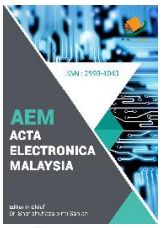
ZIBELINE INTERNATIONAL PUBLISHING

ISSN: 2590-4043 (Online)
CODEN: AEMCDV

Acta Electronica Malaysia (AEM)

DOI: http://doi.org/10.26480/aem.02.2025.53.60



RESEARCH ARTICLE

ALGEBRAIC EXPRESSION USING AN UPGRADED TAC OPTIMIZATION ALGORITHM

Yang Kai, Xu Tao

University School of Automation and Electrical Engineering, Lanzhou Jiaotong University, Lanzhou, Gansu, China.
Department of Electronic Science and Technology, Yancheng Institute of Technology, Yancheng, Jiangsu, China.
**Corresponding Author Email: danielyoung7557@gmail.com*

ARTICLE DETAILS

Article History:

Received 20 August 2025
Revised 26 September 2025
Accepted 27 November 2025
Available online 31 December 2025

ABSTRACT

Algebraic expressions are used in many application areas. In this paper, we propose a novel code optimization technique for algebraic expression. We used two techniques to reduce cost with respect to time. Experimental results are obtained by comparing optimization ratio of existing optimization approach with the proposed approach. Therefore, we used simple substitution method to solve algebraic problem and standard optimization technique and calculate the results that how both techniques reduce the complexity of Algebraic expression. The results show that cost of Algebraic expression using substitution and cost of optimize algebraic expression without substitution results is less than the original optimize Algebraic expression.

KEYWORDS

TAC Optimization, Algebraic expression, Cost, Time, Substitution, Technique.

1. INTRODUCTION

In broad-spectrum, a computer program optimized so that it execute fast and capable of operating less memory storage and take less use of other resources. Optimization in a program occurs at number of levels; program may be related to any algebraic expression or may be related to any other domain. In some particular scenarios, optimization depends on using well-designed algorithms, makes use of special cases and performs complex tasks. A fully optimized program is more difficult to find out and contain more responsibilities than un-optimized program. Most of the time, the optimization performed automatically by an optimized compiler, the problem arise while selecting a method that is more computationally efficient, when we are performing the same functionality recursively.

Mostly, compilers convert any algebraic expressions into the three address code in a direct way without following any strategy. For example; Take two arithmetic expressions and should write like this; Expression of right sideExpression of left side And store the resultant value in a temporary variables “temp” like this

Temp=Exp1 + Exp2

Modification provided by an optimizing compiler should have several benefits:

The entire meaning of program must remain same during modification. This means that optimization not change input or output string during modification. In modification, the modify piece of code must speed up that area of code for a certain limit. More often we want to reduce the size of compiled code but in fact the size of code does not matter in real. But somehow the process of modification slow down the process of program and in long terms, its task is to improve and remove the target area. The worth of effort lies in modified area of program. This is not we scène that designer of the compiler have to pay the logical effort to implement a code while improving the modify part of program and have also pay the additional time to compile the input string in the program. When we do not make this effort during execution of target area, the modification does not pay the worth of effort. An algebra expression is combination of operand and operator. Evaluating algebraic expressions is a simple process, but needs to follow set of rules for operations to get the desire results; the ordering details of operations are given according to expressions. Algebra is simple language but learning algebra is like learning another language. Although, algebraic problem is a simple to solve but it is sometime used to create mathematical models of real-world conditions and to hold the problems that we can't solve just by using calculation and another optimization technique.

Purpose work basically an algebraic expression is mathematical expressions that contain regular numbers. The basic purpose of writing this paper is to introduce new technique for solving algebraic expression through optimization technique. The result of our propose technique used less resources and minimize cost with respect to time. The propose technique convert the algebraic expression into three address code by divided into tokens and evaluate the token through applying compiler optimization technique one by one into expressions. The propose technique take less time to reduce the cost rather than original optimized technique.

When developing a control system, it's common to start with a plant that faces external obstacles. A collaborative design's goal is to limit the effect of these problems to a manageable level. As an example. State feedback control is employed in the "Disturbance Decoupling Problem" to ensure that the disturbances are entirely decoupled from the output. However, for some systems, minimizing the impact of a breakdown below a particular threshold may be difficult. As a result, there is no way to solve this problem. This is a note. We provide a method for developing a stable state feedback control that reduces the influence of obstacles to a predetermined level. The algebraic technique to rewriting magic sets is based on the multi-set relative algebra equation rules, including the semicolon operator.

