Discovering Numerous Strassen's Equivalent Equations Using a Simple Micro Multimodal GA: Evolution in Action

Azam Asilian Bidgoli*, Steven Trumble, Shahryar Rahnamayan, SMIEEE
Nature Inspired Computational Intelligence (NICI) Lab
Department of Electrical, Computer, and Software Engineering
Ontario Tech University, Oshawa, Canada
azam.asilianbidgoli@uoit.ca, steven.trumble@ontariotechu.net, shahryar.rahnamayan@uoit.ca

***Abstract*—Solving real-world complex optimization problems using simple metaheuristic algorithms is a challenging but attractive task. Making matrix multiplication efficient is one of the interesting problems. This time-consuming algebric operation is required in many applications in science and engineering, thus reducing its complexity targets more efficient computation. In fact, many of practical and theatrical complicated calculations can be modeled efficiently as matrix-based operations, therefore matrix multiplication is computationally expansive operator among all others. In this paper, a simple metaheuristic method based on Micro Genetic Algorithm is proposed to find Strassen's equivalent solutions which is an algebraic method to compute the product of two matrices with minimal number of multiplications. Since there are numerous optimal solutions, the modeled problem is a large-scale and highly multi-modal optimization problem. The proposed method could find more than $160,000$ valid solutions with same complexity as Strassen's in a large-scale search space. Among all discovered solutions found using the proposed method, there are 701 distinct solutions which is the maximum number of discovered Strassen's equivalent solutions to the best of our knowledge. The proposed algorithm is simple but very efficient to find more and more solutions, in fact, that is a great demonstration of "evolution in action" to tackle real-world complex problems like the current one, which just one set of its equations has been discovered by the Germen mathematicians and has remained mysterious for more than 50 years.**

***Index Terms*—Matrix multiplication, Strassen algorithm, Genetic algorithm, Multimodal optimization problem, Restarting strategy, Real-world problems, Evolution in action.**

I. Introduction

Key matrix multiplication is an extremely important algebraic operation with numerous applications in science and engineering. That is the main motivation of proposing a variety of algorithms to reduce time complexity of matrix multiplication. The simplest technique to calculate the resultant matrix of multiplication of two matrices of size $n \times n$ has the time complexity of $O(n^3)$ because the resulting matrix has n^2 elements and each needs n scalar multiplications and $n-1$ additions. Many studies have been conducted to reduce the number of multiplications. One of the well-known methods is the Strassen algorithm which decreases the number of multiplications using a smart technique proposed by Arnold Schönhage and Volker Strassen in 1971 [1]. By using this method, the number of multiplications for two matrices of size 2×2 has been reduced to 7. The algorithm works based on recursive strategy to calculate the multiplication of $n \times n$ matrices by dividing into $n/2 \times n/2$ submatrices and calculating them. The most challenging part of this algorithm is finding the bilinear combinations of submatrices as the optimal solution. Searching for these combining equations is difficult, because the search space of the problem is extremely large. Additionally, there are numerous optimal solutions for this problem, as a result, this problem can be defined as a highly multimodal large-scale optimization problem.

There are several conducted studies that have utilized metaheuristic algorithms to solve this problem. To the best of our knowledge, the first time that an evolutionary algorithm could find only one Strassen equivalent solution for 2×2 matrix was reported in [2]. Oh and Moon used Genetic Algorithm (GA) [3] to find as many as possible solutions [4] for 2×2 matrices. They succeed to discover 608 distinct solutions in nine recognized groups. Ant colony algorithm is a metaheuristic and biology-inspired method that is combined with Gaussian Eliminations to reduce the number of variables and to solve this optimization problem [5]. A significant achievement of that study was discovery of a new group of solutions comparing to previous works. In addition to proposing the new methods, several efforts have been made on parallelization of existing methods to accelerate finding more solutions [6]–[8]. A parallel GA has been proposed in [9] to find a solution for 3×3 matrices with 23 multiplications. The algorithm could find an accurate solution with 23 and an approximate solution with 22 multiplications. Morancho [10] has proposed a parallel implementation of Gaussian Elimination [11] which is a key algorithm in algebra. His evaluation developed a case study that searches exhaustively for equations similar to Strassen's equations. The algorithm could find 20 Strassen equivalent solutions for 2×2 matrix multiplication. In addition to EAs, machine learning aims to solve this problem as well. For instance, an artificial neural network is proposed in [12] to learn Strassen multiplication. The network observes a set

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

of matrices as inputs and the product matrices as output to generate Strassen equations by learning network weights. The network was trained by 10^8 training samples with error less than 10^{-14}.

The main goal of this paper is proposing a Micro GA-based method to find as many as possible solutions for 2×2 matrices multiplication. The proposed method applies two restarting strategy: 1) partially reinitialization if there is no more improvement in the search process; 2) complete reinitialization after running the algorithm for a predefined number of iterations. These strategies along with a proposed local search lead to discovering more than $160K$ Starssen's equivalent solutions for 2×2 matrices multiplication; by letting algorithm to search more, it is able to find more and more solutions.

