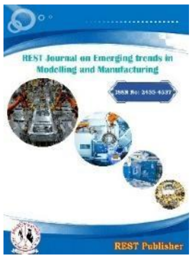
REST Journal on Emerging trends in Modelling and Manufacturing
Vol: 9(4), December 2023
REST Publisher; ISSN No: 2455-4537
Website: https://restpublisher.com/journals/jemm/
DOI: https://doi.org/10.46632/jemm/9/4/2



Evaluation of Programming in C using WSM Method

***Sathiyaraj Chinnasamy, M. Ramchandran, Vidhya Prasanth, Manjula Selvam**
REST Labs, Kaveripattinam, Krishnagiri, Tamil Nādu, India.
*Corresponding Author Email: sathiyarajrsri@gmail.com

Abstract: *Programming in C. The machine-oriented programming language C is mostly used to create many applications and operating systems, like Windows, as well as other complicated programmers, such the Oracle database, Kit, the Python interpreter, and games. Computer programmers and low-level programming applications are often written in the procedural or structured programming language C. Concatenation, data hiding, data compression, inheritance, and polymorphism are some extra aspects of C++ despite the fact that it is an object-oriented programming language. Multiple entities of the same type can be grouped together into a larger group using the C concept of an array. These entities or components may be user-defined data types or structures, such as integer, float, double, or float data types. C-written programmers compile and execute far more quickly than those written in other languages. This is because there are no additional processing overheads like garbage collection. As a result, when compared to other programming languages, the language is quick. An algorithm is a series of actions carried out in a preset order to address a challenge or finish a task. A function is a section of code that is called and carried out by other software elements. An operating system is made using it. The "C" programming language is used to create operating systems like Apple's OS X, Microsoft's Windows, and Symbian. It is utilized to create platforms for desktop and mobile devices. It is used to create compilers. One C feature that contains symbols for doing mathematical, relational, bitwise, conditional, or logical manipulations is the C operator. There are many built-in operators in the C programming language that can be used to carry out different tasks as needed by the application. C has an advantage over other dynamic languages since it is a statically typed programming language. Also, C is a compiler-based programming language, in contrast to Python and Java, which are interpreter-based. It expedites the compilation and execution of code. Weighted Sum works by multiplying the designated field values Indian Technical Institution or appraising the alternatives Analysis in Simply types, Enum types, Struct types, Nullable types. Evaluation parameters in Class types, Interface types, Array types, and Delegate types. Simply types, Enum types, Struct types, Nullable types. Class types, Interface types, Array types, and Delegate types. Nullable types got the first rank whereas Simply types have the lowest rank.*

Keywords: *Contribution, Related work, WSM Method.*

1. INTRODUCTION

There is an urgent need to offer educational coherence in computing given the technology's rapid development, impact on modern life, and requirement. A four-level model curriculum for CS for K–12 is suggested to achieve this goal. The introduction of the foundational ideas of computer science to all students, starting in elementary school, is one of the four objectives of this course. The goal of this course is to teach high school graduates how to use computational thinking (CS) concepts and skills to solve problems across a range of topics. As a result, it is recognised that learning to programme is a key element of this curriculum. Many technologies are promoted as the panacea for our educational system's problems. Motion pictures, television, media centres, and video recorders, according to its proponents, might significantly and permanently enhance how we educate our kids. However, the proponents of these technologies made more promises than they were able to keep. In this century and the ones to come, learning through computing and technology will advance. But as Solomon notes, there is frequently a disconnect between the opportunities presented by technology advancements and their real impact on education. Pointer analysis holds great promise for enhancing and linking compilers, despite recent advancements remaining in the research phase. Before it may be used as a tool, a number of problems need to be fixed. The analysis must first be effective without compromising the accuracy of the findings. Second, genuine C programmes must be handled by pointer analysis methods. It won't be extensively used if the analysis only yields accurate findings for properly working input programmes. These issues are resolved by a pointer analysis algorithm that we have created. Finding the potential values of pointers at each statement in a programme is the aim of our analysis. We use point functions to represent that data. We briefly review nested data structures as well as global and layer variables, but we do not make an effort to examine the connections between specific components of recursive data structures. It is generally accepted that efficient model checking of software systems can result in significant increases in software resilience and dependability. The state-space explosion problem, however, greatly restricts the performance of model validation of such systems, and the majority of research in this field focuses on shrinking the state-space of the model that is used for validation. The state space of software systems can be reduced in a number of ways, one of which is abstraction. By translating a set of states from a real system to small, compact, system-preserving relative behaviour, abstraction techniques shrink the programme state space. Abstracts are frequently hand-written, informal, and highly specialised.