The algebraic method specifies the search space explicitly and may be implemented in a rule-based optimizer (possibly in conjunction with heuristics to limit the research space). We illustrate how equation rules

Quick Response Code	Access this article online	
	Website: www.actaelectronicamalaysia.com	**DOI:** 10.26480/aem.02.2025.53.60

Cite the Article: Yang Kai, Xu Tao (2025). Algebraic Expression Using An Upgraded Tac Optimization Algorithm. *Acta Electronica Malaysia (AEM)*, 6(2): 53-60.

decide how to rewrite model magic for non-recurring MySQL queries by presenting a representative collection of equation rules. In recent years, there has been a growing need for high-performance and low-power solutions for devices like cellular phones, network routers, and video cameras in the embedded systems industry. As a result of this need, many high-level optimization approaches for both software and hardware have been researched and developed. In the design world, there are a number of enhancements that are popular. CSE, aggregate scalar replacement, value numbering, static and copy propaganda, and static and copy propaganda apply locally to basic blocks and globally to Control Flow Graph (CFG). Regrettably, the majority of the effort done in these compiler tools has been for general purposes. This method is ineffective for improving certain functions, such as multi-faceted expressions, which is the subject of this article. Polynomial expressions may be used in a wide range of application fields since they can be used to estimate any continuous function to the required degree of precision. A variety of scientific uses exist. On physical phenomenon statistics, use various frequencies. Multiple frequencies are frequently used in real-time computer graphics applications for surface and curve creation, structural mapping, and other tasks.

As long as the inaccuracy cannot be tolerated, most DSP algorithms employ trigonometric functions such as sine and cosine that can be approximated from their Taylor series. Furthermore, to simulate nonlinearities in high-speed communication networks, many synchronized signal processing applications need interdisciplinary usage. To implement such tasks, system designers frequently have to rely on hand-picked library routines, which may be time-consuming and error-prone. Because multidimensional feedback is costly to calculate in an embedded system with limited resources, it must be effectively tuned.

In order to minimize the amount of words in a collection of Boolean perceptions, algebra approaches have been effectively utilized in the synthesis of multi-level logic. Typically, these approaches are used on a set of Boolean feedback with hundreds of variables. In this article, we explain how to minimize the number of operations in a set of polynomial expressions using the same method. These algebraic methods are used in our algorithms to improve multidimensional feedback. The development of code for math impressions has taken a substantial amount of time and work. When just a small number of registers are available, the authors provide strategies for minimizing the number of program steps and storage references in the assessment of mathematical impressions. The article was later modified to include comments with commonly used sub-expressions. Although code generation approaches were used to fix the jobs described above, there was no methodology that effectively decreased the number of tasks in a collection of mathematical impressions.

The most difficult aspect of detecting logical changes is equality. This is simple in relative changes, when a change creates a logical expression with a consistent output connection. The outcomes of modifications involving numeric operators often do not produce exactly the same results, despite the fact that there is a change, due to the impacts of numerical precision and stability. The scientific user believes it is right. This difficulty can influence which changes can be applied to a given change, and in many situations, as explained in the subsections below, it can modify the expression's result. Do not alter your behavior. The necessity for database assistance for scientific applications has been discussed by a number of scholars.

Previous research has mostly focused on defining the specific requirements of scientific databases. Shushani, Olken, and Wong have compiled a list of the many forms of data that are utilized in scientific systems. Using a time series Experimental data is the most fundamental form of data that must be stored in a scientific database. The research did not address the operators who modified the data, while following work by the same authors particularly addresses database chores. Sampling, close neighbor search, estimation and interpolation, transposition, aggregation, and relational operations are all identified as sampling, close neighbor search, estimation and interpolation, transposition, aggregation, and relational operations.

The necessity to accommodate more complicated data types, particularly temporal data and time series, has been explored, as has the optimization of scientific database operators' queries. Attempts to locate the database system have recently been made. Two current survey papers must be given to fulfill the demands of the scientific community. The stated requirements include efficient parallel and distributed processing, database expansion, large-data management, specific data structures and storage formats, and analytics operations integration into the database management system.

For scientific databases, several academics have looked at data modeling and storage architectures. Kim has looked into how object-oriented database technology, with its support for complex data types, might be used to scientific storage concerns. While an object-oriented system may handle many of the demands of scientific and statistical systems, such as multi-dimensional bulk types, Kim essentially concludes that many significant scientific database difficulties are unrelated to data model issues.

Through the factorization of expressions using mathematical impressions, some effort was done to enhance the code. This article gives a broad overview of mathematical representations and techniques for discovering common sub-concepts. The disadvantage of this algorithm is that it relies solely on relationships to function. Mathematical operators can increase the commutative characteristics of a single kind of associative and/or commutative operator, resulting in a more commutative statement. As a result, it is unable to factorize expressions or generate complicated general sub-effects such as increments and multiplications. Math impressions must be tallied during address computation for data transfer intelligence (DTI) applications. Corrective adjustments at several vocabulary levels, such as expression distribution and grouping, induction variable analysis, and general sub-expression and factoring, are attempted to synthesize these perceptions. The algebraic approaches presented in this article may be used to factor and eliminate generic multidimensional impressions, as well as one of the modifications in address generator synthesis.

