equation 1.
Conversion of continuous valued solution agents into an operation sequence (discrete valued solutions) to deal LSJSSP using RKE scheme.
 $MS$  value Computation for every operation sequence.
While (CurrentIndex  $\leq$   $MGN$ ) do
- Step 1: Employed honeybee stage
  - Revise the location of every solution agent as per the equation 2.
  - The newly generated solution agent is in continuous form only, so it is converted in to a discrete valued operation sequence using RKE scheme.
  - Compute the  $MS$  value from the generated operation sequence.
  - Apply  $GSM$  to select the new solution agent on the basis of the  $MS$  value of its respective operation sequence.
  - The former solution agent will be substituted by the new solution agent if respective operation sequence vector has a better  $MS$  value.
- Step 2: Onlooker honeybee stage
  - Probability  $prot_i$  computation using equation 3 for each solution agent.
  - Revise the location of solution agent using the equation 4 selected as equation 3.
  - Obtain the new discrete operation sequence from the recently revised continuing solution agents using RKE scheme.
  -  $MS$  value computation for recently produced operation sequence.
  - Apply  $GSM$  to select the new solution agent for the next generation.
  - The former solution agent will be substituted by the new solution agent if corresponding discrete solution sequence has a better  $MS$  value.
- Step 3: Scout honeybee stage
  - If a solution agent does not modify its position up to limit.
  - Randomly initialized that solution agent as per equation 1 in the search space.
  - Apply RKE scheme for producing discrete solution vector from this continuous valued solution agent.
  -  $MS$  value computation for the operation scheduling list.
- Step 4: Keep in memory the best agent in the swarm
- CurrentIndex=CurrentIndex+1
end while
Outcome will be the best agent found so far;
```

5 Results analysis and Discussion

To prove the effectiveness of FABC algorithm, it is applied on LSJSSP instances. Following 105 LSJSSP instances are considered for experimentation [34, 6, 31].

- 15 SWV instances
- 50 TA instances
- 40 DMU instances

To attain the least *MS* value for all these 105 LSJSSP instances is the main goal. The experimental setting is listed as below:

1. Number of run =10
2. Number of maximum iteration =2000
3. Number of solution agents TSA =50
4. Dimension D = Number of systems × Number of jobs
5. limit = D × TSA

The parametric ambience for the FABC approach and the other considered approaches are kept same in terms of swarm size and maximum number of iterations to carry out an equitable comparison.

The reported results of FABC are compared with the following state-of-art algorithms available in the literature:

- Biased random key genetic algorithm (BRKGA(JSP)) [11]
- A guided tabu search for LSJSSP (NKPR) [17]
- Teaching learning based optimization method (TLBO) [22]
- Differential based harmony search (DHS) algorithm [41]
- A tabu search to solve LSJSSP (TS) [40]
- An advanced tabu search algorithm for LSJSSP (i-TSAB) [18]
- AlgFix [20]
- A tabu search/simulated annealing algorithm for LSJSSP (TS/SA) [39]
- Global equilibrium search technique (GES) [19]

The obtained results for the above three instances are represented in Tables 1 to 3. These tables list the name of the instance, its size, the lower bound (LB), the upper bound (UB) for the best known solution (BKS), the BKS obtained by FABC approach, and BKS value obtained from the compared algorithms. The obtained results for all the instances demonstrate that the proposed FABC is superior approach in reference to *MS* value during assessment with other considered approaches.

Further to analyse the outcomes, average relative percentage error (RPE) is also calculated and compared as tabulated in Table 4. The value of RPE is computed (with respect to the UB value of an instance) as per demonstrated in equation 6.

$$RPE = 100 \times (BKS_{algo} - UB) / UB \quad (6)$$

Here, BKS_{algo} represents the *MS* value obtained using the considered approaches. The attained outcomes of Table 4 demonstrate the significant improvement in the average RPE which assures the authenticity of the introduced approach.

6 Conclusion

This article proposed a solution to solve 105 large scale instances of job shop scheduling problem (LSJSSP) using an efficient fully informed artificial bee colony (FABC). In FABC algorithm, to improve the exploration ability in the solution search strategy of ABC, a new learning strategy is incorporated in the onlooker bee stage by which learning is done from all the neighbouring solutions. The GABC algorithm solution update strategy is adopted in the FABC with the fully informed learning mechanism. The *MS* time is used as an evaluation criterion in the LSJSSP. The results are analysed and compared to cutting-edge techniques proposed by a number of researchers. According to the results of the experiments, the proposed solution gives better solution. In Future, some more performance metrics may be considered for experimentation.

Table 1: Comparison in terms of best MS value for SMV instances

Instance	Size	LB	UB	FABC	BRKGA(JSP)	TS//SA	TS
SWV-01	20 × 10	1407	1407	1407	1407	1412	-
SWV-02	20 × 10	1475	1475	1475	1475	1475	-
SWV-03	20 × 10	1369	1398	1395	1398	1398	-
SWV-04	20 × 10	1450	1474	1465	1470	1470	-
SWV-05	20 × 10	1424	1424	1424	1425	1425	-
SWV-06	20 × 15	1591	1678	1674	1675	1679	-
SWV-07	20 × 15	1446	1600	1572	1594	1603	-
SWV-08	20 × 15	1640	1763	1762	1755	1756	-
SWV-09	20 × 15	1604	1661	1650	1656	1661	-
SWV-10	20 × 15	1631	1767	1736	1743	1754	-
SWV-11	50 × 10	2983	2983	2983	2983	-	2983
SWV-12	50 × 10	2972	2979	2975	2979	-	2979
SWV-13	50 × 10	3104	3104	3104	3104	-	3104
SWV-14	50 × 10	2968	2968	2968	2968	-	2968
SWV-15	50 × 10	2885	2886	2885	2901	-	2886

Table 2: Comparison in terms of best MIS value for TA instances

Instance	Size	LB	UB	FABC	BRKGA(JSP)	GES	AlgFix	i-TSAB	TS/SA	DHS	TLBO	NKPR
TA-01	15 × 15	1231	1231	1231	1231	1231	1231	-	1231	1321	1526	1485
TA-02	15 × 15	1244	1244	1244	1244	1244	1244	-	1244	1313	1538	1476