2. MATERIAL AND METHODS

Contribution: This study suggests using a SAT solver to create a compression algorithm. In a SAT event, calls that demonstrate the highest possible number are substituted by the number. Our method involves first creating a symbolic representation of the concrete transition connection for each fundamental block in a given project using symbolic simulation techniques. The relationship between the variables is then given predicates in the form of the current and subsequent states, creating a Boolean formula. Finally, we compute the values of the predictors numerically using the SAT solver. We employ the same technique we've already developed to build a new compression whenever the compression programme has to be improved. This method has the advantage of eliminating the exponential number of calls for theorem proving; instead, the SAT solution searches for potential assignments to the values of the predictions. The SAT solvers of today are more effective and support more variables. As a result, many more potential assignments can be verified, increasing the accuracy of the summary transformation and removing needless erroneous counterexamples. The majority of ANSI-C constructs may be encoded using CNF, making our method advantageous for a variety of projects. Bit vector operators are used to encrypt integer operators, which allows them to care for potential arithmetic overflow. Due to the incorrect assumption that the variables' possible values are unlimited, there are no false positive results. Additionally, it's possible to enable constructs for pointer manipulation, such as pointer arithmetic. Recursion and dynamic memory allocation are the only restrictions that apply. The boolean programme must be finite in order to overcome this restriction.

Related work: The usage of data compression techniques is widespread, and many scholars have studied them. The Cousot and Cousot's abstract description work serves as the foundation for many abstraction techniques, which call for the user to supply an abstract function that corresponds to concrete data types for abstract data types. The user must specify a collection of predicates that affect the verification property and establish a general-purpose theorem in order to apply the Predicate Abstraction kind of abstract description technique in earlier applications. These techniques are less effective for large applications due to user-driven identification of pertinent predictors. A set of predictors for compression has recently been computed using a variety of decision-making processes. The use of error traces to direct predictive discovery is a more typical strategy. The program's BDD representations serve as the foundation for the algorithm. This is a disadvantage for large applications since transition relation BDDs are frequently too big to handle well. The transparent state representation used by the algorithm in this work is restricted to security features. A finite localization reduction technique "frees" programme variables that don't affect the verification properties, resulting in an initial compaction of the programme. Deterministic definitions of "free" variable values lead to very approximate programme behaviours. The software is made to stop acting unrealistically by progressively returning the "free" variables to their initial values.

3. WSM Method

A selection theory Weighted sum sampling method WSM is very the well-known MCDM (multi-criteria decision-making) is one of the techniques and primarily some Alternatives based on criteria Easier to evaluate is one. WSM is valid handiest while all information supplied is in the same size or unit. The in each column Rows are compressed, using their respective rank sums Columns are sorted If the rank sum is reduced the column molecule is searched the same as the reference form will be others mixtures of rating matrix except summation have been studied. This approach is relevant to tuning parameter choice and different regions in which Subgroup variables of variables must be selected from the set This is when the SRD method is monitored The approach can be considered unsupervised (A goal vector is used) In addition to the SRD approach Can be used in molecular fitting research. Factor weights for robot selection are A weighted sum model This model has no institutional consensus on those values. In choosing robots, the best weights and subjectivity less expert on components Values are removed. The main purpose for getting rid of These values is any capacity at the last stage It is to reduce the impact of distorted desire to explain the version and program A numerical example is presented as the ranking change in comparison to a version that does not do away with those excessive values. Using weighted-sum beam forming, the microphone arrangement, which includes the variety and function of the microphones, determines the weight of every microphone signal. To determine the design parameters, diverse simulations had been finished if the listener had a head. To make amends for the and the impact is accounted for using the round head-related transfer function (HRTF). We perform simulations concerning a roundhead version. The Weighted Sum Model (GWSM) accounts for multi-year uncertainties with the aid of comparing the enterprise environment in West Africa. The deal with a first-rate problem is now not blanketed through DBP, specifically, ranking countries throughout the years by considering inside-country uncertainty and investor possibilities as criterion weights. Second, we enlarge the traditional weighted sum model. Of weights containing pure gas the sum equals a common way to use calculate the entire emissions using making a grey approximation to resolve the spectrally included RTE. An alternative method non-gray or bar formula. To decide the depth of penetration, the sum rules need to be cautiously applied. Our effects display that Normal and superconducting move the c-axis between positions A within energy There is trade, for a speed-dependent gap; This exchange in kinetic energy ought to be taken under consideration to properly derive the penetration intensity from conductance sum regulation Naïve use of conductivity sum. Important (1) part Determination of sum rule closely related the greater trendy trouble of improving the feature Out of test range is widely recognized the evaluation (holomorphic) of a complicated feature σ(ω) on a given area D can persevere analytically over the complete domain inclusive of the last boundary from a subset of the boundary of this area. The weight trouble must be solved first. Furthermore, modeling the dynamic shape factor studied with the aid of MNS is extra tricky considering that discrete Sum laws of theoretical models are satisfying. Any theory Notification of serious settlement dynamic structure issue measured in absolute devices should explain how the regulation of composition is happy or why it is violated. All like the weight of white fuel a0 the sum of the weights zero = zero; Therefore, εt, calculated by the SNB version, is the sum of the differences among L and by the WSGG version of SQP Extraordinary path with help Calculated for length set of rules. Weighted sum rules for exchange forces A very sensitive test Fourier components optimization measures, roughly speaking, it proved. Transfer potential of the two-particle interaction density. Sum (SNNMS) reduces the number of LDPC decoding network Correction factors. A single revision in a single layer by dividing the factors Through the SNNMS LDPC decoding network good performance can be achieved with a small increase in computational complexity. The weighted sum model does not require any supported solutions to be pruned with this optional correlation. To the best of our understanding, the priority relation is only implemented to given answers and non-stop multi-objective optimization troubles.