To minimize the number of literals in a collection of Boolean perceptions, algebraic approaches have been effectively used to the issue of synthesis of multi-level logic. This method is often used to create a collection of Boolean impressions with thousands of literals and hundreds of variables. Melting and factorization of boolean impressions are used to accomplish correction. An algorithm for extracting all of a collection of multiple expressions' kernels and co-kernels. This technique has been extended to handle multidimensional impressions and is comparable to the grain creation process. The key algorithm kernel is called for each expression in a collection of polynomial expressions and is repetitive. Index I and Cubed, which is the algorithm retrieved in the previous call, are the parameters for this function. The expression set's variables are sorted at random (the order is irrelevant), and the variable index reflects the variable's position in that order.

Designers have utilized software performance and size optimization for many years. Code optimization is a method of converting a high-level specification for a target processor into better machine code using compilers. Many scholars have attempted to enhance compilers in recent years. Researchers working on prototypes have come up with some incredible results. The bulk of optimizers are designed for high-performance or general-purpose computers. The development of powerful optimizing compilers for embedded processors has gotten little attention. Despite the fact that many academics are looking at automated code optimization approaches for embedded processors, the bulk of embedded processors (or DSPs) are still coded by experienced programmers, and code optimization is still mostly done by hand. Furthermore, as the market for portable devices grows, software optimization for power usage is becoming increasingly crucial.

As a result, improving code energy consumption is a key need of the system-level design approach for embedded devices. Several therapies have been proposed in the past to reduce energy use. Cathor and colleagues proposed an approach that combines automatic and manual program optimization with a focus on memory access. Tiwari et al. employ instruction-level energy models to implement compiler-driven assembly-level energy optimizations including instruction reordering, memory operation minimization, sweep operators in booth multipliers, and memory bank performance. Improved series processors as well. During the compilation process, several more enhancements were proposed, including energy-efficient register labeling, procedure inline, and loop enrolling, as well as instruction scheduling.

In other studies, many structural changes are implemented at the same time and the ensuing energy usage is tracked using simulation. All of these methods are aimed at automating instructions at a high level. Unfortunately, the existing capabilities are restricted. They can't handle mathematical corrections, as illustrated in the example in the introduction. The market requires a system-level design of low-cost integrated multimedia appliances with a short lead time. In an embedded system design context, hardware degrees of freedom are often limited, but degrees of freedom of software is generally broad. As a result, the basic requirement of embedded system-level design procedures is to effectively help improve code performance and energy usage. In order to meet the

demands of the market, it is necessary to automate as much as possible in the process of software design, starting from the algorithmic level.

Multiplying a variable by a set of multiple variable capabilities is a recurring activity in several digital signal processing (DSP) approaches. When compared to other frequent activities like addition, subtraction, and the usage of delay elements, multiplication is usually the costliest operation in the DSP algorithm. The quantity of logical resources utilized (i.e. the amount of silicon in an integrated circuit) and the speed with which it may be computed have a trade-off. Multiplication takes longer than logical resources, therefore each operation requires more logical resources at the same time. When multiplying two arbitrary variables, simple multiplication is necessary. When multiplied by a known constant, though, we may use binary multiplication's characteristics to create a less costly logic circuit. In practice, demonstrating a constant at one input is comparable to using a common multiplier. Because multiplication is somewhat expensive, performing a cheap multiplication procedure often results in considerable savings when considering the entire logic circuit.

Multiplication may also be the dominant operation, depending on the application. It is multiplied by a series of installments when multiplied by a permanent vector or permanent matrix. The dot product a b, for example, yields a scalar projection a on b. (or vice versa). Performing a dot product between several consecutive vectors (which collectively form a matrix) and a variable vector is what multiplying by a constant matrix entails (the elements of that vector are input). Multiplication from a constant matrix is therefore a linear change of points, which is useful in a variety of applications. To convert from RGB (red, green, blue) to YUV (yellow, green, blue) color space (Y denotes brightness, U and V represent chroma), for example, a constant 3x3 matrix must be multiplied.

Because the human eye is more sensitive to brightness than chroma, we can compress the information in components U and V with merely apparent distortions. Images and video are compressed using JPEG and MPEG, respectively. The static transform, such as the discrete foyer transform (DFT) and other Fourier-related transformers like the discrete cosine transform, multiply several discrete linear signals. DFT and Fourier-related transformers, as is well known, may be used for spectrum analysis, that is, determining if Sinuside supported the input signal at frequency F (for various F's). These changes can also be thought of as linear point changes, which can be utilized to compress signals. The objective is to transform the input signal into an integrated basis with only a few components containing the majority of the signal's energy (or information).

A group researcher describes in his paper how to solve an algebraic problem in a specific domain (Zory and Coelho, 2005). They solve their problem by using algebraic alterations to boost up the act of mathematical calculation concentrated process of modern system architecture because numerical possessions like associative else distributives permit the manipulation of a set of mathematically equivalent expressions. They also present attractive steps for solving mathematical problem on the basis of cost benefit model. They claim their results that applying modification leads to increase runtime performance. A group researchers discusses in their paper about optimization techniques those un-nest complex X-Path queries (Brantner et al., 2005). In this paper purpose Matthias classify X-Path expressions mainly associated to that properties that are related for un-nesting technique (Brantner et al., 2005). They also use a way to present mathematical expression into un-nested algebraic expression. In his conducted experiment he makes a comparison between evaluations of real time with offered x-path evaluators. Evaluations of x-path expression are conducted due to variety of reason:

- Algebraic Expressions are work out in a nested manner.
- Algebraic Expressions preserve in extremely nested manner.

