DOI: http://dx.doi.org/10.26483/ijarcs.v13i4.6887

Volume 13, No. 4, July-August 2022

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

RELATIONSHIP BETWEEN CBS QUALITY PARAMETERS FOR ASSESSMENT OF COMPUTATIONAL INTELLIGENCE TECHNIQUES

Shivani Yadav
Department of Computer Science and Application
Maharshi Dayanand University
Rohtak, India

Bal Kishan
Department of Computer Science and Application
Maharshi Dayanand University
Rohtak, India

Abstract: Software reliability plays a vital role in the emerging field of digitalization. Everyone wants cost and time-efficient software along with reliability which is achieved using CBS. In CBS, if the individual components are computed for a large or complicated system, then integration becomes complex which results in difficulty in predicting CBSR. To solve this problem several computational intelligence techniques such as SVM, ACO, PSO, ABC, GA, Neural network, are used but in our paper, we have focused on optimization techniques Fuzzy, ACO, ABC, PSO. These techniques help in estimating and predicting reliability models for CBS. Also, we have done, an assessment and comparative analysis based on a literature review of ABC, ACO, and PSO that have also been presented, for choosing suitable parameters for software reliability modeling.

Keywords: Reliability, ACO, ABC, PSO, assessment

1 INTRODUCTION

The evolution of technology has led to the increasing complexity and size of software systems. As a consequence, the earlier approach of designing software from scratch has become inefficient from cost, quality, and productivity perspective. This has necessitated the need for different development methodologies that are reusable, flexible, and reliable. One such methodology is Component-based software development. It focuses on the creation and use of individual components that are independent and reusable. Individual components with different functionalities are integrated to develop software. This approach offers the advantages of cost efficiency, time efficiency, increased reliability, modularity, and performance.

According to ISO/IEC 9126-1, the quality of software systems is estimated parameters like functionality, efficiency, testability, usability, reliability, etc. Reliability is one such critical aspect and can be defined as the ability of software to tolerate faults during the lifetime of its use. Apart from ISO/IEC 9126-1, some other reliability evaluation models like Boehm's Model, FURPS model, and Dromey's model have also been developed. Most of these models focus on common parameters like fault tolerance, reliability compliance, recoverability, and maturity for reliability estimation. Software reliability focuses on three main activities:

1. Error prevention
2. Fault detection and removal
3. Other reliability increasing measures

The traditional reliability estimation methods focused on software as a single monolithic structure. Hence, it is difficult to apply the same principles as these ignore the interactions between components of a software system in a Component-Based Software (CBS) system. The traditional methods are unable to incorporate the operational and integration uncertainty of a CBS environment. This requires the inclusion of factors like an individual component failure, component behavior, and interaction between components. The interdependencies between components increase system complexity leading to difficulties in reliability estimation. Hence, CBS systems emphasize on two major elements:

1. Individual component reliability
2. Integrated system reliability

1.1 Individual component selection and component assembly

The process of component selection is used to determine the 'fitness' of an existing component for use in a new system. This problem arises when some components with the same functionality already exist and the developers need to choose one that best suits their requirement. A systematic approach for component selection was proposed by Wanyama and Homayoun:

1. Defining the evaluation criteria, including functionality and component interaction, after incorporating stakeholder requirements.
2. Searching COTS products
3. Applying a filter to search results using the set of stakeholder requirements to develop a list of suitable COTS components.
4. Evaluating shortlisted COTS components in detail, including their properties and other criteria-based assessments
5. Analyzing evaluation results and selecting the COTS components with the best fit

Component assembly

The process of component assembly involves the integration of components according to a defined architecture. The architecture defines how independently developed components can be integrated to work coherently while fulfilling requirements. Sometimes, the architect may not be able to access the source code and needs to work with only the interfaces supplied by the component developer. Hence, assembly becomes a challenging task with uncertainties about

interaction patterns, behaviors, and unexpected results arising out of complex integration.

The architecture used for component assembly is one of the most important factors while designing real-world applications, as even best-designed components may not be able to deliver a world-class system if the architecture isn't able to utilize them cohesively. However, the benefits offered by reliable, reusable components are still helpful in reducing the design, implementation, and deployment time for a software system.

The reliability of large software systems relies on interconnection parameters like data translation, resource sharing, and synchronization. As the size of software systems grows, developers and architects need to identify and handle these dependencies to resolve the complexity and mismatches. Most programming languages haven't recognized these interconnection issues and the need to separate functional and integration aspects of components. Integration techniques should be able to handle software reconfiguration dynamically and flexibly [1].

1.2 Factors affecting the reliability of CBS

The performance of a reliability estimation model is contingent on its ability to consider the appropriate parameters for measurement. Hence, we have curated a list of relevant factors that affect the reliability of CBS systems, after careful consideration of available research in the field and unique characteristics of CBS systems.