TA-03	15 × 15	1218	1218	1218	1218	1218	1218	-	1218	1327	1594	1470
TA-04	15 × 15	1175	1175	1175	1175	1175	1175	-	1175	1285	1550	1519
TA-05	15 × 15	1224	1224	1224	1224	1224	1224	-	1224	1315	1581	1517
TA-06	15 × 15	1238	1238	1238	1238	1238	1238	-	1238	1346	1538	1517
TA-07	15 × 15	1227	1227	1227	1227	1228	1228	-	1228	1322	1546	1460
TA-08	15 × 15	1217	1217	1217	1217	1217	1217	-	1217	1304	1534	1446
TA-09	15 × 15	1274	1274	1274	1274	1274	1274	-	1274	1386	1614	1551
TA-10	15 × 15	1241	1241	1241	1241	1241	1241	-	1241	1334	1542	1365
TA-11	20 × 15	1323	1357	1355	1357	1357	1358	1361	1359	1520	1876	1687
TA-12	20 × 15	1351	1367	1367	1367	1367	1367	-	1371	1563	1856	1770
TA-13	20 × 15	1282	1342	1342	1342	1344	1342	-	1342	1513	1849	1713
TA-14	20 × 15	1345	1345	1345	1345	1345	1345	-	1345	1477	1780	1748
TA-15	20 × 15	1304	1339	1337	1339	1339	1339	-	1339	1557	1929	1788
TA-16	20 × 15	1302	1360	1360	1360	1360	1360	-	1360	1543	1852	1716
TA-17	20 × 15	1462	1462	1462	1462	1469	1473	1462	1464	1607	1941	1781
TA-18	20 × 15	1369	1396	1396	1396	1401	1396	-	1399	1601	1817	1776
TA-19	20 × 15	1297	1332	1331	1332	1332	1332	1335	1335	1524	1842	1722
TA-20	20 × 15	1318	1348	1348	1348	1348	1348	1351	1350	1554	1902	1710
TA-21	20 × 20	1539	1643	1643	1642	1647	1643	1644	1644	1854	2399	2165
TA-22	20 × 20	1511	1600	1600	1600	1602	1600	1600	1600	1852	2241	2126
TA-23	20 × 20	1472	1557	1557	1557	1558	1557	1557	1560	1765	2210	2145
TA-24	20 × 20	1602	1646	1646	1646	1653	1646	1647	1646	1829	2241	2173
TA-25	20 × 20	1504	1595	1594	1595	1596	1595	1595	1597	1792	2324	2117
TA-26	20 × 20	1539	1645	1641	1643	1647	1647	1645	1647	1863	2299	2206
TA-27	20 × 20	1616	1680	1680	1680	1685	1686	1680	1680	1905	2436	2194
TA-28	20 × 20	1591	1603	1600	1603	1614	1613	1614	1603	1819	2333	2100
TA-29	20 × 20	1514	1625	1625	1625	1625	1625	-	1627	1853	2280	2146
TA-30	20 × 20	1473	1584	1584	1584	1584	1584	1584	1584	1812	2247	2103
TA-31	30 × 15	1764	1764	1764	1764	1764	1766	-	1764	2037	2528	2382
TA-32	30 × 15	1774	1790	1781	1785	1793	1790	-	1795	2106	2591	2482
TA-33	30 × 15	1778	1791	1791	1791	1791	1791	1793	1796	2091	2685	2511
TA-34	30 × 15	1828	1829	1832	1829	1832	1832	1829	1831	2089	2508	2480
TA-35	30 × 15	2007	2007	2007	2007	2007	2007	-	2007	2139	2509	2512
TA-36	30 × 15	1819	1819	1819	1819	1819	1819	-	1819	2086	2705	2395
TA-37	30 × 15	1771	1771	1778	1778	1779	1784	1778	1778	2067	2512	2436
TA-38	30 × 15	1673	1673	1673	1673	1673	1673	-	1673	1980	2488	2250
TA-39	30 × 15	1795	1795	1795	1795	1795	1795	-	1795	2010	2439	2501
TA-40	30 × 15	1631	1673	1665	1669	1680	1679	1674	1676	1986	2455	2380
TA-41	30 × 20	1859	2006	2006	2008	2008	2022	-	2018	-	-	-
TA-42	30 × 20	1867	1945	1934	1937	1956	1953	1956	1953	-	-	-
TA-43	30 × 20	1809	1814	1836	1852	1870	1869	1859	1858	-	-	-
TA-44	30 × 20	1927	1983	1983	1983	1991	1992	1984	1983	-	-	-
TA-45	30 × 20	1997	2000	2000	2000	2004	2000	2000	2000	-	-	-
TA-46	30 × 20	1940	2008	2008	2004	2011	2011	2021	2010	-	-	-
TA-47	30 × 20	1789	1897	1892	1894	1903	1902	1903	1903	-	-	-
TA-48	30 × 20	1912	1945	1940	1943	1962	1962	1953	1955	-	-	-
TA-49	30 × 20	1915	1966	1962	1964	1969	1974	-	1967	-	-	-
TA-50	30 × 20	1807	1925	1925	1925	1931	1927	1928	1931	-	-	-

Table 3: Comparison in terms of best MS value for DMU instances

Instance	Size	LB	UB	FIABC	BRKGA(JSP)	TS	GES	I-TSAB	ALFlix
DMU-01	20 × 15	2501	2563	2563	2563	2566	2566	2517	2563
DMU-02	20 × 15	2651	2706	2704	2706	2711	2706	2715	2706
DMU-03	20 × 15	2731	2731	2731	2731	-	2731	-	2731
DMU-04	20 × 15	2601	2669	2669	2669	-	2669	-	2669
DMU-05	20 × 15	2749	2749	2749	2749	-	2749	-	2749
DMU-06	20 × 20	2834	3244	3242	3244	3254	3250	3265	3244
DMU-07	20 × 20	2677	3046	3045	3046	-	3053	-	3046
DMU-08	20 × 20	2901	3188	3188	3188	3191	3197	3199	3188
DMU-09	20 × 20	2739	3092	3091	3092	-	3092	3094	3096
DMU-10	20 × 20	2716	2984	2982	2984	-	2984	2985	2984
DMU-11	30 × 15	3395	3453	3454	3445	3455	3453	3470	3455
DMU-12	30 × 15	3481	3516	3512	3513	3516	3518	3519	3522
DMU-13	30 × 15	3681	3681	3681	3681	3681	3697	3698	3687
DMU-14	30 × 15	3394	3394	3394	3394	-	3394	3394	3394
DMU-15	30 × 15	3332	3343	3343	3343	-	3343	-	3343
DMU-16	30 × 20	3726	3759	3748	3751	3759	3781	3787	3772
DMU-17	30 × 20	3697	3836	3828	3830	3842	3848	3854	3836
DMU-18	30 × 20	3844	3846	3844	3844	3846	3849	3854	3852
DMU-19	30 × 20	3650	3775	3769	3770	3784	3807	3823	3775
DMU-20	30 × 20	3604	3712	3712	3712	3716	3739	3740	3712
DMU-21	40 × 15	4380	4380	4380	4380	-	4380	-	4380
DMU-22	40 × 15	4325	4725	4725	4725	-	4725	-	4725
DMU-23	40 × 15	4668	4668	4668	4668	-	4668	-	4668
DMU-24	40 × 15	4648	4648	4648	4648	-	4648	-	4648
DMU-25	40 × 15	4164	4164	4164	4164	-	4164	-	4164
DMU-26	40 × 20	4647	4647	4647	4647	4647	4667	4679	4688
DMU-27	40 × 20	4848	4848	4848	4848	-	4848	4848	4848
DMU-28	40 × 20	4692	4692	4692	4692	-	4692	-	4692
DMU-29	40 × 20	4691	4691	4691	4691	-	4691	4691	4691
DMU-30	40 × 20	4732	4732	4732	4732	-	4732	4732	4749
DMU-31	50 × 15	5640	5640	5640	5640	-	5640	-	5640
DMU-32	50 × 15	5927	5927	5927	5927	-	5927	-	5927
DMU-33	50 × 15	5728	5728	5728	5728	-	5728	-	5728
DMU-34	50 × 15	5385	5385	5385	5385	-	5385	-	5385
DMU-35	50 × 15	5635	5635	5635	5635	-	5635	-	5635
DMU-36	50 × 20	5621	5621	5621	5621	-	5621	-	5621
DMU-37	50 × 20	5851	5851	5851	5851	-	5851	5851	5851
DMU-38	50 × 20	5713	5713	5713	5713	-	5713	-	5713
DMU-39	50 × 20	5747	5747	5747	5747	-	5747	-	5747
DMU-40	50 × 20	5577	5577	5577	5577	-	5577	-	5577

Table 4: Comparison based upon Average RPE

Approach	Instances Solved	ARPE (%)	FABC(%)	Improvement (%)
TA instances				
BRKGA(JSP) [11]	50	0.015	-0.093	0.108
GES [19]	50	0.218	-0.093	0.311
AlgFix [20]	50	0.556	-0.093	0.649
i-TSAB [18]	25	0.269	-0.15	0.419
TS/SA [39]	50	0.156	-0.093	0.249
DHS [41]	40	12.412	-0.104	12.516
TLBO [22]	40	36.883	-0.104	36.987
NKPR [17]	40	28.951	-0.104	29.055
DMU Instances				
BRKGA(JSP) [11]	40	-0.021	-0.027	0.006
TS [40]	13	0.097	-0.072	0.169
GES [19]	40	0.107	-0.027	0.134
i-TSAB [18]	20	0.24	-0.06	0.3
AlgFix [20]	40	0.056	-0.027	0.083
SWU Instances				
BRKGA-JSP [11]	15	-0.156	-0.364	0.52
TS/SA [39]	10	-0.073	-0.529	0.602
TS [40]	5	0	-0.0338	0.0338

References

1. Joseph Adams, Egon Balas, and Daniel Zawack. The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3):391–401, 1988.