4. RESULT AND DISCUSSIONS

TABLE 1. Programming in C

	Class Types	Interface Types	Array Types	Delegate Types
Simple Types	96	92.53	38.15	45.05
Enum Types	87.12	74.97	43.69	27.3
Struct Types	94.08	89.58	29.18	33.1
Nullable Types	83.17	68.28	14.6	27.59

Table 1 shows Programming in C using the analysis method in WSM. Simply types, Enum types, Struct types, Nullable types Alternative and Class types, Interface types, Array types, and Delegate types Evaluation is also a data set in the value.

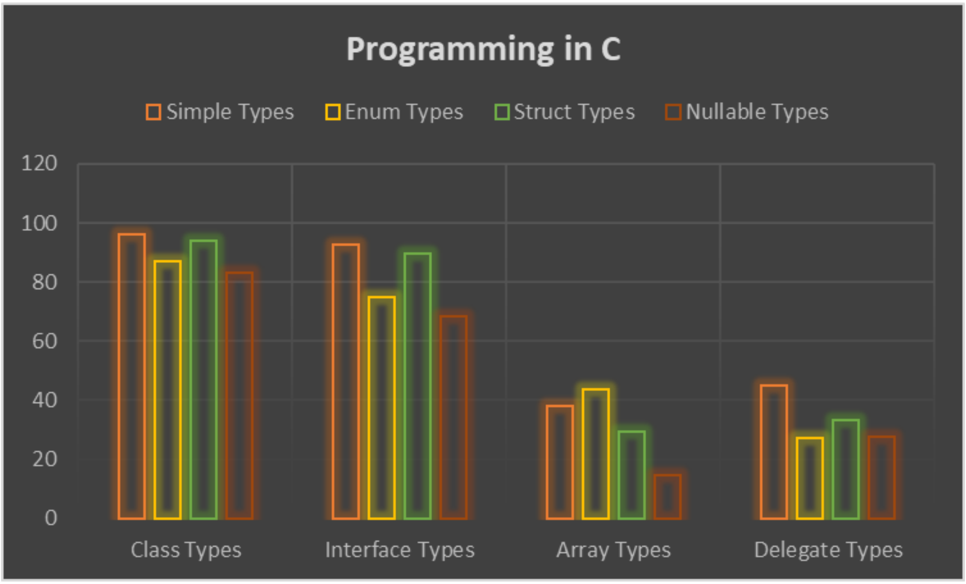


FIGURE 1. Programming in C

Figure 1 shows Programming in C using the analysis method in WSM. Simply types, Enum types, Struct types, Nullable types Alternative and Class types, Interface types, Array types, and Delegate types Evaluation is also a data set in the value.

TABLE 2. Normalized Data

Normalized Data			
1	1	0.3827	0.605993
0.9075	0.810224	0.334173	1
0.98	0.968118	0.500343	0.824773
0.866354	0.737923	1	0.989489

Table 2 shows the Normalized data for Programming in C are the Simply types, Enum types, Struct types, Nullable types and Class types, Interface types, Array types, and Delegate types it is also the Maximum in Normalized value.

FIGURE 2. Normalized Data

Figure 2 shows the Normalized data for Programming in C are the Simply types, Enum types, Struct types, Nullable types and Class types, Interface types, Array types, and Delegate types it is also the Maximum in Normalized value.