1. Complexity: Complexity of an application is closely tied to the complexity of individual components in CBS systems and their interdependencies. It increases with the increase in number of components invariably and it becomes difficult to estimate reliability. This can lead to unreliable, cumbersome systems and causes a drag on operations of organizations that manage multiple applications with a multi-layered infrastructure.
2. Reusability: As individual components are reused across multiple systems with slight or no modifications, the component becomes more reliable due to repeated testing of the component throughout the development of multiple systems. Components with high cohesion and low coupling are highly recommended for reuse. It can be concluded that higher reusability results in higher quality and performance predictability.
3. Flexibility: This can be viewed as the software's ability to adapt to conditions and requirements and can be measured as the number of changes that can be made without altering the basic functionality of the software. This plays a major role in CBS development as existing components are often reused with slight modifications to adapt to the requirements of different systems. The higher the flexibility of a component, the lower is the maintenance cost. Flexibility also safeguards against future changes in architecture and requirements, thereby increasing reliability.
4. Inter-operability: A component can communicate with other components and share information without major user intervention. As the technology industry develops new tools and technologies at a rapid pace, this is increasingly important for the seamless integration of components. It is directly related to the cost of a system and can be determined by the interfaces and communication methods used for component interactions [2].

From above, we can conclude that number of reusable components is vital to Component-Based Software Reliability (CBSR). In CBS systems, if the individual components are computed for a large or complicated system, then integration becomes complex which results in difficulty in predicting CBSR. To solve this problem several computational intelligence techniques such as SVM, ACO, PSO, ABC, GA, Neural network, are used. These techniques help in estimating and predicting reliability models for CBS.

The usefulness of a reliability model is dependent upon a number of factors, including the methodology used and parameters considered for evaluation. It is imperative to identify what methodology is most effective for different types of software systems. A reliability prediction model that worked well for traditional monolithic development approaches is not suited for reliability estimation of software developed using component-based software development approach. Soft computing based reliability estimation models have shown promising results for small and large scale systems in computer, medical and mechanical software. Further, within models used for CBS reliability prediction, there is a need to identify the right parameters that have a larger impact.

A literature review of reliability prediction using soft computing methods shows that it is being increasingly used for CBS development. Soft computing algorithms like Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Fuzzy logic have shown promising results while increasing response time and reducing errors. Further research on estimating Component-based software reliability (CBSR) using soft computing techniques may help in improving the understanding of relationship between software reliability and component-based factors like complexity, reusability, dependency etc.

2 RELATED WORK

Section 2 is focused on review of previous work carried out by researchers in the field of various optimization approaches for the estimation of reliability. In the field of CBS, there is a dearth of successful reliability prediction models, to get rid of this problem, computational intelligence techniques are introduced to get accurate results. Some of the previous work of researchers is shown below based on computational techniques like Fuzzy Logic, ACO, ABC, PSO, GA, etc.

Diwaker et al. [3] assessed different computational techniques such as Neural-Network, Fuzzy Logic, Particle Swarm Optimization, Genetic Algorithm, Support Vector Machine, Ant Colony Optimization, and Artificial Bee Colony and their ability to predict reliability using various parameters. According to the author, the concepts discussed can be used to predict the reliability of software as well as hardware.

Diwaker et al. [4] suggested a new, soft computing based model for predicting the reliability of Component-based software using parallel and series reliability models. The model is evaluated against Fuzzy Logic and Particle Swarm Optimization. The authors show that their proposed model

can predict reliability with a lower error rate than Fuzzy Logic and Particle Swarm Optimization.

Wolski et al. [5] proposed a quality measurement framework based on Boehm and McCall models within GEANT research and innovation project. The project was started and funded by the EU to make it self-financing in the future. The framework emphasizes the reuse of existing data and takes an innovative, research-oriented perspective to projects while applying specific characteristics of the networking environment. Authors' hybrid approach of combining process and product-related measurements stems from basic quality models and enables comparison between internal and external projects in a broad view.

Singh et al. [6] used object-oriented (OO) metrics, given by Chidamber and Kemerer, to examine ANN's applicability for software quality prediction. A testing effort was predicted using ANN and publically available NASA data was used to find testing effort's relationship with object-oriented metrics. In more than 72.5% of the cases, an estimated testing effort was within 35% of actual effort, with a MARE of 0.25. According to the author, studies with large data sets need to be carried out to determine the model's relevance.

Khoshgoftaar et al. [7] focused on neural network-based software quality models by training two neural networks, one with the set of components selected by multiple regression model selection and other with the entire set of principal components. A big commercial system was used to select multiple regression quality models from principal components of software metrics. According to the author, two quality measures were selected from five software systems for comparing the models and understanding the relationship between software quality and complexity metrics.