2. Leila Asadzadeh. A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy. *Computers & Industrial Engineering*, 2016.
3. Anan Banharnsakun, Booncharoen Sirinaovakul, and Tiranee Achalakul. Job shop scheduling with the best-so-far abc. *Engineering Applications of Artificial Intelligence*, 25(3):583–593, 2012.
4. Jagdish Chand Bansal, Harish Sharma, Shimpi Singh Jadon, and Maurice Clerc. Spider monkey optimization algorithm for numerical optimization. *Memetic computing*, 6(1):31–47, 2014.
5. M Clerc and J Kennedy. Standard pso 2011. *Particle Swarm Central Site [online] http://www.particleswarm.info*, 2011.
6. Ebru Demirkol, Sanjay Mehta, and Reha Uzsoy. Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109(1):137–141, 1998.
7. Liang Gao, Xinyu Li, Xiaoyu Wen, Chao Lu, and Feng Wen. A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem. *Computers & Industrial Engineering*, 88:417–429, 2015.
8. Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.
9. David E Goldberg and John Henry Holland. Genetic algorithms and machine learning. 1988.

10. José Fernando Gonçalves, Jorge José de Magalhães Mendes, and Maurício GC Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, 167(1):77–95, 2005.
11. JOSÉ FERNANDO Gonçalves and MAURICIO GC Resende. A biased random-key genetic algorithm for job-shop scheduling. *AT&T Labs Research Technical Report*, 46:253–271, 2011.
12. Dervis Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.
13. HS Keesari and RV Rao. Optimization of job shop scheduling problems using teaching-learning-based optimization algorithm. *Opsearch*, 51(4):545–561, 2014.
14. Tsung-Lieh Lin, Shi-Jinn Horng, Tzong-Wann Kao, Yuan-Hsin Chen, Ray-Shine Run, Rong-Jian Chen, Jui-Lin Lai, and I-Hong Kuo. An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, 37(3):2629–2636, 2010.
15. R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: simpler, maybe better. 8(3):204–210.
16. Renato EN Moraes. Job shop scheduling. *Universidade Federal Fluminense*.
17. Mohammad Mahdi Nasiri and Farhad Kianfar. A guided tabu search/path relinking algorithm for the job shop problem. *The International Journal of Advanced Manufacturing Technology*, 58(9-12):1105–1113, 2012.
18. Eugeniusz Nowicki and Czesław Smutnicki. An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*, 8(2):145–159, 2005.
19. Panos M Pardalos and Oleg V Shylo. An algorithm for the job shop scheduling problem based on global equilibrium search techniques. *Computational Management Science*, 3(4):331–348, 2006.
20. Panos M Pardalos, Oleg V Shylo, and Alkis Vazacopoulos. Solving job shop scheduling problems utilizing the properties of backbone and big valley. *Computational Optimization and Applications*, 47(1):61–76, 2010.
21. Bin Qian, Ling Wang, Rong Hu, Wan-Liang Wang, De-Xian Huang, and Xiong Wang. A hybrid differential evolution method for permutation flow-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 38(7-8):757–777, 2008.
22. R Venkata Rao, Vimal J Savsani, and DP Vakharia. Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems. *Information sciences*, 183(1):1–15, 2012.
23. Ravipudi V Rao, Vimal J Savsani, and DP Vakharia. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43(3):303–315, 2011.
24. Natwar Singh Rathore and VP Singh. Whale optimisation algorithm-based controller design for reverse osmosis desalination plants. *International Journal of Intelligent Engineering Informatics*, 7(1):77–88, 2019.
25. Scott R Schultz, Thom J Hodgson, and Russell E King. On solving the classic job shop makespan problem by minimizing lmax. *Raleigh, NC: Department of Industrial Engineering, North Carolina State University*, 2004.
26. Kavita Sharma, PC Gupta, and Harish Sharma. Fully informed artificial bee colony algorithm. 28(1-2):403–416.
27. Nirmala Sharma, Harish Sharma, and Ajay Sharma. Beer froth artificial bee colony algorithm for job-shop scheduling problem. *Applied Soft Computing*, 68:507–524, 2018.
28. Nirmala Sharma, Harish Sharma, and Ajay Sharma. An effective solution for large scale single machine total weighted tardiness problem using lunar cycle inspired artificial bee colony algorithm. *IEEE/ACM transactions on computational biology and bioinformatics*, 2019.
29. Nirmala Sharma, Harish Sharma, Ajay Sharma, and Jagdish Chand Bansal. Dung beetle inspired local search in artificial bee colony algorithm for unconstrained and constrained numerical optimisation. *International Journal of Intelligent Engineering Informatics*, 8(4):268–304, 2020.
30. Sourabh Sharma, Harish Sharma, and Janki Ballabh Sharma. Artificial intelligence based watermarking in hybrid dds domain for security of colour images. *International Journal of Intelligent Engineering Informatics*, 8(4):331–345, 2020.
31. Robert H Storer, S David Wu, and Renzo Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Management science*, 38(10):1495–1509, 1992.
32. Rainer Storn and Kenneth Price. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.
33. Shyam Sundar, Ponnuthurai N Suganthan, Chua Tay Jin, Cai Tian Xiang, and Chong Chin Soon. A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint. *Soft Computing*, 21(5):1193–1202, 2017.
34. Eric Taillard. Benchmarks for basic scheduling problems. *European journal of operational research*, 64(2):278–285, 1993.
35. A Tamilarasi and T Anantha Kumar. Job-shop scheduling using random key encoding scheme particle swarm optimization. *International Journal of Computational Intelligence Research*, 6(1):33–43, 2010.
36. Xiaohua Wang and Haibin Duan. A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Computers & Industrial Engineering*, 73:96–114, 2014.
37. Bao Zhen Yao, Cheng Yong Yang, Juan Juan Hu, Guo Dong Yin, and Bo Yu. An improved artificial bee colony algorithm for job shop problem. In *Applied Mechanics and Materials*, volume 26, pages 657–660. Trans Tech Publ, 2010.
38. Minghao Yin, Xiangtao Li, and Junping Zhou. An efficient job shop scheduling algorithm based on artificial bee colony. *Scientific Research and Essays*, 6(12):2578–2596, 2011.
39. Chao Yong Zhang, PeiGen Li, YunQing Rao, and ZaiLin Guan. A very fast ts/sa algorithm for the job shop scheduling problem. *Computers & Operations Research*, 35(1):282–294, 2008.
40. ChaoYong Zhang, PeiGen Li, ZaiLin Guan, and YunQing Rao. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, 34(11):3229–3242, 2007.
41. Fuqing Zhao, Shuo Qin, Guoqiang Yang, Weimin Ma, Chuck Zhang, and Houbin Song. A differential-based harmony search algorithm with variable neighborhood search for job shop scheduling problem and its runtime analysis. *IEEE Access*, 6:76313–76330, 2018.
42. Guopu Zhu and Sam Kwong. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation*, 217(7):3166–3173, 2010.

We're sorry, something doesn't seem to be working properly.

Please try refreshing the page. If that doesn't work, please contact support so we can address the problem.