The remaining parts of this paper are organized as follows. Section II presents a background review on problem definition and GA. Section III details a description of the proposed method. In Section IV, the effectiveness of the proposed method has been indicated by experiments results. This paper is concluded in Section V.

II. Background Review

In this section, we give an explanation for concepts of Strassen-like matrix multiplication. The details of GA is also described in this section.

A. Genetic Algorithm

GA is a well-know biology-inspired algorithm to solve the optimization problems. The algorithm works in a repetitive procedure to find the optimal solution. At each iteration, some stochastic changes are applied to modify current candidate solutions to generate new ones. Similar to other population-based optimization algorithms, GA needs a population of chromosome (i.e., individuals) to move toward the optimal solution(s) by evolving the population. For this purpose, the algorithm utilizes crossover and mutation operators to generate offsprings from current parents in the population.

The offsprings inherit the characteristics of the parents and will be added to the next generation if they have better fitness values copmared to their parents. By this method, generations will be improved in terms of fitness values. Therefore, that is expected that the algorithm could find optimal solution after an appropriate number of iterations.

Genetic algorithms are being used to solve a wide variety of the problems. There are some studies on utilizing GA for tackling mathematical problems [13]–[15]. Despite of its popularity, GA has weaknesses for solving complicated and large-scale problems. Particularly, it is slow in fine-tuning. Therefore, a wide variety of modifications are performed on GA to improve its performance. Some modifications are tailored for a specific application [16]–[18]. An alternative version of the GA is Micro GA that has been proposed for large-scale optimization problems [19]. A very small population (e.g., 5) is utilized in Micro GA to tackle the optimization problem. Evolving very small populations gives benefits of less fitness calls during the generations and locating promising areas of the search space. Because small populations are unable to maintain diversity for many generations, restarting the search process prevents the premature convergence.

B. Modeling problem as an optimization problem

In order to calculate the multiplication of two matrices, A and B of size $n \times n$, assuming n is a power of 2, the matrix would be divided by order 2 recursively till we get the matrix of 2×2. This is a divide-and-conquer method in which we can contain four $n/2 \times n/2$ submatrices from A, B, and C which is the multiplication matrix. The matrices A and B are written as block matrices,

$$A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}, B = \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix}, C = \begin{pmatrix} C_1 & C_2 \\ C_3 & C_4 \end{pmatrix}, \quad (1)$$

where the submatrices of C are as follows.

$$\begin{aligned} C_1 &= A_1B_1 + A_2B_2, \\ C_2 &= A_1B_3 + A_2B_4, \\ C_3 &= A_3B_1 + A_4B_2, \\ C_4 &= A_3B_3 + A_4B_4, \end{aligned} \quad (2)$$

In the mentioned divide-and-conquer method, there are 8 recursive calls of multiplication operation. The running time for multiplying two matrices A and B is $T(n) = 8T(n/2) + O(n^2)$. Analysis of the recursive equation by Master Theorem [20] shows that this algorithm will run in $O(n^3)$ time. Strassen method has been proposed to reduce the number of recursive calls to seven for 2×2 matrices. For this purpose, the equations of multiplication calculation are consist of sum of product expressions. As a result, seven product equations, P_i's, along with 4 sum equations, C_i's which are linear combinations of P_i's are produced as a Strassen solution for matrix multiplication.

$$\begin{aligned} P_1 &= A_1(B_3 - B_4) \\ P_2 &= (A_1 + A_2)B_4 \\ P_3 &= (A_3 + A_4)B_1 \\ P_4 &= A_4(-B_1 + B_2) \\ P_5 &= (A_1 + A_4)(B_1 + B_4) \\ P_6 &= (A_2 - A_4)(B_2 + B_4) \\ P_7 &= (-A_1 + A_3)(B_1 + B_3) \\ C_1 &= -P_2 + P_4 + P_5 + P_6 \\ C_2 &= P_1 + P_2 \\ C_3 &= P_3 + P_4 \\ C_4 &= P_1 - P_3 + P_5 + P_7 \end{aligned} \quad (3)$$

By decreasing the number of multiplications, the running time of Strassen method is $T(n) = 7T(n/2) + O(n^2)$ and the time complexity would be $O(n^{2.81})$. As it is well-known as the Strassen method, each multiplication equation, P_i consists of bi-linear combination of submatrices of A_i's and B_i's. The coefficient of these submatrices in P_i's can be -1, 0, or 1.

Accordingly, a general form of each multiplication equation can be defined as follows.

$$\begin{aligned} P =&(\alpha_1 A_1 + \alpha_2 A_2 + \alpha_3 A_3 + \alpha_4 A_4).\\ &(\beta_1 B_1 + \beta_2 B_2 + \beta_3 B_3 + \beta_4 B_4), \end{aligned} \tag{4}$$

where $\alpha, \beta \in \{-1, 0, 1\}$ are coefficients which are desirable to be found in a defined optimization problem. Therefore, in the search process, finding a solution for matrix multiplication is equivalent to finding such coefficients (α's and β's) so that by discovering a linear combination of P_i's, the submatrices of resultant matrix, C_i's, can be calculated. Obviously, Strassen's algorithm needs seven multiplications and 18 additions. Also, a set of similar equations with seven multiplications and only 15 additions is presented by Winograd [21].