Sedigh-Ali et al. [8] suggested a graph-based model for component selection from a family of components. The system formed from the selected components should satisfy non-functional requirements. According to the author, it was ensured by identifying the set of components which can together provide the best tradeoff among desire metrics. It was found to minimize uncertainty in the cost and quality of component-based systems.

Feurer et al. [9] used Bayesian optimization to develop an Automated Machine Learning (autoML) framework for data and feature preprocessing, algorithm choice, and hyperparameters tuning. Existing autoML methods were refined for robustness using i) a meta-learning component that uses past data sets for Bayesian optimization and ii) an ensemble construction component for combining the most suitable methods from Bayesian optimizer. The meta-learning feature ensures that the system improves over time as the number of datasets increases. According to the author, the proposed framework outperformed an existing autoML system in most cases and different variations of the framework with and without the two key additions showed that meta-learning had a bigger impact on optimization.

Mendoza et al. [10] developed AutoNet for providing feed-forward neural networks that could tune itself automatically. According to the author, results obtained from combining Auto-sklearn and Auto-Net were better than when using each of them alone.

Sagar et. al [11] defined reusability metrics for black-box components in component-based development by identifying relevant factors and their relationships. According to the author, reusability was estimated using Fuzzy logic on real-time applications. Developers can reduce maintenance efforts by using highly reusable components.

Sangwan et al. [12] described a model based on soft computing for measuring software reusability levels. They used soft computing techniques like neural networks, fuzzy logic, and neuro-fuzzy. The model used four parameters, namely i) Interface Complexity, ii) Understandability of software, iii) Documentation quality, and iv) Changeability. According to the author, the model trained using the neuro-fuzzy technique predicted good results with MARE 22% and MRE 0.007% in comparison to purely fuzzy logic or purely neural network-based techniques.

Diwaker and Tomar [13] used metrics like efficiency, dependency, and density of components for an assessment of the Ant Colony Optimization methodology to determine reusable components that lead to increased reliability. According to the author, MATLAB results showed that increase in component efficiency was accompanied by increase in component density.

Diwaker and Tomar [14] defined a Particle Swarm Optimization based fitness function using functionality, average execution time, interface complexity, which can be considered as CBS system metrics. According to the authors, sub-parameters like interaction among components, reusability, and resource usage were also used.

Diwaker and Tomar [15] evaluated Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) based models, using CBS system reusability metrics to determine the approach which yields best results. According to the author, parameters like the number of functions, lines, and reusable components were used for evaluation using MATLAB.

3 COMPUTATIONAL INTELLIGENCE TECHNIQUES

Several computational techniques are used these days to get an accurate and cost-efficient result with CBSE. For solving large and complicated problems, optimization applications are incorporated with optimization techniques such as Fuzzy logic, SVM, ABC, ACO, and PSO. This paper mainly focuses on the introduction of techniques fuzzy logic, ACO, ABC, PSO, and their assessment concerning CBSRe

Fuzzy Logic

Fuzzy Logic when combined with mining helps in predicting software reliability. The working of fuzzy logic is divided into 4 four parts: "fuzzifier, inference engine, rules, and defuzzifier". Fuzzy logic represents the analytic result based on faults depicted in failed software. The system takes faulty data from faulty software as input data to predict faults in the future as output. In predicting software reliability, fuzzy logic plays a vital role as these models can be applied with ease at different complex stages with varied failed data in the form of sets [16].

Ant Colony Optimization

The ant colony algorithm was proposed by Dorigo[35] to optimize the problems using a real example of ants. Ants' life was considered due to their extraordinary ability in searching for food at the nearest location and they traverse that path with releasing one chemical named as a pheromone. Pheromone helps other ants to reach the same destination by

traversing the same path. Using this concept, researchers have applied many problems to get optimize solutions. Ant colony algorithm is applied to many concepts like TSP, network model problem, graph coloring problem, image processing, VRP, etc [17,18,19].

Artificial Bee Colony Algorithm

Karaboga[20] introduced the concept of an artificial bee colony algorithm that resembles a real bee's role in "foraging". ABC algorithm follows their own rule and duties are provided to all the groups of bees which are classified into employed, onlooker, and scout bees. All these bees perform different tasks like, employed bees handle the process after a new source identified by the scout bees and the onlooker bees are attracted by employed bees through dance to company them for food exploitation. These bees change their duty intelligently according to the hive condition. ABC algorithm uses the concept of fluctuation, negative feedback, positive feedback, and multiple-interactions. The location of food correlates to feasible results to solve the problem of optimization. Hence, the algorithm helps in achieving optimum results through classified bees in search space and this algorithm is very helpful in predicting software defects [21].