TABLE 3. Weight

Weight			
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25

Table 3 shows the Weight ages used for the analysis. We have taken the same weights for all the parameters for the analysis.

TABLE 4. Weighted Normalized Decision Matrix

Weighted normalized decision matrix			
0.25	0.25	0.095675	0.151498
0.226875	0.202556	0.083543	0.25
0.245	0.24203	0.125086	0.206193
0.216589	0.184481	0.25	0.247372

Table 4 shows the Weighted Normalized Decision Matrix for Simply types, Enum types, Struct types, Nullable types and Class types, Interface types, Array types, and Delegate types also Multiple values.

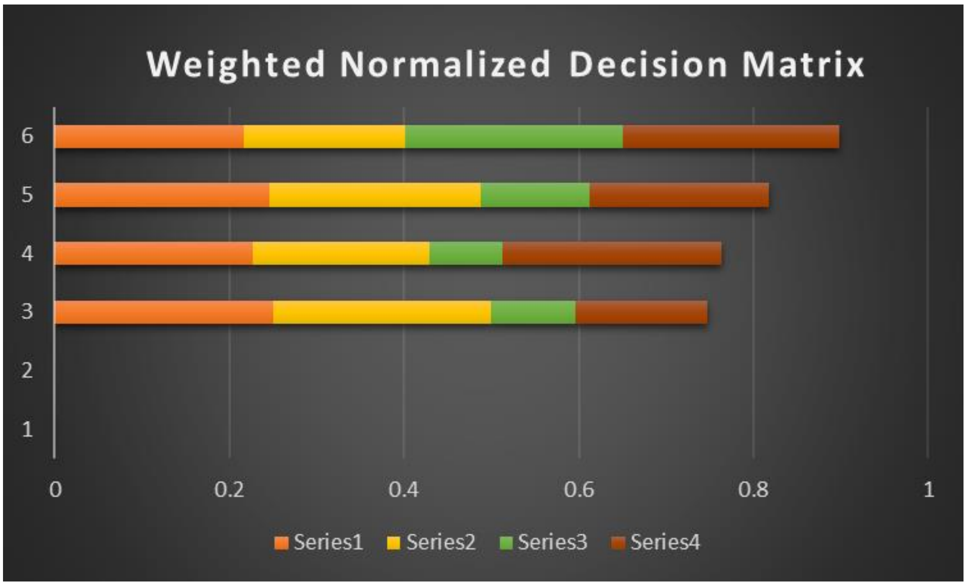


FIGURE 3. Weighted Normalized Decision Matrix

Figure 3 shows the Weighted Normalized Decision Matrix for Simply types, Enum types, Struct types, Nullable types and Class types, Interface types, Array types, and Delegate types also Multiple values.

TABLE 5. Final Result of Programming in C

	Preference Score	Rank
Simple Types	0.747173	4
Enum Types	0.762974	3
Struct Types	0.818309	2
Nullable Types	0.898441	1

Table 5 shows the final result of WSM for Programming in C. Preference Score is calculated using the Nullable Types having is Higher Value and Simple Types having a lower value.