III. Proposed method

The details of proposed genetic algorithm to solve 2×2 matrix multiplication along with used representation scheme of the problem to encode the search space are explained in this section.

A. Encoding

For solving optimization problems using metaheuristic algorithms, it is required to specify the representation of individual in the population. Based on the formulation of the matrix multiplication indicated in Eq. 4, we need to find 8 coefficients for each P_i. Therefore, for constructing seven P_i, $7 \times 8 = 56$ coefficients are needed to be obtained by using an optimization algorithm [4]. Accordingly, a 7×8 matrix would be the representation of each individual in the population which ith row indicates the coefficients ($\alpha_{i,j}, \beta_{i,j} \in \{-1, 0, 1\}$) for P_i. Consequently, as an initialization step, a uniform random population of 7×8 matrix chromosomes is generated.

B. Fitness function

As previously mentioned, the optimization algorithm is responsible to find the coefficients in P_i's. By finding the coefficients, we would be able to combine them to form a solution for matrix multiplication. In fact, in order to evaluate the quality of an individual, it is required to investigate that using the coefficients vector whether there is any bi-linear combination of P_i's to be a solution. For this purpose, an expansion format is utilized to define a fitness function to evaluate each individual based on solving a simple equation [9]. Based on the previously mentioned problem definition, the coefficients in each P_i can be expanded into a vector with 16 members as follows.

$$[\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4] \times [\beta_1\ \beta_2\ , \beta_3\ , \beta_4\ ] = [\lambda_1\ \lambda_2\ ...\ \lambda_{16}], \tag{5}$$

where each λ indicates the presence of one of the term $A_j \times B_k$ $(j, k = 1, 2, 3, 4)$ in the P_i. So by considering new representation of coefficients (λ), P_i can be calculated as follows.

$$P_i = \sum_{j=1}^{16} \lambda_{i,j} A_{\lceil j/4 \rceil} B_{((j-1) \pmod 4)+1}, \tag{6}$$

where $\lambda_{i,j}$ indicates the presence or absence of $A_{\lceil j/4 \rceil} B_{((j-1) \pmod 4)+1}$ in P_i. For instance P_i =[0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0] is a vector representation for $P_i = A_1 B_2 + A_1 B_4 + A_3 B_3$.

By considering the above template of representation, matrix multiplication can be defined as a linear equation as follows.

$$C = \Lambda \Delta, \tag{7}$$

where Λ is a 16×7 matrix of λ_i's and Δ is a 7×4 matrix of δ_i's which are the coefficients for bi-linear combinations of P_i's. Consequently, C is a 16×4 matrix representing the C_i's submatrices. As an example, three matrices of Strassen solution for C_1 is presented as follows.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Optimization algorithm attempts to find solutions to solve the Eq. 7. In order to investigate that an individual is a valid solution for mentioned equation, we expand the chromosome to λ matrix, then the algorithm check whether there is any Δ matrix to define a bi-linear combination of P_i's to obtain C_i or not.

As a result, for a candidate solution, the difference between the actual C and C' which is obtained by candidate solution can be defined as the following error:

$$Err(X) = \min ||C' - C|| = \min ||X\Delta - C||, \tag{8}$$

where X is expanded matrix of a candidate solution in the population. The value of the Err would be 0 if X be a valid solution. In order to calculate the error function, we need the value of Δ. By taking the gradient of the error function which is $X^T X \Delta - X^T C$, Δ would be obtained in zero point of the gradient as follows.

$$\Delta = (X^T X)^{-1} X^T C \tag{9}$$

By substituting obtained Δ in error function, the final equation is calculated as:

$$Err(X) = ||X(X^T X)^{-1} X^T C - C|| \tag{10}$$

Finally, the fitness function is designated as follows which maximizing this function leads to minimizing the error of found solution.

$$F(X) = \frac{1}{1 + Err(X)} \tag{11}$$

Thus, $F(X^*)$ is equal to 1 where X^* is the optimal solution [9].

C. Micro Genetic Algorithm to find Strassen's equivalent equations

In order to find the Strassen-like solutions of matrix multiplication, Micro GA has been utilized with tailored modification and operators. Micro GA is a version of GA with a small population size. A small number of individuals aims to limited fitness calls in one iteration of running the algorithm. As mentioned in Section II, for large-scale optimization problems, this version of GA is more beneficial if the premature convergence of the algorithm is prevented by using a restarting strategy. The details of such strategy is provided in the following subsection. The algorithm starts the optimization process using a smart initialization. Because the solutions are sparse vectors which their variables have mostly zero values, the initial values will be selected from $\{-1, 0, 1\}$ by allocating different probabilities. So that a higher value of probability will be considered for assigning 0 value to the variables and same probabilities for $1, -1$. Each initial chromosome should be evaluated based on defined fitness function.