Original Article | [Published: 11 July 2015](#)

Fitness based particle swarm optimization

[Kavita Sharma](#), [Varsha Chhamunya](#), [P C Gupta](#), [Harish Sharma](#)  & [Jagdish Chand Bansal](#)

[International Journal of System Assurance Engineering and Management](#) **6**, 319–329 (2015)

178 Accesses | **11** Citations | [Metrics](#)

Abstract

Particle swarm optimization (PSO) is a popular population based approach used to solve nonlinear and complex optimization problems. It is simple to implement and swarm based probabilistic algorithm but, it also has drawbacks like it easily falls into local optima and suffers from slow convergence in the later stages. In order to reduce the chance of stagnation, while improving the convergence speed, a new position updating phase is incorporated with PSO, namely fitness based position updating in PSO. The proposed phase is inspired from the onlooker bee phase of Artificial Bee Colony (ABC) algorithm. In the proposed position updating phase, solutions update their

Fitness based Particle Swarm Optimization

Kavita Sharma · Varsha Chhamunya · P C Gupta ·
Harish Sharma · Jagdish Chand Bansal

the date of receipt and acceptance should be inserted later

Abstract Particle Swarm Optimization (PSO) is a popular population based approach used to solve nonlinear and complex optimization problems. It is simple to implement and swarm based probabilistic algorithm but, it also has drawbacks like it easily falls into local optima and suffers from slow convergence in the later stages. In order to reduce the chance of stagnation, while improving the convergence speed, a new position updating phase is incorporated with PSO, namely fitness based position updating in PSO. The proposed phase is inspired from the onlooker bee phase of Artificial Bee Colony (ABC) algorithm. In the proposed position updating phase, solutions update their positions based on probability which is a function of fitness. This strategy provides more position updating chances to the better solutions in the solution search process. The proposed algorithm is named as Fitness Based Particle Swarm Optimization (FPSO). To show the efficiency of FPSO, it is compared with standard PSO 2011 and ABC algorithm over 15 well known benchmark problems and three real world engineering optimization problems.

Keywords Particle Swarm Optimization · Artificial Bee Colony · Swarm intelligence · Fitness based position updating · Optimization

1 Introduction

After being inspired from social behavior of fish schooling and birds flocking while searching the food, Kennedy and Eberhart [1], [2] developed a swarm intelligence based optimization technique called Particle swarm optimization (PSO) in 1995. PSO is a population based, easy to understand and implement, robust meta heuristic optimization algorithm. PSO can be a better choice for multi model, non convex, non linear and complex optimization problems but like any other evolutionary algorithm, it also has drawbacks like trapping into local optima[3], computationally inefficient as measured by the number of function evaluations required [4]. These points restrict PSO to less applicability [5]. Researchers are continuously working to achieve these goals i.e., increasing convergence speed and ignoring the local optima to explore PSO applicability. As a result, a huge variants of PSO algorithm have been proposed [6],[3],[7],[8],[9], [4] to get rid of these weaknesses. However, achieving both goals simultaneously is difficult like liang et al. proposed the comprehensive-learning PSO (CLPSO) [3] which aims at ignoring the

Kavita Sharma
Government Polytechnic College, Kota
E-mail: er_kavita28@yahoo.com

Varsha Chhamunya
Gurukul Institute of Engineering & Technology, Kota
E-mail: varshagupta_28@yahoo.co.in

P C Gupta
University of Kota, Kota
E-mail: dr.pcgupta@uok.ac.in

Harish Sharma
Rajasthan Technical University Kota, India,
E-mail: harish.sharma0107@gmail.com

Jagdish Chand Bansal
South Asian University, New Delhi
E-mail: jcbansal@gmail.com

local optima, but results show that it also suffers from slow convergence. Ratnaweera et al. [6] proposed time varying acceleration factors to balance cognitive and social component in initial and later stages. Zhan et al. [7] also tried to adopt the acceleration factors increasing or decreasing depending on different exploring or exploiting search space stages. Zhang et al. [8] studied effect of these factors on position expectation and variance and suggested that setting the cognitive acceleration factor as 1.85 and the social acceleration factor as 2 works good for improving system stability. Gai-yun et al. [9] also worked for self adaption of cognitive and social factors. In this paper, a fitness based position update strategy is proposed in PSO to balance the exploration and exploitation capabilities. Further, the velocity update equation of PSO is also modified to improve the convergence ability.

Rest of the paper is organized as follows: Standard PSO is explained in section 2. In section 3, Fitness based Particle Swarm Optimization (FPSO) is proposed. In Section 4, performance of the proposed strategy is analyzed. Applications of FPSO to engineering optimization problems are explained in section 5. Finally, in section 6, paper is concluded.

2 Standard Particle Swarm Optimization Algorithm

PSO is an optimization technique which simulates the birds flocking behavior. PSO is a dynamic population of active, interactive agents with very little in the way of inherent intelligence. In PSO, whole group is called *swarm* and each individual is called *particle* which represents possible candidate's solution. The swarm finds food for its self through social learning by observing the behavior of nearby birds who appeared to be near the food source. Initially each particle is initialized within the search space randomly and keeps the information about its personal best position known as *pbest*, swarm best position known as *gbest* and current velocity V with which it is moving, in her memory. Based on these three values, each particle updates its position. In this manner, whole swarm moves in better direction while following collaborative trail and error method and converges to single best known solution.

For an D dimensional search space, the i^{th} particle of the swarm is represented by a D - dimensional vector, $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$. The velocity of this particle is represented by another D -dimensional vector $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$. The previously best visited position of the i^{th} particle is denoted as $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$. g is the index of the best particle in the swarm. PSO swarm uses two equations for movement called *velocity update equation* and *position update equation*. The velocity of the i^{th} particle is updated using the velocity update equation given by equation (1) and the position is updated using equation (2).

$$v_{ij} = v_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (p_{gj} - x_{ij}) \quad (1)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (2)$$

where $j = 1, 2, \dots, D$ represents the dimension and $i = 1, 2, \dots, S$ represents the particle index. S is the size of the swarm and c_1 and c_2 are constants (usually $c_1 = c_2$), called cognitive and social scaling parameters respectively or simply acceleration coefficients. r_1 and r_2 are random numbers in the range $[0, 1]$ drawn from a uniform distribution.

The right hand side of velocity update equation (1) consists of three terms, the first term v_{ij} is the memory of the previous direction of movement which can be thought of as a momentum term and prevents the particle from drastically changing direction. The second term $c_1 r_1 (p_{ij} - x_{ij})$ is called cognitive component or persistence which draws particle back to their previous best situation and enables the local search in swarm. The last term $c_2 r_2 (p_{gj} - x_{ij})$ is known as social component which allows individuals to compare themselves to others in it's group and is responsible for global search. The Pseudo-code for Particle Swarm Optimization, is described as follows :

Based on the neighborhood size, initially two versions of PSO algorithm were presented in literature namely, global version of PSO which is the original PSO (PSO-G) and the local version of PSO (PSO-L)[10]. The only difference between PSO-G and PSO-L is that the term p_g in social component in velocity update equation (1). For PSO-G, it refers the best particle of whole swarm while for PSO-L it represents the best particle of the individual's neighborhood. The social network employed by the PSO-G reflects the star topology which offers a faster convergence but it is very likely to converge prematurely. While PSO-L uses a ring social network topology where smaller neighborhoods are defined for each particle. It can be easily observed that due to the less particle inter connectivity in PSO-L, it is less susceptible to be trapped in local minima but at the cost of slow convergence. In general, PSO-G performs better for unimodal problems and PSO-L for multimodal problems.

Velocity update equation in PSO determines the balance between exploration and exploitation capability of PSO. In Basic PSO, no bounds were defined for velocity, due to which in early iterations the particles far from *gbest*, will take large step size and are very much intended to leave the search space. Thus to control velocity so that particle update step size is balanced, velocity clamping concept was introduced. In velocity clamping, whenever velocity exceeds from its bounds, it is set at its bounds.

Algorithm 1 Particle Swarm Optimization Algorithm:

```

Initialize the parameters,  $w$ ,  $c_1$  and  $c_2$ ;
Initialize the particle positions and their velocities in the search space;
Evaluate fitness of individual particles;
Store gbest and pbest;
while stopping condition(s) not true do
  for each individual,  $X_i$  do
    for each dimension  $j$ ,  $x_{ij}$  do
      (i) Evaluate the velocity  $v_{ij}$  using (1);
      (ii) Evaluate the position  $x_{ij}$  using (2);
    end for
  end for
  Evaluate fitness of updated particles;
  Update gbest and pbest;
end while
Return the individual with the best fitness as the solution;

```

To avoid the use of velocity clamping and to make balance between exploration and exploitation, a new parameters called inertia weight [11] was introduced in velocity update equation as:

$$v_{ij} = w * v_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (p_{gj} - x_{ij}) \quad (3)$$

where inertia weight is denoted by w . In subsequent section, the proposed PSO algorithm is explained in details.