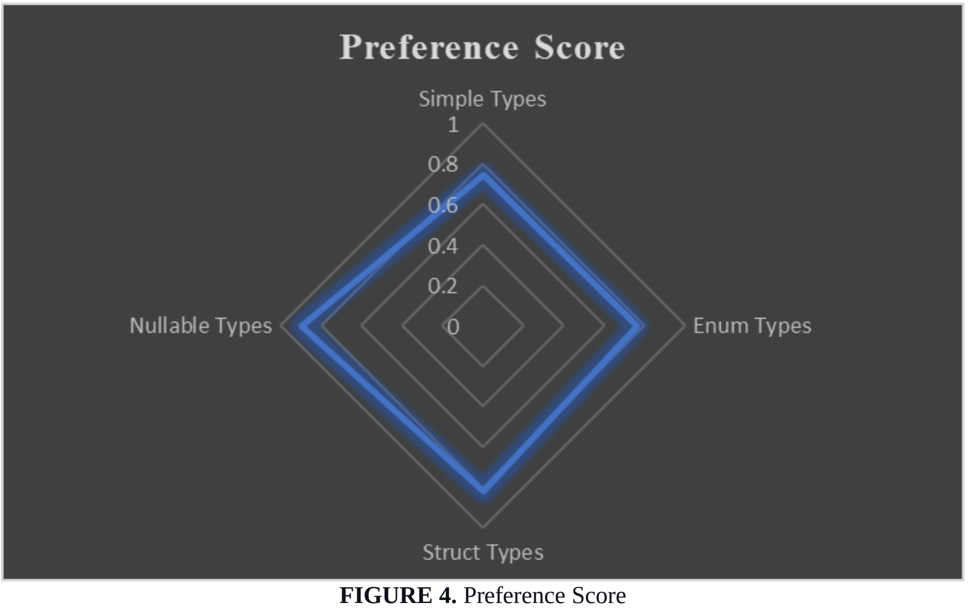


FIGURE 4. Preference Score

Figure 4 shows the final result of WSM for Programming in C. Preference Score is calculated using the Nullable Types having is Higher Value and Simple Types having a lower value.

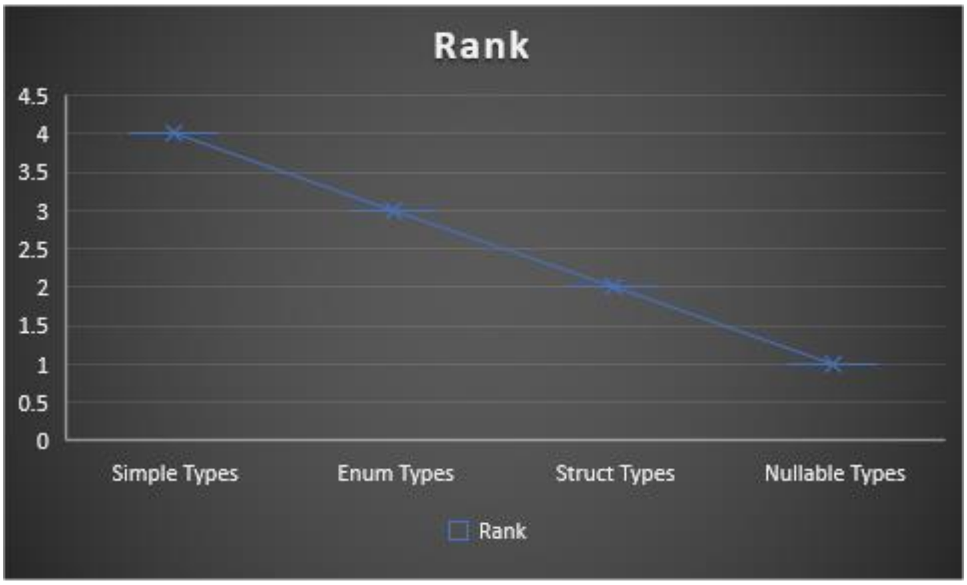


FIGURE 5. Show the Rank

Figure 5 shows the Ranking of Programming in C. Nullable types is got the first rank whereas is the Simple types is having the lowest rank.

5. CONCLUSIONS

The LECGO web-based learning environment, which highlights the effectiveness of designing tasks for fundamental learning in programming using C, is the subject of this paper's discussion of the design and pilot evaluation phases. The suggested environment incorporates multimedia, multiple representations, and multi-layered, activity-based hyperlinked material. LECGO was created using a modelling strategy and was motivated by contemporary constructivist and social learning theories. Beginners' comprehension seemed to be a difficult task, especially when they utilised paper and pencil and Turbo C, according to data analysis from the LECGO field-piloted comparison evaluation research. Contexts get little to no assistance. The data analysis showed drastically different findings when comparing the outcomes of the suggested learning environment: more students succeeded in LECGO than in the paper-and-pencil and Turbo C settings. This is due to the fact that LECGO typically provided features that were distinct from those of the typical paper-and-pencil environment and Turbo C's generic compiler. Students are actually compelled to use programming commands in p-p and Turbo C. written directly in C. Instead, with LECGO, students are given the chance to convey a variety of sorts of information, including intuitive knowledge expressed through the creation of drawings for problems, knowledge expressed through the use of natural language to describe answers, and knowledge expressed through shape. giving the computer instructions in both imperative and fake code to carry out the current tasks. We have demonstrated the high efficiency of a completely context-sensitive pointer analysis technique for a set of C programmes. The basic premise of this technique is that aliases in procedure inputs are often the same across all calling contexts. We can identify partial transfer functions for input aliases in the programme, despite the fact that it is challenging to summarise a process' behaviour for all inputs. It enables a process to be examined once, with the findings being applied to other scenarios. The predictive compactness of an ANSI-C programme is calculated using a new method that is presented in this study. The use of the SAT solver in this novel approach eliminates the need for theorem provers. Because computing a single SAT instance substitutes an exponential number of theorem prover calls, we propose that SAT-based predictive abstraction performs better than methods that employ theorem provers. When there are much fewer abstract transitions than possibilities to be verified, the benefits become very clear. Additionally, employing a SAT engine produces a more accurate exchange relation of the abstract programme compared to an abstraction produced using theorem provers because modern SAT solvers allow the evaluation of a large number of potential assignments to the abstract programme variables. The summation programme no longer exhibits some of the unrealistic behaviour that would otherwise be introduced during over-approximations of the summation transition relation computed using a theorem prover.