Similar to other EAs, new members should be added to population to move toward the optimal solutions. Accordingly, for each member of the population, single-point crossover is applied on corresponding member as the first parent and a random selected candidate solution as the second parent to generate two offspring vectors. In this type of crossover, a point is picked randomly, and variables to the right of that point are swapped between the two parent chromosomes. As previously mentioned, the number of variables are 56, so a random number in $(0, 56)$ is selected as the breaking point.

In order to apply mutation operator on two offspring vectors, according to the mutation rate, some variables are selected. The value of each selected variables would be changed to a feasible randomly selected value. As previously mentioned, a value in $\{-1, 0, 1\}$ should be assigned to each variable. Therefore, by considering same probability, the arbitrary variables are changed to something among other two options which are different from the current value of the variable. The two mutated vectors are compared with parent to select the best one among three vectors. If one of the offsprings outperforms the parent, the current individual would be replaced with better offspring, otherwise the parent will remain in the population.

D. Local Search

One of the strategies to aim EAs to find candidate solutions with better fitness values is the utilizing of the local search. Local search is based on the concept of neighborhood search. This strategy searches around a candidate solution to generate better points in the search space. A neighborhood of a candidate solution is a set of solutions that are in some sense close to corresponding one. To generate the neighbors, a variety of functions can be defined based on the structure of the problem. In matrix multiplication problem, because the the algorithm needs to tackle a very large-scale search space,

input : $ItMax$: Maximum number of iterations, NP: Population size, $PurgeRate$: Rate of purge
output: Solutions

while $true$ **do**
  // Initialization
  Generate NP random individuals as POP ;
  $It = 1$;
  $Improvement_{POP} = 0$;
  $Improvement_i = 0$; // For all individuals
  Calculate fitness function for each individual;
  // Main algorithm
  while $It < ItMax$ **do**
    for $i \leftarrow 1$ **to** NP **do** // For each individual, x_i, in the population
      // Crossover and Mutation operators
      Select a randomly individual, x_r, from population;
      $u_i = Crossover(x_i, x_r)$;
      $v_i = Mutate(u_i)$;
      // Local Search
      if $F(x_i) < F(v_i)$ **then**
        $Improvement_i = 1$; $POP(i) = v_i$;
      else
        if *!*$(Improvement_i)$ **then**
          $\text{Lx}_i = localsearch(x_i)$;
          if $F(x_i) < F(Lx_i)$ **then**
            $Improvement_i = 1$;
            $POP(i) = Lx_i$
          end
        end
      end
      if $F(x_i) < F(best)$ **then**
        $best = x_i$
      end
    end
    if $F(Pr\_best) < F(best)$ **then**
      $NImprovement_{POP} = 0$
    else
      $NImprovement_{POP} = NImprovement_{POP} + 1$
    end
    $Pr\_best = best$;
    // Purge strategy
    if $NImprovement_{POP} == 2$ **then**
      Reinitialize $PurgeRate\%$ of population;
      $NImprovement_{POP} = 0$;
    end
  end
  if $F(best) = 0$ **then**
    Return $best$ as a solution;
    Break;
  end
end

Algorithm 1: Pseudo-code for proposed method.

local search can lead to find more promising points around each candidate solution. In order to generate neighbors for a candidate solution, two other options for each variable would be evaluated. For instance if the value of a gene is 0, two vectors will be created by changing this value to 1 and -1 while all other genes are untouched. The current candidate solution is replaced with one of the generated vectors if the new vectors have better fitness value. Otherwise, the process continues by changing other variables of this candidate solution until an improvement is obtained. Thus, the maximum number of fitness evaluations for a candidate solution in local search step is $2\times7\times8$ that would be happened in the case that all neighbors of solution should be checked.

Despite of the benefits of local search, using this strategy for all individuals of the population at each iteration needs numerous fitness evaluations which increase the computational complexity of the algorithm. Thus, as an alternative scheme, the proposed method searches only around the individuals with no improvement during a generation. Based on this condition, at the end of each iteration, the algorithm checks the improvement of each candidate solution. For those candidate solutions which came from previous generation without any improvement, local search strategy is utilized to find better candidate solution in its neighborhood.

E. Restart strategy

As mentioned previously, the matrix multiplication is a large-scale and highly multi-modal optimization problem. The optimization algorithm can find many of the Strassen's equivalent solutions but finding as many as possible is the goal of the proposed method in this study. One of the strategies that aims to reach more promising regions is restarting strategy of the algorithm. On the other hand, restarting prevents the algorithm to get stuck in non-global optimum because of premature convergence. In the proposed method, two schemes of restarting occurs: partial restarting and complete restarting. At the first case, if the population doesn't have more improvement in two consecutive generations, a certain portion of the population are randomly selected to be re-initialized. The number of chromosomes which should be selected to be replaced with new vectors is a parameter ($PurgeRate$) that is set based on the other settings of the algorithm. The improvement of the population is defined according to updating the best candidate solution in the population. If in two consecutive iterations, the best candidate solution has not been changed, the new individuals will be injected to the population to move toward more promising regions. It should be noticed that removing a number of individuals will be done by not-preserving the best individual in the population.