3 Fitness Based Particle Swarm Optimization

However the standard PSO has the capability to get a good solution at a significantly faster rate but, when it is compared to other optimization techniques, it is weak to refine the optimum solution, mainly due to less diversity in later search [12]. On the different side, problem-based tuning of parameters is also important in PSO, to get optimum solution accurately and efficiently[13]. In standard PSO velocity update equation (1) contains three terms. The first term has the global search capability, the second and third terms are the particles cognitive and social information sharing capability respectively. More cognitive capability force particle to move towards personal best position fast and more social information force particle to move towards global best position fast. It can be seen from (1), the movement of swarm towards optimum solution is guided by the acceleration factor c_1 and c_2 . Therefore, acceleration coefficient c_1 and c_2 should be tuned carefully to get the desired solution.

Kennedy and Eberhart [1] explained that more value of the cognitive component compared to the social component, results in excessive wandering of individuals through the search space while on the other hand more value of the social component may results that particles will converge prematurely toward a local optimum. These two component play important role for balancing the exploration and exploitation capabilities of PSO. Therefore, in this paper two modifications are proposed for improving the solution search efficiency of PSO.

1. Velocity update equation (refer equation 3) of PSO is modified as follows:

$$v_{ij} = w * v_{ij} + c * r (p_{gj} - x_{ij}) \quad (4)$$

It is clear from equation (4) that the $Pbest$ component (cognitive component) is removed $\{c_1 r_1 (p_{ij} - x_{ij})\}$ from the velocity update equation of PSO. Now the magnitude of velocity of each individual will depend on its distance from the current global best solution. Therefore, this strategy will improve the exploitation capability of PSO.

2. A new position update process, which is inspired from the Artificial Bee Colony (ABC) algorithm's onlooker bee phase [14] is incorporated with PSO. In employed bee phase of ABC, all the employed bees search the food source and calculate their fitness using equation (5):

$$fitness_i = \begin{cases} 1/(1 + f_i), & \text{if } f_i \geq 0 \\ 1 + abs(f_i), & \text{if } f_i < 0. \end{cases} \quad (5)$$

and then in the onlooker bee phase, onlooker bees analyze the available information and select a solution with a probability, $prob_i$, related to its fitness. The probability $prob_i$ may be calculated using equation (6):

$$prob_i(G) = \frac{0.9 * fitness_i(G)}{maxfit(G)} + 0.1, \quad (6)$$

where G is the iteration counter, $fitness_i(G)$ is the fitness value of i^{th} solution and $maxfit(G)$ is the maximum fitness of the solutions in G^{th} iteration. Position update equation of ABC is shown in equation (7):

$$y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (7)$$

where $k \in \{1, 2, \dots, S\}$ and $j \in \{1, 2, \dots, D\}$ are randomly chosen indices, k must be different from i , ϕ_{ij} is a random number between $[-1, 1]$ and x_{kj} is a random individual in the current population. In the basic ABC, at any given time, only one dimension is updated in employed or onlooker bee phase. In onlooker bee phase this update takes place based on a probability which is a function of fitness.

The proposed position update strategy is incorporated with PSO and the newly developed algorithm is named as Fitness based PSO (FPSO). In FPSO, Algorithm 2 is applied after basic PSO operators. The insertion of Algorithm 2 makes FPSO more capable of exploitation in the better search regions. It is expected because in FPSO after applying basic PSO operators, better candidate solutions are offered more chances to update themselves than worse candidates. The pseudo-code of the proposed position update strategy which works after PSO operators is shown in Algorithm 2.

Algorithm 2 Fitness based Position Update Phase:

```

for each individual,  $x_i$  do
  if  $prob_i > rand(0, 1)$  then
     $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$ ,
    Calculate fitness of  $y_i$ ,
    Apply greedy selection between  $y_i$  and  $x_i$ ,
  end if
end for

```

The Pseudo-code for the proposed FPSO algorithm is shown in Algorithm 3.

Algorithm 3 Fitness based Particle Swarm Optimization(FPSO):

```

Initialize the parameters,  $w$ , and  $c$  and  $S$ ;
Initialize the particle positions and their velocities in the search space;
Evaluate fitness of individual particles;
Store the gbest solution;
while stopping condition(s) not true do
  for each individual,  $X_i$  do
    for each dimension  $j$  of  $x_{ij}$  do
      (i) Evaluate the velocity  $v_{ij}$  using (4);
      (ii) Evaluate the position  $x_{ij}$  using (2);
    end for
  end for
  Evaluate fitness of updated particles;
  Update gbest solution;
  /*****Fitness based position update phase in FPSO *****/
   $t = 1, i = 1$  /***  $t$  counts number of updates ***/
  while  $t \leq S$  do
    if  $prob_i > rand(0, 1)$  then
      /*****  $prob_i$  is the probability of an individual  $x_i$  described by equation (6)*****/
       $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$ ,
      { $k, j$  is randomly selected index}
      Calculate fitness of  $y_i$ ;
      Apply greedy selection between  $y_i$  and  $x_i$ .;
       $t = t + 1$ 
    end if
     $i = i + 1$ 
    if  $i > S$  then
       $i = 1$ 
    end if
  end while
end while
Return the individual with the best fitness as the solution;

```



Kavita Sharma <erkavita28@gmail.com>

CIS 2021 notification for paper 323 : Acceptance

2 messages

CIS 2021 <cis2021-0@easychair.org>
To: Kavita Sharma <erkavita28@gmail.com>

Mon, Jul 12, 2021 at 9:51 PM

Dear Kavita Sharma,

Greetings!

Thank you for submitting your research article to the CIS 2021. This is for your information that we received around 700 articles. Most of the papers were very competitive and it was difficult to select only a few papers out of 700 papers.

We are pleased to inform you that based on reviewers' comments your paper has been accepted for presentation during the CIS 2021 and publication in the proceedings to be published by Lecture Notes on Data Engineering and Communications Technologies series of Springer subject to the condition that you submit a revised version as per comments. At the end of this letter, you will find the comments from the reviewers. It is also required that you prepare a response to each comment from the reviewer and upload it as a separate file along with the paper. The similarity index in the final paper must be less than 20%. Please note that the high plagiarism and any kind of multiple submissions of this paper to other conferences or journals will lead to rejection at any stage.

Please carry out the following steps to submit the camera-ready paper and online registration:

1. Register online at <http://cis2021.scrs.in/page/registration-fee>. Click the "Register Here" of the appropriate category box on the registration page.
2. Fill in details and complete the payment process. Note down the transaction ID and take a screenshot of the completed registration window.
3. Convert your paper in Springer template using the instructions given at <http://cis2021.scrs.in/page/paper-submission>
4. Go to the link <https://easychair.org/conferences/?conf=cis2021>. If you want to update any information or the authors click the top right links. To upload the camera-ready paper click on the 'update file' and upload the .zip folder as mentioned in 5.
5. Upload a .zip file in easy-chair containing the following:
 - a. PDF of the revised paper.
 - b. Source files (word or all latex files)
 - c. Reviewers' comments response
 - d. Payment proof (screenshot of the payment or the confirmation email)
 - e. SCRS Membership Certificate if you have registered in the SCRS membership category.

Please note that the Last date for submission of the camera-ready paper, payment of registration fee, and online registration is August 20, 2021.

Feel free to write to "General Chairs, CIS2021" at scrs.cis@gmail.com should you have any questions or concerns. Please remember to always include your paper ID, 323, whenever inquiring about your paper.

Looking forward to meeting you online during the conference.