A group researchers introduce a new technique that can reduce any high order random field with binary label into first order one that has the same minimum as the original (Hiroshi Ishikawa, 2009). He uses two algorithms to optimize high order multi-level problem. The problem discuss in uses field of expert model to optimized high order energy (Hiroshi Ishikawa, 2009). The result shows that the algorithms reduce Boolean function into an equivalent quadratic one, this approach optimize the capability and speed. Main focus of is to validate the techniques by minimizing a third order potential for imaging demolishing (Hiroshi Ishikawa, 2009). The goal of Gopalakrishnanet-al is to create algebraic techniques for performing such a transformation and to propose a methodology for integrating it with CSE to further increase the possibilities for optimization (Hiroshi Ishikawa, 2009; Gopalakrishnan and PriyankKalla, 2007). Common sub-expression elimination (CSE) is a constructive optimization approach in which the arithmetic expression tree of isomorphic patterns is identified and merged.

This method lowers the expenses of repeatedly implementing many copies of the same expression. However, as previously stated, CSE has limited optimization potential if the common expressions are not revealed in the polynomial system form. This integrated technique outperforms traditional approaches in developing area efficient hardware implementations of polynomial systems, according to the results. Some researcher develop algebraic strategies that are capable of generating linear equations utilizing coefficients and cubes from a polynomial system (Gopalakrishnan and PriyankKalla, 2007). When compared to current data route synthesis approaches, the experimental findings show that the methodology saves a substantial amount of space. Polynomial expressions are commonly used in a wide range of applications. Because CSE is not well adapted to handling polynomial expressions, designers frequently resort to manually optimizing these equations (Gopalakrishnan and PriyankKalla, 2007). At the same study also uses algebraic techniques initially used for multilevel circuit synthesis to factor and eliminate common subexpressions in polynomial formulations (Gopalakrishnan and PriyankKalla, 2007).

On a set of benchmark polynomial expressions, the proposed Algorithm was tested (Gopalakrishnan and PriyankKalla, 2007). The techniques described in are used to minimize the number of operations in a collection of polynomial expressions (Gopalakrishnan and PriyankKalla, 2007). The results demonstrate that these algebraic approaches are used in the algorithms for optimizing polynomial expressions. Many early works on optimizing algebraic expressions have been published (Aho and Johnson, 1976; Sethi and Ullman, 1970). In recent study the authors discuss an algorithm for reducing the number of steps and storage reference in the computation of algebraic expression (Aho and Johnson, 1976; Sethi and Ullman, 1970). The paper later extended to include expression with common sub expression elimination (Gopalakrishnan and PriyankKalla, 2007). Encouraged by these results, we plan to enhance our approach by incorporating further optimization techniques into it.

Work has recently been done to increase the performance of data extraction activities in particular circumstances. Provide ways to improve the specified entity's identification while also providing a technique for cutting documents, for example. Consider the challenge of successfully assembling several extraction modules into a bigger program as a separate concurrent endeavor. IE predictors and users are considered basic extraction modules. Explain if or not this Internet Explorer prediction fits certain criteria. As a result, specialized optimization approaches can be used. On the other hand, we recommend using a text-specific algebra. Existing optimization approaches and operations that take advantage of Characteristics. It's simple to improve comparable impressions that just include free impressions, and it doesn't need cardinality. We may work on the independent sections individually and then combine the results. It's worth noting that no matter how the expression's operators evaluate the node set, the outcome is always the same boolean value. As a result, material expenses are minimal.

There are numerous common sub-expressions in a multiplayer system describing a mathematical data route. In such systems, General Sub-Expression Elimination (CSE) is a helpful corrective approach that identifies and integrates isomorphic patterns in a mathematical expression tree. The cost of applying numerous copies of the same phrase is reduced as a result. However, if the common consequences are not expressed in the multi-system representation, the CSE's reform potential is restricted. As a result, changing a representation to disclose the more common sub-expressions gives more CSE correction. The goal of this paper is to present an algebraic technique. RTL's comprehensive control/data flow graphs (CDFGs) extraction, as well as scheduling, resource sharing, retention, and control synthesis, are all well-versed in modern high-end synthesis tools. They are, however, limited in their capacity to employ contemporary algebraic operations to lower implementation costs. As a result, there has been an increase in interest in investigating the application of algebraic techniques for RTL synthesis of mathematical data. For successful synthesis, the authors use multinational estimations to create novel multimodal models of complicated computational blocks.