Particle Swarm Optimization (PSO)

Particle Swarm Optimization technique was proposed in 1995, by Doctor Eberhart and Kennedy. It is a heuristic global optimization method based on bird and fish flock movement behavior research.

In their search for food, birds travel from one place to another, either as a group or in a scattered manner. The bird with a good perception of food smell can locate the place where food can be found and transmit it to the rest of the flock, who converge at the food location. Applying the same principles for developing the particle swarm optimization algorithm, a model was developed, where bird movement between spaces corresponds to the solution swarm. In such a scenario, useful information is considered equivalent to optimal local solution and food equals global optimal solution. The solution swarm so obtained, is then compared to bird swarm and global optimal solution is worked out using PSO through cooperation of birds. PSO model has been found useful in solving complex optimization problems in the field of model classification, neural network training, signal processing, vague system control, machine study, automatic adaptation control etc., due to its ease of implementation and relative simplicity [22].

Table 1 indicates the main computational intelligence techniques which are used by researchers and practitioners for predicting the reliability of CBS. As shown in Table 1, ACO, PSO, ABC, and fuzzy logic are used for predicting CBSRe due to the utilization of CBSE factors. Therefore, there is a need to assess the optimization techniques such as ACO, PSO, ABC, and Fuzzy logic. The assessment of these techniques provides the component interface, components integration, and reusability in Fuzzy logic, PSO, ACO, and ABC.

Table 1 Assessment of Computational Intelligence Techniques to Predict CBSRe

S. No.	Computational Intelligence Techniques	Parameters Used
1	Fuzzy Logic	KDLOC, effort multipliers, performance, fault density, usability, serviceability, availability, adaptability, maintainability, capability, interface complexity
2	PSO	Reliability, fitness value, component interface complexity, average execution time, computational time
3	ACO	Probability, time interval, number of intervals, ants, failure rate, fitness function
4	ABC	Adaptability, computational time, waggle dance

The attributes like component interface, components integration, reusability in any techniques provide an idea for the suitability for predicting software reliability.

4 ASSESSMENT OF PSO

In this part, the evaluation of PSO is evaluated by utilizing CBSE metrics. Numbers of cycles/iterations of elements are produced or updated with considered fitness capacity that assists in discovering optimal values. Few CBSE metrics are contrasted that are well-matched for examining PSO with new fitness capacity and Standard PSO assessment of enhanced algorithm is done on MATLAB.

In PSO, every element comprises of its nearby best cost and locality cost figured by wellness capacity. An effort has been prepared to determine the cost of elements of PSO by utilizing CBSE measurements. The associated CBS reliability measurement has already been discussed in the previous section and metrics like Average Execution Time (AET) and Degree of Reuse of Inheritance Methods (DRIM) can also be utilized for assessment of PSO.

Table 2. A Comparison between Various CBSE Metrics w.r.t PSO

Metrics Used / Optimization Techniques	CICM	AET	CFM	DRIM	CRUM	CSM	CCM	CRM
PSO	√	√	√	X	X	X	X	√

The utilization of AET is to discover the normal carrying out time of elements cooperation [23]. The degree of Reuse of Inheritance Methods/ the Proportion of Potential (PP): It characterizes the proportion of potential techniques utilizes factual reused techniques [24]. Table 2 shows a relative study of different metrics that are well-suited with PSO. This analysis is based on the work of different researchers presented in their work. They provide various case studies for the description of each metric. From Table 3 CICM, AET, CFM is chosen by analyzing different real case studies for analysis.

Table 3 shows the suitability of the interaction of elements that support PSO. Various costs are monitored using the imitation of PSO by using CICM, CFM, and AET metrics.

Table 3 Analysis of CBSE Metrics with Various Parameters

Metrics Used	Reusability	Interface of Components	Complexity	Resource Utilization
CICM	High	High	High	High
AET	-	Low	High	Medium
CFM	-	High	High	-

Table 4 presents the most select normal cost of these measurements monitored for the above-mentioned measurements. In Table 4, various costs of suitable metrics have been purposed to determine the element interface based on the past study made in this area.

Table 4 Evaluation of Modified PSO with CICM, CFM, and AET Metrics

Metrics Used	Standard PSO	PSO utilizing new Fitness Capacity
CICM	51%	58.5%
CFM	65.2%	74.8%
AET	73.3%	79.21%

Table 5 Values of Multiple Iterations of PSO

CICM	Standard PSO	PSO utilizing New Fitness Capacity
Generation1	16000	16038
Generation 2	16800	17254
Generation 3	17025	18452
Generation 4	17078	18490
Generation 5	21410	21492

CICM: Table 5 presents the cost of Standard PSO and modified PSO by using new wellness capacity with CICM generations.

Figure 1 presents, when new wellness capacity is considered for PSO, and then the amount of interface of elements is improved, the outcome in raise in reliability.