REFERENCE

[1]. Kordaki, Maria. "A drawing and multi-representational computer environment for beginners' learning of programming using C: Design and pilot formative evaluation." *Computers & Education* 54, no. 1 (2010): 69-87.

[2]. Mandala, Vishwanadham, Srinivas Naveen Reddy Dolu Surabhi, Phani Durga Nanda Kishore Kommisetty, Bala Maruthi Subba Rao Kuppala, and Roopak Ingole. "Towards Carbon-Free Automotive Futures: Leveraging AI And ML For Sustainable Transformation." *Educational Administration: Theory and Practice* 30, no. 5 (2024): 3485-3497.

[3]. Shanmugasundar, G., R. Sivaramakrishnan, and M. Rajmohan. "Computer aided simulation for workspace plot of a newly designed inspection robot." In *2014 IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1-6. IEEE, 2014.

[4]. Spoorthi. S,; Harshith. T. N,; M. Ramachandran; Chandrasekar Raja, " A Review on Child Safety Monitoring System Based on IOT", Recent trends in Management and Commerce 4(2), 2023: 130-135.

[5]. Sunitha, R. "A study on competency mapping scale to map the competencies of university teachers (with special reference to karnataka state)." *South Asian Journal of Engineering and Technology* 11, no. 1 (2021): 1-3.

[6]. Ponnada, Venkata Tulasiramu, and SV Naga Srinivasu. "Efficient CNN for lung cancer detection." *Int J Recent Technol Eng* 8, no. 2 (2019): 3499-505.

[7]. Schwartz, Scott, Laura Elnitski, Mei Li, Matt Weirauch, Cathy Riemer, Arian Smit, NISC Comparative Sequencing Program, Eric D. Green, Ross C. Hardison, and Webb Miller. "MultiPipMaker and supporting tools: Alignments and analysis of multiple genomic DNA sequences." *Nucleic acids research* 31, no. 13 (2003): 3518-3524.

[8]. Lewis, GeorgeP, WilliamJ Jusko, ChristopherW Burke, Linda Graves, and Boston Collaborative Drug Surveillance Program. "Prednisone side-effects and serum-protein levels: A collaborative study." *The Lancet* 298, no. 7728 (1971): 778-781.

[9]. Ponnada, Venkata Tulasiramu, and SV Naga Srinivasu. "Integrated clinician decision supporting system for pneumonia and lung cancer detection." *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* (2019).

[10]. Sunitha, R. "Work life balance of women employees of teaching faculties in karnataka state." *Journal of Management and Science* 10, no. 4 (2020): 40-42.

[11]. T. Santhosh; Harshitha. T. N,; Sathiyaraj Chinnasamy; M. Ramachandran, "Adaptive Subgradient Methods for Leadership And Development", Recent trends in Management and Commerce 4(2) 2023, 101-106.

[12]. Mandala, Vishwanadham, Srinivas Naveen Dolu Surabhi, V. R. Balaji, Dipak Raghunath Patil, and Saiyed Faiayaz. "Wild Horse Optimizer and Support Vector Machine (SVM) Classifier Predicts the Heart Disease Converging Nature-Motivated Optimization and Machine Learning."

[13]. Necula, George C., Scott McPeak, Shree Prakash Rahul, and Westley Weimer. "CIL: Intermediate language and tools for analysis and transformation of C programs." *CC* 2 (2002): 213-228.

[14]. Shanmugasundar, G., Gaurav Sapkota, Robert Čep, and Kanak Kalita. "Application of MEREC in multi-criteria selection of optimal spray-painting robot." *Processes* 10, no. 6 (2022): 1172.

[15]. Clarke, Edmund, Daniel Kroening, and Flavio Lerda. "A tool for checking ANSI-C programs." In *Tools and Algorithms for the Construction and Analysis of Systems: 10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29-April 2, 2004. Proceedings 10*, pp. 168-176. Springer Berlin Heidelberg, 2004.