In symbolic computers, algebra tools are used to look for the disintegration of a given polygon based on the components in a design library. Other algebraic transformations, such as factoring with the removal of common sub-expressions, leveraging the structure of mathematical circuits, rewriting words, and employing the Taylor Expansion Diagram (TED), have also been investigated for efficient hardware synthesis. Data flow transformations, for example. The lowest

level in the Sophos library is the algebraic style for function declaration, which describes some operations for multidimensional mesh. Since the mesh class is parameterized by class T, all mesh operations are parameterized by T. Float or scalar field classes will be common parameters.

The actions taken are intended to act as a pure process, which means that they do not modify their arguments or update any internal situation. Such methods are usually easier to deal with and explain the logic because there are no unintended consequences for their environment. It is required to examine the program, the combination of self-mutating operators and function announcements, and match them with the kinds of operators in order to transform algebraic expressions into self-mutation forms while concurrently decreasing temporal types. After any overloading is resolved, the functions are actually used. Temporary announcements must also be accompanied by the proper kind. Other source-to-source optimizations might be accomplished with such a pre-processor. In reality, the second technique is the same as what Code Boost does.

Starting with the starting count, the optimizer applies the rules of change to logical expressions over and over again in order to generate equivalent versions of that expression. Will come up with As it is performed, joining affiliation is an illustration of such a modification in a connected question. In time filtering or scientific computations, Fourier space is used. The selection of appropriate change rules is crucial in the reform of scientific arithmetic. Given the difficulties of numerical precision, unique concerns should be considered, such as the equation of two-digit impressions. The reformer must then find a set of physical algorithms that can process or execute each expression once the applicable change laws have been used to generate logical impressions. This procedure is carried out by following the rules for logical impression implementation. Each rule has one or more translations. In one or more physical operators, there are logical operators (algorithms).

A join operator, for example, can be implemented using an integration or hash-based technique, but an interpolation can be done using any curve fitting algorithm. Physical concerns such as data arrangement are taken into account by the optimizer when picking physical algorithms. The database implementer's job is to make logical modifications, give implementation guidelines, and explain the optimizer's cost model. New operations and algorithms, as well as their changes and implementation rules, may need to be introduced to a scientific database system as it is utilized, especially when the operator sets that we are computationally using change. Aren't finished (which will present difficulties in optimization). Such ongoing growth is possible with an extensible modifier. The database implementer will be in charge of managing this expansion.

2.METHODOLOGY

An algebra expression is combination of operand and operator. Evaluating algebraic expressions is a simple process, but needs to follow set of rules for operations to get the desire results; the ordering details of operations are given according to expressions. Algebra is simple language but learning algebra is like learning another language. Although, algebraic problem is a simple to solve but it is sometime used to create mathematical models of real-world conditions and to hold the problems that we can't solve just by using calculation and another optimization technique. The goal of query optimization is to devise a strategy for implementing a query that minimizes the cost function. The optimization method is divided into two stages, each of which uses traditional tools like heuristic optimization and systematic cost estimates. The first level is based on theoretical principles for organizing operations in a query implementation strategy for an equivalent phrase that should enhance performance.

A system-based cost model is used at the second level. Information for deciding on a cost-effective implementation strategy. For scientific data, many sophisticated data models have been developed that capture both regular and irregular grids, as well as topologies that are slightly different from geometry. However, modifications to an algebraic grid structure are not supported, and experimental results are not published, making it unclear whether performance requirements can be met. Others have shown that relative databases are insufficient for large scientific datasets. Viewing scientific datasets as external data sources and accessing them using SQL standards to handle external data is an option (SQL-MED). Papini et al. Claim to have had some success in giving space to instructions to deal with turbulent impressions.

Many scientific data processes are covered by standard database systems. The capacity to integrate different datasets, including data from numerous experiments and historical data from previous studies, is becoming increasingly important as the amount of data available to scientists grows. Despite the fact that database systems are capable of processing enormous amounts of data, they lack the capability for the numerical computations required for scientific data transmission. This constraint restricts scientific users' access to database systems. This study focuses on the notion of a database query with numerical computations on a scientific database, integrating standard database operators for patterns that are comparable to scientific and numerical operators to build an integrated algebra. We acquire the potential to automate the entire computation as a result of this integration into a single, common algebra, with the benefits of query optimization, algorithm selection, and data freedom.

Scientific databases can be used to compute. Database Optimizer may now control significant sections of the program since the barrier between database systems and computation has been removed. Existing database systems have seen considerable performance gains as a result of the enhancements. We show that many of the algebraic optimization approaches utilized in existing database optimizers may be extended to scientific computations available through optimization in traditional database systems by looking at operators on Time Series and Spectra. Enhances efficiency. Furthermore, optimization frees the user from cancer through the selection of computation stages, algorithms, indicators, and other physical features such as data distribution. As a result, correction provides the user data freedom, which is beneficial to both scientists and non-scientists.