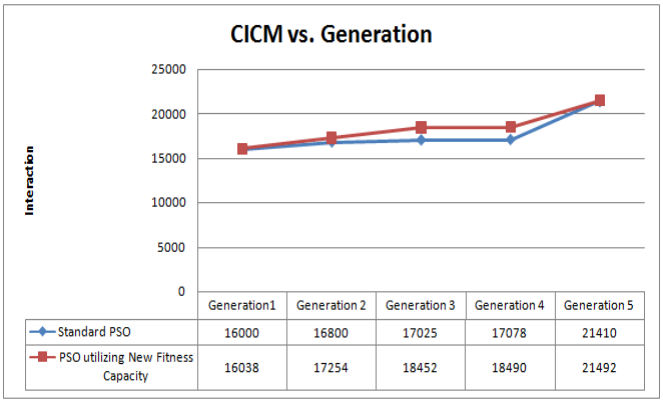

Figure 1 CICM vs. Generation

CFM: Table 6 presents the cost of Standard PSO and changed PSO utilizing new wellness with different CFM cycles.

Table 6 Different Cycles for CFM among Standard PSO and Modified PSO using New Wellness Capacity

CFM	Standard PSO	PSO utilizing New Fitness Capacity
Generation1	2999	3320
Generation 2	3292	3774
Generation 3	3345	4113
Generation 4	3398	4123
Generation 5	4112	4157

Figure 2 shows that the precision of elements interface is soaring in the planned proposal, which assists in reliability approximation.

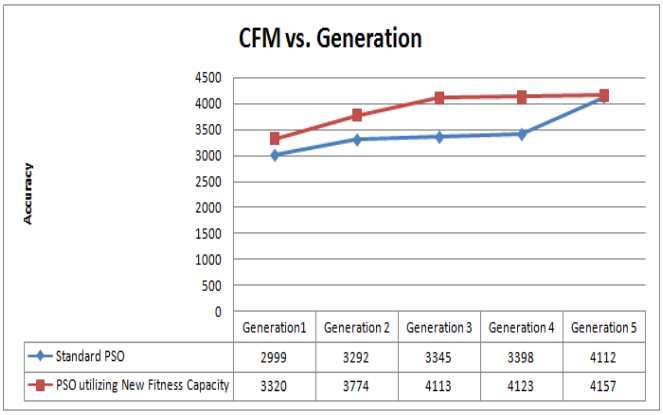

Figure 2 CFM vs. Generation

AET: Table 7 presents the cost of both Standard PSO and changed PSO with modified wellness capacity utilizing AET generations.

Table 7 Number of Generations for AET with Standard PSO and Modified PSO using Modified Fitness Capacity

AET	Standard PSO	PSO utilizing New Fitness Capacity
Generation1	468.27	511.38
Generation 2	551.31	592.28
Generation 3	585.64	751.23
Generation 4	722.33	771.51
Generation 5	738.62	742.50

Figure 3 shows that AET provides a higher value when modified PSO is evaluated. The execution time increases by increasing accuracy.

PSO having a new wellness capacity utilizes few CBSE measurements that hold up the interface between elements, reusability of elements, and successful consumption of resources by using new wellness capacity.

The outcome demonstrates that the changed proposal has provides improved outcomes as contrasted with established PSO, as CBSE measurements are utilized for evaluation. In advance research, CBSE measurements can be utilized for assessment of various soft computing methods for assessment of CBS reliability.

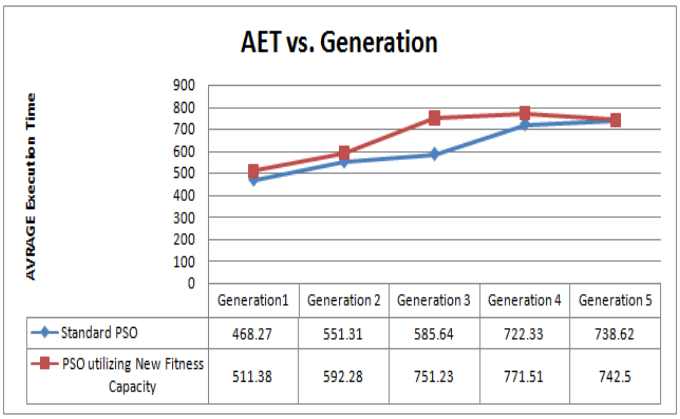

Figure 3 AET vs. Generation

5 ASSESSMENT OF ACO

One of the challenges in Component-based Systems (CBS) is finding an optimal path between the components. One such method that has been used to determine optimal path between components is a soft computing technique called Ant Colony Optimization (ACO). Metrics like component dependency and component density are pivotal in the assessment of ACO. Results have shown the efficiency of component interaction when ACO is used for optimization. Hence, it is suggested that real data sets be used in the future for proposing a new reliability framework.