Team CIS 2021

CIS 2021 <cis2021-0@easychair.org>
To: Kavita Sharma <erkavita28@gmail.com>

Mon, Jul 12, 2021 at 9:53 PM

[Quoted text hidden]

SUBMISSION: 323

TITLE: Fitness based PSO for Large Scale Job Shop Scheduling problem

----- REVIEW 1 -----

SUBMISSION: 323

TITLE: Fitness based PSO for Large Scale Job Shop Scheduling problem

AUTHORS: Kavita Sharma and P.C. Gupta

----- Overall evaluation -----

SCORE: 2 (accept)

--- TEXT:

In this paper, a large-scale job-shop scheduling problem (LSJSSP) is solved using namely fitness-based particle swarm optimization (FitPSO) algorithm. In the proposed solution, a fitness-based solution update strategy is incorporated with the PSO strategy to get the desired results. The paper is very well written and the results analysis part is also convincing. The followings are the minor suggestions:

1. There is some grammatical error in the paper that should be corrected in the final manuscript submission.
2. Parameter setting should be mentioned in the experiment section for the reproduction of the results.
3. The results should be presented in the tabular form with the analysis of the proper results
4. The future scope of the work should be included in the conclusion section.
5. A clear recommendation should be included in the results analysis or conclusion section.

----- REVIEW 2 -----

SUBMISSION: 323

TITLE: Fitness based PSO for Large Scale Job Shop Scheduling problem

AUTHORS: Kavita Sharma and P.C. Gupta

----- Overall evaluation -----

SCORE: 2 (accept)

--- TEXT:

This paper proposed a nature-inspired strategy FitPSO to solve the LSJSSP instances. The paper is well structured but suggests the following modifications in the final manuscript:

1. The recent techniques to solve the JSSP like TLBO, NKPR and DHS should also be taken into consideration for a fair comparison of the results.
2. A detailed results analysis on the basis of Table 4 should be done.
3. Analysis of the results mentioned in Tables 2 and 3 may be elaborated and a clear research finding should be presented.
4. 15 SWV instances, 50 TA instances, and 40 DMU instances are taken for experiments. The results analysis should be specifically instance-oriented.
5. There are some grammatical errors and typos that should be corrected in the revised manuscript.

Fitness based PSO for Large Scale Job Shop Scheduling problem

Kavita Sharma · P C Gupta

the date of receipt and acceptance should be inserted later

Abstract The large-scale job-shop scheduling problem (LSJSSP) is among one of the complex scheduling problems. Researchers are continuously working to deal with the LSJSSP through applying the various probabilistic algorithms which includes swarm intelligence based as well as the evolutionary algorithms even though not able to get the optimum results and it is still an interesting area. Therefore, in this paper a recently developed non-deterministic algorithm namely fitness based particle swarm optimization (FitPSO) is applied to solve the LSJSSP problem instances. In the proposed solution, fitness based solution update strategy is incorporated with the PSO strategy to get the desired results. The obtained outcome is motivating and through results analysis, a confidence is achieved that the proposed FitPSO can be recommendation to solve the existing and the new LSJSSP instance. A fair comparative analysis is also presented which also supports the proposed recommendation.

Keywords Job shop scheduling problem · Fitness based Learning · Swarm Intelligence · Particle Swarm Optimization

1 Introduction

Efficient scheduling is crucial for making the best use of available resources. In the domain of production management, the Large Scale Job-shop Scheduling Problem (LSJSSP) is a complicated combinatorial optimization problem. JSSP needs n jobs to be accomplished on m systems (machines). The system order for all jobs is fixed and varies depending on the jobs. The jobs are put in place in a non-preemptive manner, which means that while one job is running on one system, it cannot be disrupted by another. The primary goal of JSSP is to find an appropriate sequence scheme that reduces the time it takes for all jobs to be completed, which is referred to as makespan (MS). The goal is to minimize the makespan (MS) [9, 35].

The LSJSSP is one of the most important NP-hard problem. To solve LSJSSP, several deterministic conventional mathematical models and heuristic methods have been used. To small size LSJSSP cases, mathematical models have a successful solution in a reasonable amount of time. [1]. The computational time increases exponentially as the size of the instances grows. So, for a larger scale LSJSSP, Non-conventional nature inspired algorithms (NIAs) are preferred alternatives [8]. The numerous processes found in nature are used to create NIAs. Swarm intelligence based algorithms (SIA) and evolutionary algorithms (EAs) are the two main types of NIAs. The design of SIA was influenced by the intellectual actions of creatures. Some state-of-art SIA are Artificial bee colony (ABC)[14], spider monkey optimization (SMO) [5], teaching learning based optimization (TLBO) [26] etc.. EAs like differential evolution

Kavita Sharma (Corresponding author)
Ph.D. Scholar at University of Kota, Kota, India
E-mail: erkavita28@gmail.com

P C Gupta
University of Kota, Kota, India
E-mail: dr.pcgupta@uok.ac.in

(DE) [34], genetic algorithm (GA) [10] etc., are based on biotic transformation like crossover, selection etc.

In recent years, NIAs are performing very well to solve physical world problems [31,30]. In this series, many NIAs emerged well to solve LSJSSP such as genetic algorithm (GA) [11], particle swarm optimization (PSO) [6], hybrid biogeography based optimization (BBO) algorithm [38], hybrid differential evolution algorithm [24], multiple type individual enhancement PSO (MPSO) algorithm [18], classical LSJSSP [28], differential based harmony search algorithm with variable neighborhood search [43], biased random key genetic algorithm [12], new neighbored structure based algorithm [8], teaching learning based optimization (TLBO) algorithm [15], improved ABC (IABC) algorithm [39], discrete ABC (DABC) [40], best so far ABC [4], parallel ABC (pABC) algorithm [3], beer froth ABC [30] etc. In terms of computational time and solution efficiency, the obtained results are acceptable. At the same time, finding a solution for larger JSSP instances is a challenging task. These findings motivate researchers to continue their work in order to solve LSJSSP.

So in this paper, a novel solution is proposed to solve the LSJSSP instances through the recent variant of PSO algorithm, namely fitness based particle swarm optimization algorithm (FitPSO). The FitPSO algorithm was developed by K. Sharma et. al. [29]. In the FitPSO algorithm, a fitness based solution search mechanism is incorporated in the standard PSO. As the FitPSO algorithm efficiently balances the diversification of the population during the solution search process [29]. In this paper the FitPSO algorithm is applied to solve 105 LSJSSP instances. The results are analysed and compared to other important methods available in the literature. The obtained findings substantiate the validity of the proposed strategy.

The remainder of the paper is structured as follows: FitPSO is explained in Section 2. Formulation of LSJSSP is discussed during the section 3. The entire process for solving LSJSSP using the proposed strategy is discussed in section 4. The implementation and experimental results are shown in section 5. Finally, the section 6 summarises the proposed work and suggests future research directions.

2 Fitness Based PSO

PSO is an optimization technique which simulates the birds flocking behavior. PSO is a dynamic population of active, interactive agents with very little in the way of inherent intelligence. In PSO, whole group is called *swarm* and each individual is called *particle* which represents possible candidate's solution. The swarm finds food for its self through social learning by observing the behavior of nearby birds who appeared to be near the food source. Initially each particle is initialized within the search space randomly and keeps the information about its personal best position known as *pbest*, swarm best position known as *gbest* and current velocity V with which it is moving, in her memory. Based on these three values, each particle updates its position. In this manner, whole swarm moves in better direction while following collaborative trail and error method and converges to single best known solution.

For an D dimensional search space, the i^{th} particle of the swarm is represented by a D - dimensional vector, $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$. The velocity of this particle is represented by another D -dimensional vector $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$. The previously best visited position of the i^{th} particle is denoted as $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$. g is the index of the best particle in the swarm. PSO swarm uses two equations for movement called *velocity update equation* and *position update equation*. The velocity of the i^{th} particle is updated using the velocity update equation given by equation (1) and the position is updated using equation (2).

$$v_{ij} = v_{ij} + c_1 r_1 (p_{ij} - x_{ij}) + c_2 r_2 (p_{gj} - x_{ij}) \quad (1)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (2)$$

where $j = 1, 2, \dots, D$ represents the dimension and $i = 1, 2, \dots, S$ represents the particle index. S is the size of the swarm and c_1 and c_2 are constants (usually $c_1 = c_2$), called cognitive and social scaling parameters respectively or simply acceleration coefficients. r_1 and r_2 are random numbers in the range $[0, 1]$ drawn from a uniform distribution.