The investigation of search progress discloses that the population may fail to improve after a few iterations. Therefore, as the second scheme of restarting, the search process will be restarted if the algorithm reach to a predefined number of iterations. In this case, the entire population are discarded and the search will restarted with a randomly-generated population as explained in population initialization phase.

The overall structure of the proposed method has been presented in the Algorithm 1. The algorithm starts with a randomly-generated individuals. New generation is created using crossover and mutation operators. The better solution in competition between parent and new related individuals remains in the population. Then local search is applied on candidate solutions with no improvement. To terminate the premature convergence of the search process, in the case of no more improvement of the population, a portion of the individuals will be replaced by newly generated chromosomes. After a predefined iterations, the search process is restarted to continue with the new population.

F. Experimental Results

Based on the proposed method, we found more than $160K$ Strassen's equivalent solutions for 2×2 matrix multiplication. The parameters for experiments are set as follows. The population size is set to 5 (because the GA algorithm is its Miro version), however we investigate the effect of this parameter on the speed of finding a solution. We run the algorithm with different number of individuals to see how the population size affect solving a large-scale optimization problem. The number of maximum iterations would be 240. The value for mutation rate and purge rate are 0.14 and 0.25, respectively. As mentioned previously, purge rate is the portion of population (randomly selected) which reinitialized if no improvement is made. Additionally, to initialize the population, the probability value for 0 is set to 0.5 while for other two options $\{1,-1\}$, it would be 0.25. Implementation is conducted in Python 3.7 on an Intel Core i5 @2.20 GHz computer with 12 GB RAM. By setting the parameters, the algorithm starts to run so that the total number of solutions that the algorithm could find during two weeks is 161,096. Because the number of nonzero entries in P_i's determines the number of operations in multiplication equations, the solutions are divided to 10 different types of solutions based on this characteristic. Table I displays a representative solution for each group along with the characteristic of each group. For instance, the characteristic of P_i's in Group 4 is 3333444 which ith digit indicates the number of nonzero elements for P_i; this group is related to Strassen solutions. The number of solutions found are represented in Table II. In this table, the characteristic of nonzero coefficients for Δ matrix are also presented. By adding the total nonzero elements for all equations in a solution, the total required number of summations and subtractions is indicated in the forth column of the table as number of nonzero elements.

By analyzing the power of search algorithm, that is worthy to mention that during the search process, the algorithm excludes solutions presented with different permutations of P_i's. However, from the optimization point of view, those solutions are distinct, because they have different values in variables. Therefore, each of them is a valid optimal solution and distinct point in the search landscape. The optimization algorithm finds the solutions based on the defined objective function, consequently, the number of found solutions indicates the the power of the algorithm. Although, by considering

TABLE I: Different groups of discovered solutions.