The efficiency of Component: Efficiency can be represented as the number of required components divided by total number of components. The improvement achieved for optimal selection of retrieved components, with target search, by using ACO can be seen in Table 8 and Figure 4.

Table 8 Efficiency Value with and without ACO

S. No.	Search Value	Efficiency with ACO	Efficiency without ACO
1	5	75%	62%
2	8	71%	58%
3	9	55%	19%
4	4	31%	49%
5	10	24%	8%

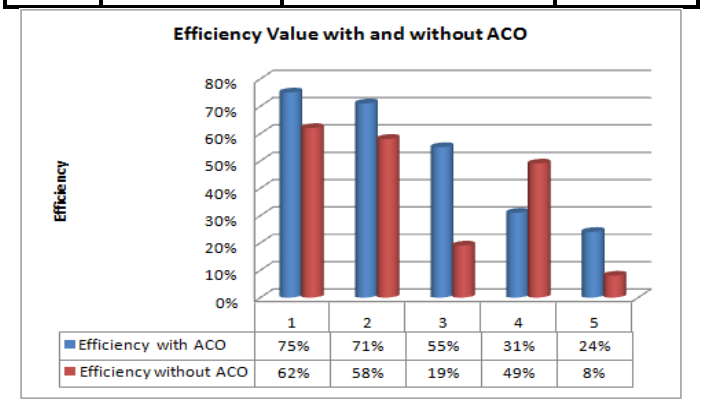

Figure 4 Efficiency vs. Generations

The results predict that after applying ACO the efficiency of interaction between components increases.
Component Density: It is the ratio of a definite number of interfaces to access a number of interfaces.

Table 9 Component Density Values

S. No.	Without using ACO	With using ACO
1	0.2447	0.4264
2	0.2625	0.4626
3	0.2753	0.5101
4	0.2615	0.5589
5	0.2743	0.5327

Table 9 and Figure 5, shows that the component density is decreased when the interfaces between components is low.

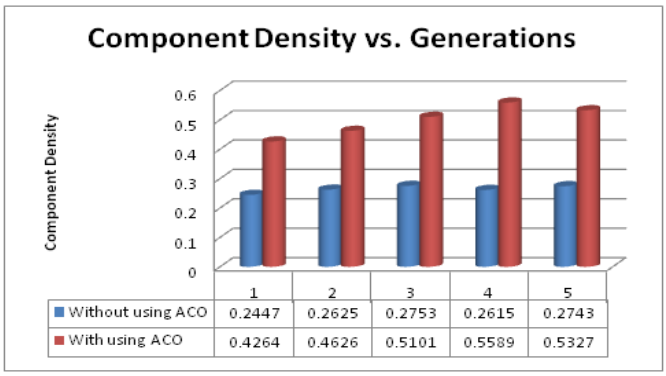

Figure 5 Component Density vs. Generations

Component Dependency: The dependency represents the influence of one component interaction to another. An individual component can be used as a function/module which is needed at the time of component/system integration.

6 COMPARISON OF ASSESSMENT OF ACO, ABC and PSO

Table 10 below shows the comparison between ACO, ABC and PSO over different parameters

Table 10 Comparison of PSO, ACO, and ABC

Parameters	Optimization Techniques		
	PSO	**ACO**	**ABC**
Movement	Movement in search space for optimal solution	Activities of ant colonies.	Activities of bee colonies
Selection of Path	Global optimal value for each particle is taken as the optimal solution.	Based on the quantity of Pheromone on track. The path with the highest quantity is selected.	Waggle Dance (WD) provides both quality and direction of food.
Enrollment Methods	Indirect - Particle velocity is calculated using fitness function	Indirect – Use of pheromones by Ants based on food type and quantity	Direct – WD gives target direction and distance
Navigation Method	Random walk to the target and collection of path details	Random walk while laying down pheromones	Random walk and collection of path details
Adaptability	Less adaptive	More adaptive	Less adaptive
Computational Time	Higher than ABC	Higher than ABC	Least out of the three
Steps Requirement for Computing Result	Requires more Steps than ABC.	Requires more steps than ABC.	Least number of steps out of the three
Scalability	Least scalable.	More scalable.	More scalable.
Advantages	i) Less calculations vis-à-vis other methods. ii) Provides choice of fitness function selection for minimization and maximization.	i) robustness ii) distributed computation avoids premature convergence i) Natural parallelism. ii) The quick finding of the good quality result. iii) Can be applied to components with difficult to predict behavior	Team job: Scout and hunter bees work jointly for getting healthy food.
Limitations	i) Lower accuracy owing to difference in particle motion and direction ii) Difficult to use in non-coordinate and contact scattered particle environment	i) Early convergence ii) identification of design parameters is difficult	i) WD mapping to outcome is complex ii) Prior knowledge is required about some factors
Applications	Telecom, Power Systems, Signal Processing, Combinatorial Optimization etc.	Scheduling problems, Assignment problems, vehicle routing, TSP, image processing, network model problem	TSP, Planning, Spam identification, etc.