The optimizer is given a phrase with logical operators on the bulk data type to enhance computation on a scientific database system. Sets (relationships), time series, and spectra are the bulk kinds we investigate. The type system is extended to the scientific level using time series and spectra. Each operator in the computation accepts one or more instances of the bulk type as input and outputs a new instance of the bulk type. Physical concerns such as index availability, seat configuration, and particular selection are not included in the initial count supplied by the user to the optimizer.

Example: Case Study; Standard Method, Proposed Method

2.1 Example by Proposed method

(Three Address Code)
Expression: 6*a-(4*a-3) =0
Let take the value of A=2
t0=2;
a=t0;
t1=6;
t2=6*a;
t3=4;
t4=4*a;
t5=3;
t6=t4-t5;
t7=t2-t6;

BY Applying Common sub Expression Elimination, We have

t0=2;
a=t0;
t1=6;
t2=t1*t0;
t3=4;
t4=t3*t0;
t5=3;
t6=t4-t5;
t7=t2-t6;

By Applying Copy propagation Elimination, We have

t0=2;
a=t0;
t1=6;
t2=6*2;
t3=4;
t4=4*2;
t5=3;
t6=t4-3;
t7=t2-t6;

By Applying Dead code Elimination we have

t0=2
a=t0;
t1=6;

t2=t1*t0;
t3=4;
t4=t3*t0;
t5=3;
t6=t4-3;
t7=t2-t6;

After Dead code Elimination, we have

t2=6*2;
t4=4*2;
t6=t4-3;
t7=t2-t6;

By Apply Constant folding

t2=12;
t4=8;
t6=t4-3;
t7=t2-t6;

By Apply copy propagation, we have

t2=12;
t4=8;
t6=8-3;
t7=12-t6;

By Apply Constant folding, we have

t2=12;
t4=8;
t6=5;
t7=12-t6;

By Apply copy propagation, we have

t2=12;
t4=8;
t6=5;
t7=12-5;

By Apply Constant folding, we have

t2=12;
t4=8;
t6=5;
t7=7;

By Apply the Packing Temporizes, we have

Table 1: Apply the packing temporizes on variables

Variable	Availability	After packing
T0		•
T1		•
T2	•	•
T3		•
T4	•	
T5		
T6	•	
T7	•	

Replace t2 by the First available temp variable

Replace t4 by the 2nd available temp variable

Replace t3 by the 3rd available temp variable

Replace t7 by the 4th available temp variable

The statement will be:

t2=12; t0=12;

t4=8; t1=8;

t6=5; t2=5;

t7=7; t3=7;

Comparisons of proposed Original and Optimized code

Table 2: Comparisons of proposed Original and Optimized code

	Original	Optimized
Number of Statements	9	4
Number of temp variable	8	4
Named Variable	1	0
Addition	0	0
Subtraction	2	0
Multiplication	2	0
Division	0	0

2.2 Standard Method solved by TAC

6*a-(4*a-3) =0
6a-4a+3=0
2a+3=0

Optimized code of algebraic expression

t1=2
t2=a
t3=2*a
t4=3
t5=t3+t4

Applying copy propagation, we have

t1=2
t2=a
t3=t1*t2
t4=3
t5=t3+t4

Applying Dead Code Elimination, we have

t3=t1*t 2
T5=t3+t4

Applying Reduction in strength, we have

t3=a+a;
t5=t3+t4;

By Applying the Packing Temporizes

Table 3: Apply the packing temporizes on variables

Variable	Availability	After packing
t0		•
T1		•
T2		•
T3	•	
T4	•	
T5	•	

Replace t3 by the First available temp variable

Replace t5 by the 2nd available temp variable

Replace t4 by the 3rd available temp variable

The statement will be

t0=a+a;

t1=t2+t0;

Comparisons of Standard Original and Optimized code

Table 4: Comparisons of Standard Original and Optimized code		
	Original	**Optimized**
Number of Statements	5	2
Number of temp variable	5	2
Named Variable	1	1
Addition	1	2
Subtraction	0	0
Multiplication	1	0
Division	0	0

The Graphical Comparison shown in Fig No.2.

Example2: Proposed method

Expression: 24a+6b+6a+6b-36 =0

Simpler way of original equation through TAC

t0=24;
t1=a;
t2=24*a;
t3=6;
t4=b;
t5=6*b;
t6=6;
t7=a;
t8=6*a;
t9=6;
t10=b;
t11=6*b;
t12=36;
t13=t2+t8;
t14=t5+t11;
t15=t13+t14-12;

By Applying Common Sub Expression, we have

t0=24;
t1=a;
t2=24*a;
t3=6;
t4=b;
t5=6*b;
t6=6;
t7=t1;
t8=6*a;
t9=t3;
t10=t4;
t11=t5;
t12=36;
t13=t2+t8;
t14=t5+t11;
t15=t13+t14-12;

By Applying copy propagation, we have

t0=24;
t1=a;
t2=t0*t1;
t3=6;
t4=b;
t5=t3*t4;
t6=t3;
t7=t1;
t8=t3*t1;
t9=t3;
t10=t4;
t11=t5;
t12=36;
t13=t2+t8;
t14=t5+t11;
t15=t13+t14-12;