Group 1: 3333445	Group 2: 2245667	Group 3: 3344556
$P_1 = A_1(-B_1 - B_3)$ $P_2 = (A_3 - A_4)(-B_3)$ $P_3 = A_4(-B_3 - B_4)$ $P_4 = -A_2(-B_2 - B_4)$ $P_5 = (A_1 - A_3)(B_1 + B_2)$ $P_6 = (A_1 - A_2 - A_3 + A_4)B_2$ $P_7 = (A_1 + A_3 - A_4)(B_2 - B_3)$ $C_1 = -P_1 - P_2 - P_3 + P_5$ $C_2 = P_2 + P_3 - P_5 + P_7$ $C_3 = -P_1 - P_3 + P_5 - P_6$ $C_4 = -P_4 - P_5$	$P_1 = (A_1)(-B_1)$ $P_2 = (A_2)(-B_2)$ $P_3 = (A_3 - A_4)(-B_3 + B_4)$ $P_4 = A_3(B_1 + B2 - B_3 - B_4)$ $P_5 = (-A_1 + A_2 - A_3 + A_4)(B_1 + B_3)$ $P_6 = (A_2 + A_4)(-B_1 - B_2 - B_3 - B_4)$ $P_7 = (A_2 - A_3 + A_4)(-B_1 + B_2 - B_3 - B_4)$ $C_1 = -P_5 - P_6$ $C_2 = -P_1 - P_4 + P_5 - P_6 + P_7$ $C_3 = -0.5P_2 + 0.5P_3 + 0.5P_4 + P6$ $C_4 = -0.5P_2 - 0.5P_3 + 0.5P_4 + P6 - P_7$	$P_1 = -A_4(-B_2 - B_4)$ $P_2 = (-A_3)(-B_1 - B_3)$ $P_3 = (-A_2 + A_4)(-B_1 + B2)$ $P_4 = (A_1 + A_3)(-B_3 + B_4)$ $P_5 = (-A_1 - A_2 + A_3 + A_4)B_1$ $P_6 = (-A_1 - A_2 - A_3 - A_4)(-B_4)$ $P_7 = (A_1 + A_2 + A_3 - A_4)(B_1 + B_4)$ $C_1 = P_1 - P_2 - 0.5P_3 - 0.5P5 + 0.5P_7$ $C_2 = 0.5P_3 - P_4 + 0.5P_5 - P_6 + 0.5P7$ $C_3 = P_1 + 0.5P_3 - 0.5P_5 + P_7$ $C_4 = -0.5P_3 + 0.5P_5 - 0.5P_7$
Group 4: 3333444 (Strassen's Solution)	Group 5: 2266668	Group 6: 2244556 (Winograd's Solution)
$P_1 = (A_3 - A_4)B_3$ $P_2 = A_1(B_1 + B_2)$ $P_3 = (A_1 - A_2)B_2$ $P_4 = -A_4(-B_3 - B_4)$ $P_5 = (-A_1 + A_3)(-B_1 + B_3)$ $P_6 = (-A_2 + A_4)(B_2 - B_4)$ $P_7 = (A_1 - A_4)(-B_2 - B_3)$ $C_1 = P_4 - P_6$ $C_2 = P_3 - P_5 - P_6 + P_7$ $C_3 = P_1 - P_2 + P_4 + P_5$ $C_4 = P_1 + P_7$	$P_1 = (A_3)(-B_3)$ $P_2 = (-A_4)(B_4)$ $P_3 = (-A_2 - A_4)(-B_1 + B_2 - B_3 + B_4)$ $P_4 = (A_1 - A_3)(B_1 - B_2 - B_3 + B_4)$ $P_5 = (A_1 + A_2 + A_3 + A_4)(-B_1 - B_3)$ $P_6 = (-A_1 - A_2 + A_3 + A_4)(-B_2 + B_4)$ $P_7 = (-A_1 - A_2 + A_3 - A_4)(B_1 - B_2 + B_3 + B_4)$ $C_1 = -0.5P1 + 0.5P_2 - 0.5P - 3 + 0.5P_4 + P_5 + P_6$ $C_2 = -0.5P_1 - 0.5P_2 - P_5 + P_6 - 0.5P_7$ $C_3 = -0.5P_3 - 0.5P_4 + P_5 - P_6 + 0.5P_7$ $C_4 = -P_5 - P_6$	$P_1 = (-A_1)(B_1)$ $P_2 = (-A_2)(-B_2)$ $P_3 = (A_3 + A_4)(-B_1 - B_3)$ $P_4 = (-A_1 + A_3)(-B_3 + B_4)$ $P_5 = -A_4(-B_1 + B_2 - B_3 + B_4)$ $P_6 = (-A_1 - A_2 + A_3 + A_4)(-B_4)$ $P_7 = (A_1 - A_3 - A_4)(-B_1 - B_3 + B_4)$ $C_1 = -P_3 + P_6$ $C_2 = P_2 + P_3 - P_4 - P_5$ $C_3 = -P_1 - P_3 + P_5 + P_7$ $C_4 = P_3 - P_4 - P_5 - P_7$
Group 7: 4455666	Group 8: 3333888	Group 9: 2233666
$P_1 = (A_3 - A_4)(B_1 - B_3)$ $P_2 = (A_1 + A_2)(B_1 + B_3)$ $P_3 = -A_4(-B_1 - B_2 + B_3 + B_4)$ $P_4 = A_2(B_1 - B_2 + B_3 - B_4)$ $P_5 = (A_1 - A_2 + A_3 - A_4)(-B_1 + B_4)$ $P_6 = (A_1 + A_2 - A_3 - A_4)(-B_3 - B_4)$ $P_7 = (-A_1 + A_2 + A_3 - A_4)(-B_3 + B_4)$ $C_1 = 0.5P_1 - P_3 + 0.5P_4 + 0.5P_6 - 0.5P_7$ $C_2 = -0.5P_1 - 0.5P_4 + 0.5P_6 + 0.5P_7$ $C_3 = 0.5P_1 + P_2 - 0.5P_4 + P_5 + P_6$ $C_4 = -0.5P_1 - 0.5P_3 + 0.5P_5 + 0.5P_6$	$P_1 = (-A_4)(B_1 + B_2)$ $P_2 = (-A_2)(B_3 - B_4)$ $P_3 = (-A_3 + A_4)(B_1)$ $P_4 = (A_1 + A_2)(-B_3)$ $P_5 = (A_1 + A_2 - A_3 - A_4)(-B_1 - B_2 - B_3 - B_4)$ $P_6 = (-A_1 - A_2 + A_3 - A_4)(AB_1 - B_2 + B_3 - B_4)$ $P_7 = (-A_1 + A_2 + A_3 - A_4)(-B_1 + B_2 - B_3 + B_4)$ $C_1 = 0.5P_2 - P_4 + 0.5P_5 - P_6$ $C_2 = P_4 - P_7$ $C_3 = -P_3 - P_6$ $C_4 = 0.5P_1 + 0.5P_2 + P_3 - P_7$	$P_1 = (A_4)(-B_4)$ $P_2 = (-A_3)(B_3)$ $P_3 = (A_1 + A_2)(B_2)$ $P_4 = (-A_1)(-B_1 + B_2)$ $P_5 = (-A_2 - A_4)(-B_1 + B_2 + B_3 - B_4)$ $P_6 = (A_1 + A_2 + A_4)(-B_1 + B_2 + B_3)$ $P_7 = (A_1 + A_2 + A_3 + A_4)(-B_1 + B_3)$ $C_1 = P_3 + P_5$ $C_2 = P_1 + P_2 + P_4 + P_5$ $C_3 = -P_3 + P_4 - P_6 - P_7$ $C_4 = -P_2 - P_6$
	Group 10: 4446666	
	$P_1 = (A_1 + A_2)(B_1 - B_4)$ $P_2 = (-A_1 + A_2)(B_1 - B_2)$ $P_3 = (-A_1 - A_2)(-B_1 - B_2)$ $P_4 = (A_1 - A_3)(-B_1 - B_2 + B_3 + B_4)$ $P_5 = (-A_1 + A_2 + A_3 - A_4)(-B_2 + B_4)$ $P_6 = (-A_1 - A_2 - A_3 - A_4)(-B_2 - B_4)$ $P_7 = (-A_1 - A_3)(B_1 - B_2 - B_3 - B_4)$ $C_1 = -0.5P_5 + 0.5P_6$ $C_2 = -P_1 + 0.5P_2 + 0.5P_5 + 0.5P_6 - 0.5P_7$ $C_3 = P_1 + 0.5P_3 + 0.5P_4 - 0.5P_5 - 0.5P_6$ $C_4 = -0.5P_2 - 0.5P_3 + 0.5P_4 + 0.5P_5 - 0.5P_6 - P_7$	