Each optimization technique comes with its own set of costs, contingent upon factors like fitness functions, iterations, number of interactions etc. A mathematical solution to calculate this cost is [27]:

$$Cs = Cnr - Cr \quad \text{…. eq 1}$$

Where,
Cs = saving cost or mean cost,
Cnr = software development cost without reusable components, and
Cr = software development cost with reusable components

Table 11 show data about lines of code (including duplicate statements and multi-usage functions), functions which have been used repeatedly and functions with single time usage.

Table 11 LOC and Function used in Optimization Techniques using MATLAB

Optimization Techniques	Line of Code	Function used Repeatedly	Function not used Repeatedly
PSO	232	8	4
ACO	123	9	5
ABC	142	7	4

Table 12 summarizes the parameters that have been used for comparison of ABC, ACO and PSO. Iterations can be increased or decreased according to the requirements.

Table 12 Simulation Parameters

No. of Iterations	150
Optimization Techniques	ABC, ACO and PSO
Simulation Time	25000 seconds
Fitness Function Used	$Y= x_1^2- 3x_2+10$ Where $0<=x_1, x_2<=8$
Operating System	Windows 10
Platform	Matrix Laboratory 2009 v2

6.1 Evaluating the Best Cost of PSO and ACO algorithm

The following work compares the performance of PSO, ACO, and ABC by simulating these techniques w.r.t the best cost and number of iterations.

The best cost presents the mean cost as mentioned in the reusability metric that is estimated using a line of code and functions used in the program for several iterations as shown in Table 12. From the analysis, it can be observed that ACO shows higher reusability factor as compared to PSO and ABC. The MATLAB output shows the relationship between iteration and cost. It can vary depending upon the analysis required to obtain optimal results.

It depends on the user fitness function and user requirement that whether ACO is better or PSO is better. When it is necessary to complete the problem or solve a problem within the time then ACO may be chosen but it affects the reliability of components. If the problem is on a large scale then the PSO algorithm can show better performance compared to other techniques. CBSE reliability factors can be utilized using PSO. The momentum effect on particle movement in Particle Swarm Optimization (PSO) provides variety in search trajectories along with faster convergence. It has been observed that Fuzzy Logic and PSO provide minimum error results when the space is small, and may be used for further research in reliability prediction. Other techniques like GA, NN, ACO, and SVM can be used for optimization in large spaces.

Software reliability prediction is an important research area due to the challenges associated with it. This study reviews different computational intelligence techniques, considering various parameters, for reliability prediction in Component-Based Systems (CBS).

7 CONCLUSION

This paper focuses on factors affecting the reliability of CBS and importance of CBS using computational intelligence techniques for optimization. There are many computational techniques such as NN, GA, SVM, Cuckoo, Tabu search, etc but we focused on Fuzzy logic, ABC, ACO, and PSO. It is analyzed that optimization techniques PSO, ACO, ABC, and Fuzzy rationale have been utilized for anticipating CBSRe. These advancement methods utilized those components that influence the CBS framework. This paper presents the examination and evaluation of ACO, ABC, and PSO methods dependent on an audit of the writing. It is seen that PSO and Fuzzy rationale might be used where a reaction is quick and yield with fewer mistakes is required. ACO might be utilized where the shortest path's length is processed. In the future, further research on these techniques may yield a model for improving CBS reliability prediction using CBS specific parameters.

REFERENCES

[1] Vijayalakshmi K and Ramaraj N (2014) Modeling for component selection Assembly and quality assurance of Component based software. PhD Thesis, Anna University.

[2] Dubey, S.K., Jasra, B. Reliability assessment of component based software systems using fuzzy and ANFIS techniques. Int J Syst Assur Eng Manag 8, 1319–1326 (2017). https://doi.org/10.1007/s13198-017-0602-z.

[3] Diwaker, C., Tomar, P., Poonia, R.C. *et al.* Prediction of Software Reliability using Bio Inspired Soft Computing Techniques. J Med Syst 42, 93 (2018). https://doi.org/10.1007/s10916-018-0952-3

[4] Diwaker, C., *et al.* A New Model for Predicting Component-Based Software Reliability Using Soft Computing. *IEEE Access*,7:147191-147203 (2019). doi: 10.1109/ACCESS.2019.2946862.

[5] Wolski Marcin, Walter Bartosz, Kupiński Szymon and Chojnacki Jakub. Software quality model for a research-driven organization—An experience report. Journal of Software: Evolution and Proces 30(5), 1-14(2017). doi:10.1002/smr.1911. e1911.