By Applying Dead code Elimination, we have

t2=t0*t1;
t5=t3*t4;
t8=t3*t1;
t9=t3;
t10=t4;
t11=t5;
t12=36;
t13=t2+t8;
t14=t5+t11;
t15=t13+t14-12;

By Applying Dead code Elimination, we have

t2=t0*t1;
t5=t3*t4;
t8=t3*t1;
t11=t5;
t13=t2+t8;
t14=t5+t11;
t15=t13+t14-12;

By Applying Dead code Elimination, we have

t2=t0*t1;
t5=t3*t4;
t8=t3*t1;
t13=t2+t8;
t14=t5;
t15=t13+t14-12;

Common identities used are

T2=24*a;
T5=6*b;
T8=6*a;
t13=t2+t8;
t14=t5;
t15=t13+t14-12;

Applying Reduction in strength, we have

t2=12a+12a;
t5=3b+3b;
t8=3a+3a;
t13=t2+t8;
t14=t5;
t15=t13+t14-12;

By Applying the Packing Temporizes

Table 5: Applying the Packing Temporizes		
Variable	**Availability**	**After packing**
t0		
T1		
T2	•	•
T3		
T4		
T5	•	•
T6		•
T7		•
T8	•	•
T9		•
T10		•
T11		•
T12		•
T13	•	•
T14	•	•
T15	•	•

Replace t2 by the First available temp variable

Replace t5 by the 2nd available temp variable

Replace t8 by the 3rd available temp variable

Replace t13 by the 4th available temp variable

Replace t14 by the 5th available temp variable

Replace t15 by the 6th available temp variable

The statement will be

t0=12a+12a;
t1=3b+3b;
t2=3a+3a;
t3=t0+t2;
t4=t1;
t5=t3+t4-12;

3.Simulation And Results

Although well-understood and effective, algebraic optimization techniques for relational database languages cannot be completely applied to the layered relational model. As a result, the algebraic characteristics of layered relational operators are significant in both theory and practice. In this work, we look at the problem and come up with a set of algebraic equivalences.

Comparisons of Proposed Original and Optimized code

Table 6: Comparisons of Proposed Original and Optimized code

	Original	Optimized
Number of Statements	16	6
Number of temp variable	16	6
Named Variable	2	2
Addition	3	5
Subtraction	1	1
Multiplication	4	0
Division	0	0

Graphical Comparison shown in Fig No.3.
Example: Standard Method:

24a+6b+6a+6b-36 =0

Replacing complex linear equation by simpler equation

30a+12b-36=0
t0 =30;
t1=a;
t2=30*a;
t3=12;
t4=b;
t5=12*b;
t6=36;
t7=t2+t5-t6;

BY Apply Common sub Expression

t0 =30;
t1=a;
t2=30*t1;
t3=12;
t4=b;
t5=12*t4;
t6=36;
t7=t2+t5-t6;

By Apply Dead code Elimination

t2=30*t1;
t5=12*t4;
t7=t2+t5-t6;

Common identities used are

T2=30*a;
T5=12*b;

Graphical Representation of the Expression: 6*a-(4*a-3) =0

T7=t2+t5-t6;

Applying Reduction in strength, we have

t2=15a+15a;
t5=6b+6b;
t7=t2+t5-t6;

By Applying the Packing Temporizes

Table 7: Applying the Packing Temporizes

Variable	Availability	After packing
t0		
T1	•	
T2	•	•
T3		
T4	•	
T5	•	•
T6	•	•
T7	•	•

Replace t2 by the First available temp variable

Replace t5 by the 2nd available temp variable

Replace t7 by the 3rd available temp variable

Replace t6 by the 4th available temp variable

The statement will be

T0=15a+15a;
T1=6b+6b;
T2=t0+t1-t3;

Comparisons of Standard Original and Optimized code

Table 8: Comparisons of Proposed Original and Optimized code

	Original	Optimized
Number of Statements	8	3
Number of temp variable	8	4
Named Variable	2	2
Addition	1	3
Subtraction	1	1
Multiplication	2	0
Division	0	0

Graphical Comparison shown in Fig No.4.