[16]. U. Midhundev; Harshith. T. N,; M. Ramachandran; Kurinjimalar Ramu, "An Empirical Investigation of Innovation and Technology in Banking" , Recent trends in Management and Commerce 4(2), 2023: 121-129.

[17]. Sunitha, R., and J. K. Raju. "RISK MANAGEMENT IN BANKING SECTOR--AN DESCRIPTIVE STUDY." (2013).

[18]. Surabhi, Srinivas Naveen D., Chirag Shah, Vishwanadh Mandala, and Priyank Shah. "Range Prediction based on Battery Degradation and Vehicle Mileage for Battery Electric Vehicles." *International Journal of Science and Research* 13 (2024): 952-958.

[19]. Ball, Thomas, Rupak Majumdar, Todd Millstein, and Sriram K. Rajamani. "Automatic predicate abstraction of C programs." In *Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation*, pp. 203-213. 2001.

[20]. Ponnada, Venkata Tulasiramu, and SV Naga Srinivasu. "End to End System for Pneumonia and Lung Cancer Detection using Deep Learning." *Int. J. Eng. Adv. Technol* 8 (2019).

[21]. Surabhi, Srinivas Naveen D., Chirag Vinalbhai Shah, Vishwanadh Mandala, and Priyank Shah. "Advancing Faux Image Detection: A Hybrid Approach Combining Deep Learning and Data Mining Techniques." *International Journal of Science and Research (IJSR)* 13 (2024): 959-963.

[22]. Xiao, Rui, Bi Zhang, Yongming Dong, Jianke Gong, Tao Xu, Jianfeng Liu, and XZ Shawn Xu. "A genetic program promotes C. elegans longevity at cold temperatures via a thermosensitive TRP channel." *Cell* 152, no. 4 (2013): 806-817.

[23]. Prechelt, Lutz. "An empirical comparison of seven programming languages." *Computer* 33, no. 10 (2000): 23-29.

[24]. Stoet, Gijsbert. "PsyToolkit: A software package for programming psychological experiments using Linux." *Behavior research methods* 42 (2010): 1096-1104.

[25]. Shanmugasundar, G., Tapan K. Mahanta, Robert Čep, and Kanak Kalita. "Novel fuzzy measurement alternatives and ranking according to the compromise solution-based green machining optimization." *Processes* 10, no. 12 (2022): 2645.

[26]. E. Chambers, Charles, Kenneth A. Fetterly, Ralf Holzer, Pei-Jan Paul Lin, James C. Blankenship, Stephen Balter, and Warren K. Laskey. "Radiation safety program for the cardiac catheterization laboratory." *Catheterization and Cardiovascular Interventions* 77, no. 4 (2011): 546-556.

[27]. Ponnada, Venkata Tulasiramu, and SV Naga Srinivasu. "Edge AI system for pneumonia and lung cancer detection." Int J Innov Technol Exploring Eng 8, no. 9 (2019).

[28]. Chang, Pohua P., and W-W. Hwu. "Inline function expansion for compiling C programs." In *Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation*, pp. 246-257. 1989.

[29]. Palumbo, David B. "Programming language/problem-solving research: A review of relevant issues." *Review of educational research* 60, no. 1 (1990): 65-89.