[6] Singh Y., Kaur A., and Malhotra R. Predicting Testing Effort using Artificial Neural Network. Proceedings of the World Congress on Engineering and Computer Science (WCECS), 1-6, 2008, ISBN: 978-988-98671-0-2

[7] Khoshgoftaar T. M., Szabo R. M. and Guasti P. J. Exploring the behaviour of neural network software quality models, Software Engineering Journal, 10(3), 89-96 (1995). doi:10.1049/sej.1995.0012

[8] Sedigh-Ali S., and Ghafoor A. A Graph-Based Model for Component-Based Software Development. Proceedings of the 10th IEEE International Workshop on Object-Oriented Real Time Dependable Systems, IEEE, 1-6 (2005).

[9] Feurer M., Klein A., Eggensperger K., Tobias Springenberg J., Blum M., and Hutter F. Efficient and Robust Automated Machine Learning. Advances in Neural Information Processing Systems 28 (NIPS), 1-9 (2015).

[10] Mendoza H., Klein A., Feurer M., Tobias Springenberg J. and Hutter F. Towards Automatically-Tuned Neural Networks. JMLR: Workshop and Conference Proceedings, ICML 2016 AutoML Workshop, 58–65 (2016).

[11] Sagar S., Nerurkar N.W. and Sharma A. A soft computing based approach to estimate reusability of software components. ACM SIGSOFT Software Engineering Notes 35(4): 1-5 (2010). doi:10.1145/1811226.1811235

[12] Sangwan O. P., Bhatia P. K. and Singh Y. Software reusability assessment using soft computing techniques. ACM SIGSOFT Software Engineering Notes 36(1): 1-7(2011). doi:10.1145/1921532.1921548

[13] Diwaker C., and Tomar P. Assessment of Ant Colony using Component-Based Software Engineering Metrics. Indian Journal of Science and Technology 9(44):1–5(2016). DOI:10.17485/ijst/2016/v9i44/105159

[14] Diwaker C., and Tomar P. Optimization and appraisal of PSO for CBS using CBSE metrics. 3rd International Conference on Computing for Sustainable Global Development (INDIACom), IEEE, 1024–1028 (2016).

[15] Diwaker C., and Tomar P. Evaluation of swarm optimization techniques using CBSE reusability metrics. IJCTA 2(22): 189–197 (2016).

[16] Rana, S. and Yadav, R. K. A Fuzzy Improved Association Mining Approach to Estimate Software Quality International Journal of Computer Science and Mobile Computing 2(6) 116-122(2013).

[17] Dorigo M. (1992) Optimization, learning, and natural algorithms. Ph. D. Thesis, Politecnico di Milano, Italy

[18] Zheng T. (2019) Automatic Test Case Generation Method of Parallel Multi-population Self-adaptive Ant Colony Algorithm. In: Patnaik S., Jain V. (eds) Recent Developments in Intelligent Computing, Communication and Devices. Advances in Intelligent Systems and Computing, vol 752. Springer, Singapore

[19] Dahiya O., Solanki K., Dalal S., and Dhankhar A. An Exploratory Retrospective Assessment on the Usage of Bio-Inspired Computing Algorithms for Optimization. International Journal of Emerging Trends in Engineering Research 8(2): 414-434 (2020).

[20] Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes University, Computer Engineering Department, 200, 2005.

[21] Akay R., Akay B. (2020) Artificial Bee Colony Algorithm and an Application to Software Defect Prediction. In: Bennis F., Bhattacharjya R. (eds) Nature-Inspired Methods for Metaheuristics Optimization. Modeling and Optimization in Science and Technologies, vol 16. Springer, Cham

[22] Bai Q. Analysis of Particle Swarm Optimization Algorithm. Computer and Information Science 3(1): 180-184 (2010).

[23] Caldiera G., and Basili V. R. Identifying and Qualifying Reusable Software Components. Computer 24(2): 61-70 (1991). DOI: 10.1109/2.67210

[24] Caballero R.E., Demurjian S.A. (2002) Towards the Formalization of a Reusability Framework for Refactoring. In: Gacek C. (eds) Software Reuse: Methods, Techniques, and Tools. ICSR 2002. Lecture Notes in Computer Science, vol 2319. Springer, Berlin, Heidelberg

[25] Rumbaugh J., Blaha M., Premerlani W., Eddy F., and Lorensen W. E. Object-Oriented Modeling and Design, 199(1). Englewood Cliffs, NJ: Prentice-hall, 1991.

[26] International Organization for Standardization. ISO/IEC 9126-1: Software engineering - product quality - part 1: Quality model, 2001.

[27] ISO/IEC 25010:2011 Systems and Software Engineering -- Systems and Software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models, 2011.