TABLE II: The characteristics and number of solutions found by proposed method

Group Index	Characteristic of P_i's	Characteristic of C_i's	# of nonzero elements	# of solutions	# of distincts
Group 1	3333455	2444	40	52095	129
Group 2	2245667	2455	48	21537	136
Group 3	3344556	4455	48	21135	81
Group 4	3333444	2244	36	13192	39
Group 5	2266668	2556	54	4109	40
Group 6	2244556	2444	42	16869	71
Group 7	4455666	4455	54	12205	68
Group 8	3333888	2244	48	2638	34
Group 9	2233666	2244	40	8718	66
Group 10	4446666	2556	54	8598	37
Total solutions				**161,096**	**701**

matrix multiplication problem, duplicate solutions based on the permutation and changing the sign of coefficients don't provide new solutions. Hence, we report both type of results: all found solutions and distinct ones. Table II represents the number of distinct solutions in each group. The total number of distinct solutions are 701 which to the best of our knowledge, none of related studies could find this number of solutions. The maximum and minimum number of distinct solutions in defined groups are 34 and 136, respectively.

In order to investigate the probable size of the search space and the power of optimization algorithm, we calculate all possible solutions that can be constructed using found 701 distinct solutions. Since there are 7! permutations for different orders of P_i's, the total number of solutions by considering all possible combinations would be $701 \times 7! = 3,533,040$. In addition, if we change the sign of any parentheses in a P_i, we get a new solution while changing the sign of corresponding P_i in C_i's doesn't affect the validity of the solution. Each P_i consists of two parentheses, therefore by considering seven P_i there are $2^{(7\times2)} = 16,384$ different combinations of assigning signs to parentheses. However, this modification doesn't affect the validity of the solution, it finds an optimal solution in the search space. The total number of solutions by considering all such combinations is $701 \times 16,384 = 11,485,184$ which the proposed method could find $161,096$ of them. By considering the symmetricity property to change the signs and the order of P_i's, totally $701 \times 7! \times 2^{(7\times2)} = 57,885,327,360$ solutions can be generated by 701 distinct solutions. This number indicates the possible minimum existing solutions for 2×2 matrix multiplication. However, this is an indicative that a highly multimodal optimization problem is tackled, comparing to the size of the search space, the discovered solutions set is a very small portion of the search space equal to $57,885,327,360/(3^{56}) = 1.106\text{e}{-16}$. This indicates how the defined problem is a complex optimization problem which is large-scale and highly multimodal one.

Population size is one of the important parameters that should be set in evolutionary algorithms. Because the matrix multiplication is a large-scale optimization problem, we expect to increase the speed of finding a solution with a small population size, specially when the algorithm apply a restart strategy. Table III shows the average time of finding a solution with different size of population. As seen in the table, the time decreases when the size of population reduces. On the other hand, since matrix multiplication is a multimodal optimization problem with numerous solutions, decreasing the population size with restart strategy decreases the processing time of one iteration while the algorithm most likely finds a solution during each iteration. As a result, small population size leads more efficient search process.

TABLE III: Average time to find a solution using different population size. Times are in seconds.