The right hand side of velocity update equation (1) consists of three terms, the first term v_{ij} is the memory of the previous direction of movement which can be thought of as a momentum term and prevents the particle from drastically changing direction. The second term $c_1 r_1 (p_{ij} - x_{ij})$ is called cognitive component or persistence which draws particle back to their previous best situation and

enables the local search in swarm. The last term $c_2 r_2 (p_{gj} - x_{ij})$ is known as social component which allows individuals to compare themselves to others in it's group and is responsible for global search. The Pseudo-code for Particle Swarm Optimization, is described as follows :

Algorithm 1 Particle Swarm Optimization Algorithm:

```

Initialize the parameters,  $w$ ,  $c_1$  and  $c_2$ ;
Initialize the particle positions and their velocities in the search space;
Evaluate fitness of individual particles;
Store gbest and pbest;
while stopping condition(s) not true do
  for each individual,  $X_i$  do
    for each dimension  $j$ ,  $x_{ij}$  do
      (i) Evaluate the velocity  $v_{ij}$  using (1);
      (ii) Evaluate the position  $x_{ij}$  using (2);
    end for
  end for
  Evaluate fitness of updated particles;
  Update gbest and pbest;
end while
Return the individual with the best fitness as the solution;

```

Based on the neighborhood size, initially two versions of PSO algorithm were presented in literature namely, global version of PSO which is the original PSO (PSO-G) and the local version of PSO (PSO-L)[27]. The only difference between PSO-G and PSO-L is that the term p_g in social component in velocity update equation (1). For PSO-G, it refers the best particle of whole swarm while for PSO-L it represents the best particle of the individual's neighborhood. The social network employed by the PSO-G reflects the star topology which offers a faster convergence but it is very likely to converge prematurely. While PSO-L uses a ring social network topology where smaller neighborhoods are defined for each particle. It can be easily observed that due to the less particle inter connectivity in PSO-L, it is less susceptible to be trapped in local minima but at the cost of slow convergence. In general, PSO-G performs better for unimodal problems and PSO-L for multimodal problems.

However the standard PSO has the capability to get a good solution at a significantly faster rate but, when it is compared to other optimization techniques, it is weak to refine the optimum solution, mainly due to less diversity in later search [2]. On the different side, problem-based tuning of parameters is also important in PSO, to get optimum solution accurately and efficiently[32]. In standard PSO velocity update equation (1) contains three terms. The first term has the global search capability, the second and third terms are the particles cognitive and social information sharing capability respectively. More cognitive capability force particle to move towards personal best position fast and more social information force particle to move towards global best position fast. It can be seen from (1), the movement of swarm towards optimum solution is guided by the acceleration factor c_1 and c_2 . Therefore, acceleration coefficient c_1 and c_2 should be tuned carefully to get the desired solution.

Kennedy and Eberhart [16] explained that more value of the cognitive component compared to the social component, results in excessive wandering of individuals through the search space while on the other hand more value of the social component may results that particles will converge prematurely toward a local optimum. These two component play important role for balancing the exploration and exploitation capabilities of PSO. Therefore, in this paper two modifications are proposed for improving the solution search efficiency of PSO.

1. Velocity update equation of PSO is modified as follows, here w is the inertia weight:

$$v_{ij} = w * v_{ij} + c \times r(p_{gj} - x_{ij}) \quad (3)$$

It is clear from equation (3) that the $Pbest$ component (cognitive component) is removed $\{c_1 r_1 (p_{ij} - x_{ij})\}$ from the velocity update equation of PSO. Now the magnitude of velocity of each individual will depend on its distance from the current global best solution. Therefore, this strategy will improve the exploitation capability of PSO.

2. A new position update process, which is inspired from the Artificial Bee Colony (ABC) algorithm's onlooker bee phase [13] is incorporated with PSO. In employed bee phase of ABC, all the employed bees search the food source and calculate their fitness using equation (4):

$$\text{if } f_i \geq 0, \text{ then } fitness_i = 1/(1 + f_i), \text{ else } fitness_i = 1 + abs(f_i). \quad (4)$$

and then in the onlooker bee phase, onlooker bees analyze the available information and select a solution with a probability, $prob_i$, related to its fitness. The probability $prob_i$ may be calculated using equation (5):

$$prob_i(G) = \frac{0.9 \times fitness_i(G)}{maxfit(G)} + 0.1, \quad (5)$$

where G is the iteration counter, $fitness_i(G)$ is the fitness value of i^{th} solution and $maxfit(G)$ is the maximum fitness of the solutions in G^{th} iteration. Position update equation of ABC is shown in equation (6):

$$y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (6)$$

where $k \in \{1, 2, \dots, S\}$ and $j \in \{1, 2, \dots, D\}$ are randomly chosen indices, k must be different from i , ϕ_{ij} is a random number between $[-1, 1]$ and x_{kj} is a random individual in the current population. In the basic ABC, at any given time, only one dimension is updated in employed or onlooker bee phase. In onlooker bee phase this update takes place based on a probability which is a function of fitness.

The proposed position update strategy is incorporated with PSO and the newly developed algorithm is named as Fitness based PSO (FitPSO). In FitPSO, Algorithm 2 is applied after basic PSO operators. The insertion of Algorithm 2 makes FitPSO more capable of exploitation in the better search regions. It is expected because in FitPSO after applying basic PSO operators, better candidate solutions are offered more chances to update themselves than worse candidates. The pseudo-code of the proposed position update strategy which works after PSO operators is shown in Algorithm 2.

Algorithm 2 Fitness based Position Update Phase:

```

for each individual,  $x_i$  do
  if  $prob_i > rand(0, 1)$  then
     $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$ ,
    Calculate fitness of  $y_j$ ,
    Apply greedy selection between  $y_j$  and  $x_j$ ,
  end if
end for

```

The Pseudo-code for the proposed FitPSO algorithm is shown in Algorithm 3.

3 Job shop scheduling problem organisation

The LSJSSP can be interpreted in following manner: There are a set of n jobs to be processed using m machines. To complete the execution, each job has to be passed through all the m systems in a given predefined sequence. Each job consists of total m operations. To perform operations a job uses one of the machine. When any of the job is executing on any machine it cannot be interrupted by other jobs. The total number of operations are $m \times n$ that are scheduled on m systems [19].

The objective of the LSJSSP is to minimize the total completion time for all the jobs i.e. makespan (MS). Mathematically the problem is stated as :

$$\text{Minimize } MS_{max} \quad (7)$$

where, $MS_{max} = \max(MS_1, MS_2, MS_3, MS_4, \dots, MS_n)$. $MS_1, MS_2, MS_3, MS_4, \dots, MS_n$ are the completion time for all the n jobs. Followings are the constraints for LSJSSP [12]:

- Each system can process at most one operation at a time.
- The completion time of any operation must be a positive integer.
- Precedence relationships among the different jobs must be satisfied.

Algorithm 3 Fitness based Particle Swarm Optimization(FitPSO):

```

Initialize the parameters,  $w$ , and  $c$  and  $S$ ;
Initialize the particle positions and their velocities in the search space;
Evaluate fitness of individual particles;
Store the gbest solution;
while stopping condition(s) not true do
  for each individual,  $X_i$  do
    for each dimension  $j$  of  $x_{ij}$  do
      (i) Evaluate the velocity  $v_{ij}$  using (3);
      (ii) Evaluate the position  $x_{ij}$  using (2);
    end for
  end for
  Evaluate fitness of updated particles;
  Update gbest solution;
  /*****Fitness based position update phase in FitPSO *****/
   $t = 1, i = 1$  /***  $t$  counts number of updates ***/
  while  $t \leq S$  do
    if  $prob_i > rand(0,1)$  then
      /*****  $prob_i$  is the probability of an individual  $x_i$  described by equation (5)*****/
       $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$ ,
      { $k, j$  is randomly selected index}
      Calculate fitness of  $y_i$ ;
      Apply greedy selection between  $y_i$  and  $x_i$ ;
       $t = t + 1$ 
    end if
     $i = i + 1$ 
    if  $i > S$  then
       $i = 1$ 
    end if
  end while
end while
Return the individual with the best fitness as the solution;

```

4 FitPSO for LSJSSP

The FitPSO algorithm is used to solve LSJSSP instances, and the whole method is detailed here. Since LSJSSP is a discrete optimization problem, a solution in the proposed algorithm is a discrete valued vector (representing a potential operation scheduling list). The reordering of jobs for FitPSO is used to estimate each solution in the search field. To generate the discrete valued sequence from a continuous valued vector we have used random key encoding (RKE) scheme [37].