[30]. Shanmugasundar, G., R. Sivaramakrishnan, R. Sridhar, and M. Rajmohan. "Computer aided modelling and static analysis of an inspection robot." *Applied Mechanics and Materials* 766 (2015): 1055-1060.
[31]. Wilson, Robert P., and Monica S. Lam. "Efficient context-sensitive pointer analysis for C programs." *ACM Sigplan Notices* 30, no. 6 (1995): 1-12.
[32]. Le Goues, Claire, Neal Holtschulte, Edward K. Smith, Yuriy Brun, Premkumar Devanbu, Stephanie Forrest, and Westley Weimer. "The ManyBugs and IntroClass benchmarks for automated repair of C programs." *IEEE Transactions on Software Engineering* 41, no. 12 (2015): 1236-1256.
[33]. Mandala, Vishwanadham. "Optimizing Fleet Performance: A Deep Learning Approach on AWS IoT and Kafka Streams for Predictive Maintenance of Heavy-Duty Engines." *International Journal of Science and Research (IJSR)* 8, no. 10 (2019): 1860-1864.
[34]. Mandala, Vishwanadham. "Predictive Failure Analytics in Critical Automotive Applications: Enhancing Reliability and Safety through Advanced AI Techniques." *Journal of Artificial Intelligence and Big Data* (2024): 48-60.
[35]. Shanmugasundar, G., Tapan K. Mahanta, Robert Čep, and Kanak Kalita. "Novel fuzzy measurement alternatives and ranking according to the compromise solution-based green machining optimization." *Processes* 10, no. 12 (2022): 2645.
[36]. Thinakaran, Georgia L. *Sustainable Asset Creation Under Mgnrega*. Sankalp Publication.
[37]. Griewank, Andreas, David Juedes, and Jean Utke. "Algorithm 755: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++." *ACM Transactions on Mathematical Software (TOMS)* 22, no. 2 (1996): 131-167.
[38]. Clarke, Edmund, Daniel Kroening, Natasha Sharygina, and Karen Yorav. "Predicate abstraction of ANSI-C programs using SAT." *Formal Methods in System Design* 25 (2004): 105-127.
[39]. Ishizue, Ryosuke, Kazunori Sakamoto, Hironori Washizaki, and Yoshiaki Fukazawa. "PVC. js: Visualizing C programs on web browsers for novices." *Heliyon* 6, no. 4 (2020).
[40]. Mandala, Vishwanadham. "From Reactive to Proactive: Employing AI and ML in Automotive Brakes and Parking Systems to Enhance Road Safety." *International Journal of Science and Research (IJSR)* 7, no. 11 (2018): 1992-1996.
[41]. Chang, Pohua P., Scott A. Mahlke, William Y. Chen, and Wen-Mei W. Hwu. "Profile-guided automatic inline expansion for C programs." *Software: Practice and Experience* 22, no. 5 (1992): 349-369.
[42]. Sakawa, Masatoshi. "Interactive computer programs for fuzzy linear programming with multiple objectives." *International Journal of Man-Machine Studies* 18, no. 5 (1983): 489-503.
[43]. Shanmugasundar, G., R. Sivaramakrishnan, R. Sridhar, and M. Rajmohan. "Computer aided modelling and static analysis of an inspection robot." *Applied Mechanics and Materials* 766 (2015): 1055-1060.
[44]. Cheng, Harry H. "Scientific Computing in the C^ H Programming Language." *Scientific Programming* 2, no. 3 (1993): 49-75.
[45]. Thinakaran, Georgia L. "A Study On The Problems Faced By The Trans Entrepreneurs With Special Reference To Selected Districts In Tamilnadu-An Explorative Analysis." Think India Journal 22, no. 10 (2019): 6584-6590.
[46]. Ousterhout, John K. "Scripting: Higher level programming for the 21st century." *Computer* 31, no. 3 (1998): 23-30.
[47]. Carbonneaux, Quentin, Jan Hoffmann, Tahina Ramananandro, and Zhong Shao. "End-to-end verification of stack-space bounds for C programs." In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 270-281. 2014.
[48]. Shanmugasundar, G., R. Sivaramakrishnan, R. Sridhar, and M. Rajmohan. "Computer aided modelling and static analysis of an inspection robot." *Applied Mechanics and Materials* 766 (2015): 1055-1060.
[49]. Ball, Thomas, Andreas Podelski, and Sriram K. Rajamani. "Boolean and Cartesian abstraction for model checking C programs." In *Tools and Algorithms for the Construction and Analysis of Systems: 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2–6, 2001 Proceedings 7*, pp. 268-283. Springer Berlin Heidelberg, 2001.
[50]. Thies, William, Vikram Chandrasekhar, and Saman Amarasinghe. "A practical approach to exploiting coarse-grained pipeline parallelism in C programs." In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*, pp. 356-369. IEEE, 2007.
[51]. Mandala, Vishwanadham. "Integrating AWS IoT and Kafka for Real-Time Engine Failure Prediction in Commercial Vehicles Using Machine Learning Techniques." *International Journal of Science and Research (IJSR)* 8, no. 12 (2019): 2046-2050.
[52]. Shanmugasundar, G., G. Fenneth Moses, S. Jayachandran, VD Rathnavel Subramanian, and R. Rajagopalan. "Design and fabrication of solar powered multi-purpose agricultural vehicle with iot control." *Journal of Advanced Research in Dynamical and Control Systems* 12, no. 7 (2020).
[53]. Ponnada, Venkata Tulasiramu, Venkata Tulasikrishna Ponnada, Rama Krishna Raju Sammeta, and Hymavathi Jasti. "Making Healthcare Decisions: An Evolution." In *Intelligent Decision Making Through Bio-Inspired Optimization*, pp. 85-110. IGI Global, 2024.
[54]. Venkatesh, G. A. "Experimental results f rom dynamic slicing of C programs." *ACM Transactions on Programming Languages and Systems (TOPLAS)* 17, no. 2 (1995): 197-216.