Population size	5	10	30	40	150
Average time	57.2	62.4	97.2	107.1	161.9

IV. Conclusion remarks

In this paper, a Micro multimodal Genetic Algorithm is proposed to discover solutions equivalent to Strassen equations for matrix multiplication. A large-scale highly multimodal optimization problem is modeled to be solved using GA. The algorithm is tailored for this problem by proposing a local search method and two strategies for restarting the search to prevent premature convergence. Also, the population initialization was not uniform random; it was biased toward the sparsity of candidate solutions. In fact, despite of the simplicity of the algorithm, comparing to previous studies, it is able to find more solutions for this problem. Therefore, by proposing some simple but effective strategies, a multimodal optimization problem can solve efficiently. The algorithm could find 701 distinct solutions (as the first time) among $160k$ discovered valid solutions for matrix multiplication of size 2×2. We realized that using 701 unique solutions, more than 57 billion solutions can be generated which are a small portion of very large-scale exhaustive search possibilities. . Two studies can be conducted as the future work in this regard. Discovering the solutions for 3×3 matrices is a more challenging task that is the target of our next study; the goal would be discovering equations with 21 multiplications. In addition, parallelization of the proposed method provides more benefits to utilize computational power to discover more solutions in a very shorter time.

References

[1] V. Strassen, "Gaussian elimination is not optimal," *Numerische mathematik*, vol. 13, no. 4, pp. 354–356, 1969.

[2] J. F. Kolen and P. Bruce, "Evolutionary search for matrix multiplication algorithms." in *FLAIRS conference*, 2001, pp. 161–165.

[3] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," 1988.

[4] S. Oh and B.-R. Moon, "Automatic reproduction of a genius algorithm: Strassen's algorithm revisited by genetic search," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 246–251, 2009.

[5] Y. Zhou, X. Lai, Y. Li, and W. Dong, "Ant colony optimization with combining gaussian eliminations for matrix multiplication," *IEEE transactions on cybernetics*, vol. 43, no. 1, pp. 347–357, 2012.

[6] V. Arrigoni and A. Massini, "Fast strassen-based A^tA parallel multiplication," *arXiv preprint arXiv:1902.02104*, 2019.

[7] P. Ramanan, "Six pass mapreduce implementation of strassen's algorithm for matrix multiplication," in *Proceedings of the 5th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond*. ACM, 2018, p. 7.

[8] R. Ciucanu, M. Giraud, P. Lafourcade, and L. Ye, "Secure strassen-winograd matrix multiplication with mapreduce," 2019.

[9] A. Joo, A. Ekart, and J. P. Neirotti, "Genetic algorithms for discovery of matrix multiplication methods," *IEEE transactions on evolutionary computation*, vol. 16, no. 5, pp. 749–751, 2012.

[10] E. Morancho, "A vector implementation of gaussian elimination over gf (2): Exploring the design-space of strassen's algorithm as a case study," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2015, pp. 111–118.

[11] M. Mucha and P. Sankowski, "Maximum matchings via gaussian elimination," in *45th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2004, pp. 248–255.

[12] V. Elser, "A network that learns strassen multiplication," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3964–3976, 2016.

[13] M. A. Ghorbani, V. P. Singh, B. Sivakumar, M. H. Kashani, A. A. Atre, and H. Asadi, "Probability distribution functions for unit hydrographs with optimization using genetic algorithm," *Applied Water Science*, vol. 7, no. 2, pp. 663–676, 2017.

[14] C. Banks, "Searching for lyapunov functions using genetic programming," *Virginia Polytech Institute, unpublished*, 2002.

[15] A. Pourrajabian, R. Ebrahimi, M. Mirzaei, and M. Shams, "Applying genetic algorithms for solving nonlinear algebraic equations," *Applied Mathematics and Computation*, vol. 219, no. 24, pp. 11 483–11 494, 2013.

[16] Z. Michalewicz, C. Z. Janikow, and J. B. Krawczyk, "A modified genetic algorithm for optimal control problems," *Computers & Mathematics with Applications*, vol. 23, no. 12, pp. 83–94, 1992.

[17] H. Jia, A. Y. Nee, J. Y. Fuh, and Y. Zhang, "A modified genetic algorithm for distributed scheduling problems," *Journal of Intelligent Manufacturing*, vol. 14, no. 3-4, pp. 351–362, 2003.

[18] R. K. Hasda, R. K. Bhattacharjya, and F. Bennis, "Modified genetic algorithms for solving facility layout problems," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 11, no. 3, pp. 713–725, 2017.

[19] K. Krishnakumar, "Micro-genetic algorithms for stationary and non-stationary function optimization," in *Intelligent control and adaptive systems*, vol. 1196. International Society for Optics and Photonics, 1990, pp. 289–296.

[20] J. L. Bentley, D. Haken, and J. B. Saxe, "A general method for solving divide-and-conquer recurrences," *ACM SIGACT News*, vol. 12, no. 3, pp. 36–44, 1980.

[21] S. Winograd, "On multiplication of $2\times$ 2 matrices," *Linear algebra and its applications*, vol. 4, no. 4, pp. 381–388, 1971.