Graphical Representation of the Expression: 6*a-(4*a-3) =0

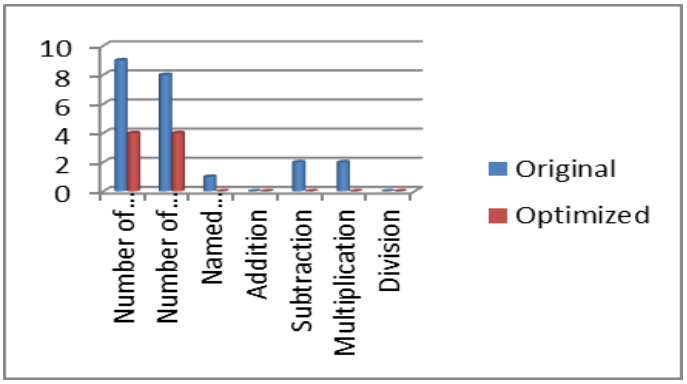


Figure 1: Comparison of Algebraic Expression optimization by standard and proposed approach

6a-4a+3=0
2a+3=0

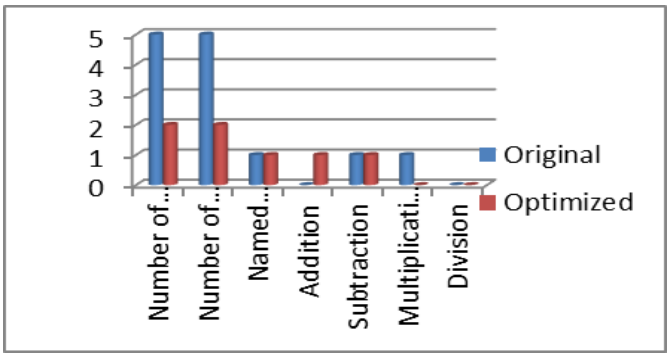


Figure 2: Graphical Representation of the Expression: 24a+6b+6a+6b-36 =0

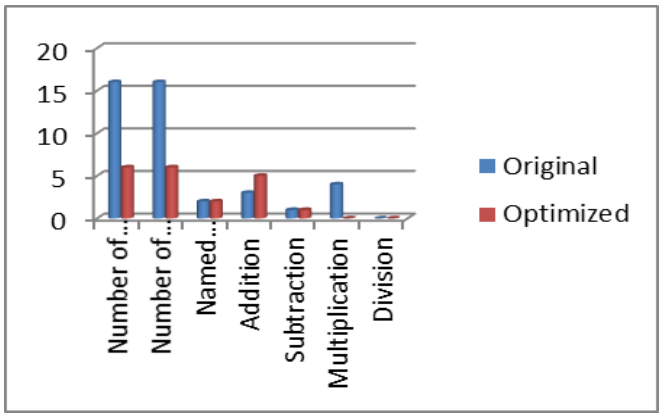


Figure 3: Graphical Representation of the Expression: 30a+12b-36=0

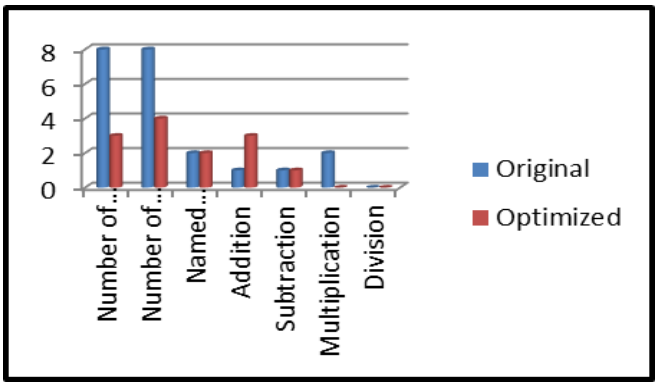


Figure 4: Graphical Comparison Representation

4. Conclusion

The main aim of this research is to improve three address code optimizations for an algebraic expression through substitution method. By using this approach, we conclude that we achieve better results than previous work done using existing optimization technique. Our propose technique utilize less resources with minimum time. We can also modify any complex algebraic expression using our propose optimization technique. During the experimental work we observe that our propose technique attract the user because it is quite simple flexible and more efficiently produce the results using minimum number of parameters.

References

Aho, A.V., and Johnson, S.C., 1976. Optimal code generation for expression trees. Journal of the ACM, 23(3), 488–501. https://doi.org/10.1145/321958.321970

Brantner, M., Kanne, C.C., Helmer, S., and Moerkotte, G. 2005. Full-fledged algebraic XPath processing in Natix. In Proceedings of the International Conference on Data Engineering (ICDE) (pp. 705–716).

Gopalakrishnan, S., and Kalla, P. 2006. Algebraic techniques to enhance common subexpression elimination for polynomial system synthesis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(10), Pp. 2012–2022. https://doi.org/10.1109/TCAD.2006.871385

Gopalakrishnan, S., and Kalla, P. 2007. Optimization of polynomial datapaths using finite ring algebra. ACM Transactions on Design Automation of Electronic Systems, 12(4), Article 49.

Ishikawa, H., 2009. High-order cliques reduction in binary graph order [Unpublished doctoral dissertation or technical report]. Nagoya City University, Japan.

Sethi, R., and Ullman, J. D. 1970. The generation of optimal code for arithmetic expressions. Communications of the ACM, 13(12), 715–728. https://doi.org/10.1145/362790.362812

Zory, J., and Coelho, F. 2005. Using algebraic transformations to optimize expression evaluation in scientific code. Centre de Recherche en Informatique, École Nationale Supérieure des Mines de Paris, Fontainebleau, France.