In RKE encoding scheme, first a continuous valued vector is sorted in an ascending order using an integer series from 1 to $n \times m$, where n represents the total number of jobs and m shows the total number of available systems. As each job has to go through m systems for completing its execution so further transformation from this integer sequence is performed using (Integer value mod $n + 1$). The integer series is transformed to operation order sequence using this transformation, and each job index has m occurrences. Figure 1 depicts the transformation of a continuous valued vector into a discrete valued vector, followed by an operation scheduling sequence. Our goal is to find an operation sequencing list (a vector of discrete values) that decreases the makespan value. The goal is to figure out a series of operations that reduces the overall time it takes to complete all of the jobs. The detailed procedure is described in the subsequent steps:

4.1 Step 1:

The parameters of the proposed FitPSO algorithm namely, cognitive, social scaling parameters (c_1, c_2), Inertia Weight (w), total members of population (solution agents), and total number of iterations are initialized. Each solution agent is initialized in the search space in a uniformly distributed way. As all the initialized sources are continuous in nature so RKE scheme is used to generate the corresponding discrete valued operation sequence. Now the MS value (objective value) for each operation sequence is calculated.

Continuous valued solution	0.9	0.6	0.8	0.2	0.5	0.3
Decoded as	6	4	5	1	3	2
Operation sequence	1	2	3	2	1	3

Fig. 1: Random Key (RKE) Encoding Scheme

4.2 Step 2:

As the step 2 and 3 are iterative steps, the solutions refined themselves in these steps to get the optimum solution. In step2, all the solution agents update themselves using the standard PSO algorithm. The updated solution agent is in continuing form, so again RKE encoding scheme is applied to alter this continuous valued solutions in to corresponding discrete operation sequence list. The MS value for this operation sequence is computed. The *pbest* and *gbest* solutions are updated on the basis of the MS value.

4.3 Step 3:

In this step, the probability for all the solution agents are assessed using the equation 5. This probability will help to decide that which solution is high fit than the other solutions in the swarm. The solution agents are chosen and updated as per the equation 5. Again the solution agent is updated based upon the information obtained from the neighbouring solution agents. To obtain the corresponding operation sequence, RKE scheme is applied on the produced continuous valued solution agent. The *MS* value is computed from the generated operation sequence and the *pbest* and *gbest* solutions are updated.

4.4 Step 3:

In this step, the best solution found so far is memorized (*gbest* solution). Thus obtained solution is termed is the optimum solution generated by the FitPSO.

The pseudo-code of the designed approach for LSJSSP is shown in Algorithm 4.

5 Implementation and experimental results

To prove the effectiveness of FitPSO algorithm, it is applied on LSJSSP instances. Following 105 LSJSSP instances are considered for experimentation [36, 7, 33].

- 15 SWV instances
- 50 TA instances
- 40 DMU instances

To attain the least *MS* value for all these 105 LSJSSP instances is the main goal. The experimental setting is listed as below:

1. Number of run =10
2. Number of maximum iteration =2000
3. Number of solution agents TSA =50
4. Dimension D = Number of systems × Number of jobs
5. Inertia weight $w = 0.8$,
6. Acceleration coefficients $c = c_1 = c_2 = 0.5 + \log_2$ (for PSO)[17],

Algorithm 4 FitPSO algorithm for LSJSSP

```

Parameter Initialization
Total solution agents =  $TSA$ .
D (Dimension) =  $m \times n$ 
Total generation count =  $MGN$ 
CurrentIndex=1.
Step 1: Random initialization of the Solution members in the search space.
Conversion of continuous valued solution agents into an operation sequence (discrete valued solutions) to deal LSJSSP using RKE scheme.
 $MS$  value Computation for every operation sequence.
While (CurrentIndex  $\leq$   $MGN$ ) do
- Step 2 PSO stage:
- for each individual,  $X_i$  do
  for each dimension  $j$  of  $x_{ij}$  do
    (i) Evaluate the velocity  $v_{ij}$  using (3);
    (ii) Evaluate the position  $x_{ij}$  using (2)
  end for
- end for
- Obtain the new discrete operation sequence from the recently revised continuing solution agents using RKE scheme
-  $MS$  value computation for recently produced operation sequence.
- Evaluate fitness of updated particles
- Update the respective pbest solutions
- Update gbest solution
- Step 3 FitPSO stage:
- Probability  $prob_i$  computation using equation 5 for each solution agent.
-  $t = 1, i = 1$  / $***t$  counts number of updates  $***$ /
- while  $t \leq S$  do
  if  $prob_i > rand(0, 1)$  then
     $y_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$ .
    { $k, j$  is randomly selected index}
    Calculate fitness of  $y_i$ 
    Apply greedy selection between  $y_i$  and  $x_i$ .;
    Obtain the new discrete operation sequence from the recently revised continuing solution agents using RKE scheme.
     $MS$  value computation for recently produced operation sequence
    Evaluate fitness of updated particles;
    Update the respective pbest solutions;
     $t = t + 1$ 
  end if
   $i = i + 1$ 
  if  $i > S$  then
     $i = 1$ 
  end if
- end while
- Update gbest solution
- Step 4 Memorize the best solution found so far.
- CurrentIndex=CurrentIndex+1
end while
Output the best solution;

```

The parametric ambience for the FitPSO approach and the other considered approaches are kept same in terms of swarm size and maximum number of iterations to carry out an equitable comparison.

The reported results of FitPSO are compared with the following state-of-art algorithms available in the literature:

- Biased random key genetic algorithm (BRKGA-JSP) [12]
- A guided tabu search for LSJSSP (NKPR) [20]
- Teaching learning based optimization method (TLBO) [25]
- Differential based harmony search (DHS) algorithm [43]
- A tabu search to solve LSJSSP (TS) [42]
- An advanced tabu search algorithm for LSJSSP (i-TSAB) [21]
- AlgFix [23]
- A tabu search/simulated annealing algorithm for LSJSSP (TS/SA) [41]
- Global equilibrium search technique (GES) [22]

The obtained results for the above three instances are represented in Tables 1 to 3. These tables list the name of the instance, its size, the lower bound (LB), the upper bound (UB) for the best known solution (BKS), the BKS obtained by FitPSO approach, and BKS value obtained from the compared algorithms. The obtained results for all the instances demonstrate that the proposed FitPSO is superior approach in reference to MS value during assessment with other considered approaches.

Further to analyse the outcomes, average relative percentage error (RPE) is also calculated and compared as tabulated in Table 4. The value of RPE is computed (with respect to the UB value of an instance) as per demonstrated in equation 8.

$$RPE = 100 \times (BKS_{algo} - UB) / UB \quad (8)$$

Here, BKS_{algo} represents the MS value obtained using the considered approaches. The attained outcomes of Table 4 demonstrate the significant improvement in the average RPE which assures the authenticity of the introduced approach.

6 Conclusion

This article proposed a solution to solve 105 large scale instances of job shop scheduling problem (LSJSSP) using an efficient fully informed artificial bee colony (FitPSO). In FitPSO algorithm, to balances the diversification in the swarm during the solution search process, a fitness based strategy is incorporated in the standard PSO algorithm. The MS time is used as an evaluation criterion in the LSJSSP. The results are analysed and compared to cutting-edge techniques proposed by a number of researchers. According to the results of the experiments, the proposed solution gives better solution. In Future, some more performance metrics may be considered for experimentation.