Applied Mathematics and Computational Intelligence
Volume 14, No. 1, 2025 [37 − 56]

A Scalar Modification of Three-term PRP-DL Conjugate Gradient Method for Solving Large-scaled Unconstrained Optimization Problems

Muhammad Aqiil Iqmal Ishak[1], Nurin Athirah Azmi[2*], Siti Mahani Marjugi[3]

[1,2*,3]Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia

Received: 23 August 2024
Revised: 1 September 2024
Accepted: 18 October 2024

ABSTRACT

Unconstrained optimization problems arise in numerous fields. This study presents the introduction of a hybrid Polak-Ribi'ere-Polyak(PRP)-Dai-Liao(DL) conjugate gradient(CG) method with a modified scalar for the purpose of solving large-scaled unconstrained optimization problems. The proposed method involves the modification of the scalar in the PRP-DL conjugate gradient method in order to improve the performance of the algorithm, specifically when addressing large-scale problems. The convergence analysis of the proposed method is established and proved under the strong Wolfe-Powell line search. Numerical results on various test functions show that the proposed method is more efficient and robust than several existing CG methods. Overall, the proposed method is a new promising CG method for solving unconstrained optimization problems.

Keywords: global convergence, large-scale unconstrained problems, line search, modified hybrid conjugate gradient method, test functions.

1 INTRODUCTION

Unconstrained optimization problem covers a diverse range of problem types that arise in various science and engineering areas in order to find the minimum value of certain function.

In general, the problem of unconstrained optimization can be formulated as

$$\min_{x \in R^n} f(x),$$

where $f : R^n \to R$ is continuously differentiable smooth and bounded.

Various approaches, including the conjugate gradient method can be utilized to address the previously mentioned problems. The CG method is widely recognized as an iterative technique that is highly regarded for its computational efficiency and ability to handle a wide range of problem conditions. This method was developed by Eduard Stiefel and Magnus [1]. Due to its efficient use of memory and ability to converge quickly, the CG method has found applications in many fields of research [2].

CG method involves refining an initial approximation with each iteration, resulting in the formation of a sequence denoted as x_k of the form

$$x_{k+1} = x_k + \alpha_k d_k, k = 0, 1, 2, \dots \tag{1}$$

where x_k is the k^{th} iterative point and d_k is the search direction and $\alpha_k > 0$ is the step size. Step size is determined via a one-dimensional search known as line search. Developed as an extension of the well-known Wolfe-Powell line search, the Strong Wolfe-Powell variant introduces additional conditions to ensure a more robust and reliable convergence towards the optimal solution. It is presented in the form of

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{2}$$

$$|g(x_k + \alpha_k d_k)^T d_k| \geq \sigma |g_k^T d_k|, \tag{3}$$

where scalar g_k is the derivative of $f(x)$ at the point of x_k, g_k^T is the transpose of g_k and δ is a very small positive value, $0 < \delta < \sigma < 1$.

A major criterion for a line search algorithm in the convergence analysis is that the search direction d_k has to satisfy the sufficient descent property, which is defined in the form:

$$g_k^T d_k \leq -c||g_k||^2, \tag{4}$$

where $c > 0$ is a constant.

The search direction, d_k is determined by:

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -g_k + ß_k d_{k-1}, & \text{if } k \geq 1. \end{cases} \tag{5}$$

where $ß_k$ is known as the CG coefficient.

There are several formulas available for $ß_k$, such as the Fletcher-Reeves (FR) by [3], Polak-Ribière-Polyak (PRP) by [4], Hestenes-Stiefel (HS) by [5], Liu-Storey by [6] (LS), Dai-Yuan (DY) by [7], and Conjugate Descent (CD) by [8] as stated below:

$$ß_k^{\text{HS}} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \quad ß_k^{\text{PRP}} = \frac{g_k^T y_{k-1}}{g_{k-1}^T g_{k-1}}, \quad ß_k^{\text{LS}} = \frac{g_k^T y_{k-1}}{-d_{k-1}^T g_{k-1}}$$

$$ß_k^{\text{DY}} = \frac{g_k^T g_k}{d_{k-1}^T y_{k-1}}, \quad ß_k^{\text{FR}} = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}, \quad ß_k^{\text{CD}} = \frac{g_k^T g_k}{-d_{k-1}^T g_{k-1}}$$

where $y_{k-1} = g_k - g_{k-1}$.

Hybrid conjugate gradient method is a method that combines several standard CG methods in order to exploit the attractive features of each of them. It is often said that the hybrid CG method is more efficient and more robust than the standard CG methods. Born out of the need to address both the computational efficiency and memory constraints posed by large-scale optimization challenges, this hybrid method provides a balanced approach that seeks to accelerate convergence and minimize memory requirements.

Three-term CG method is an extension of the CG method that introduces a novel recursion formula involving three terms. The three-term recursion formula in the method introduces a new level of sophistication to the optimization process, allowing for enhanced convergence speed and efficiency compared to two-term methods. While the three-term method may require more complex calculations in each iteration compared to the standard CG method, its potential for faster convergence and improved performance on certain types of optimization problems makes it a valuable addition to the toolkit of optimization practitioners.

According to [9], numerical performance of algorithm of new hybrid three-term CG method with modified secant condition(HTTCGSC) was presented and compared with other methods such as modified three-term Hestenes–Stiefel (MTTHS) by [10]. It is shown that HTTCGSC outperform by solving the test problems with the least iteration number, least number of computing functions and gradients and least CPU time consumed. [11] studied a hybrid CG method(HCGM) that combines PRP method with FR method. HCGM is said to has the sufficient descent property under the suggestion of a suitable line search and appropriate conditions. MZZ is the method suggest by [12] in which it guarantees the sufficient descent condition. Subsequently, the new modified hybrid three-term(MTT) CG method proposed in this study will also be compared with MTTHS, MZZ, HTTCGSC and HCGM in the unconstrained optimization problem.

This study aims to address these challenges by proposing a new modified hybrid PRP-DL conjugate gradient optimization algorithm tailored specifically for unconstrained optimization problems. The three-term conjugate gradient method is chosen as the foundation due to its efficiency and widespread use in optimization tasks. Following the inspiration provided by [9], some modifications to the scalar of the presented method have been made by adopting an idea from the research conducted by [11]. The modification involves incorporating novel strategies and inquiries to enhance the convergence speed, stability, and ability of the algorithm.

This paper is organised as follows. In section 2, the underlying idea of modification of the algorithm will be presented. In section 3, sufficient descent property and global convergence property with the strong Wolfe-Powell line search will be established. For section 4, a numerical result of the proposed method will be discussed. Finally, the conclusions will be highlighted in section 5.

2 MODIFICATIONS

The primary sources of our motivation are the works of [9] and [11], wherein

$$ß_k^{\mathrm{N}} = \frac{g_k^T(y_{k-1} - ts_{k-1})}{\max\{y_{k-1}^T s_{k-1}, ||g_{k-1}||^2\}}, \tag{6}$$

$$\delta_k^{\mathrm{N}} = \frac{g_k^T s_{k-1}}{\max\{y_{k-1}^T s_{k-1}, ||g_{k-1}||^2\}}. \tag{7}$$

The parameter t is defined as $t = \max\{\bar{t}, \frac{||y_k||^2 g_{k-1}^T s_k}{z_k}\}$ Some modification on (6) and (7) have been made. The parameter $s_{k-1}^T g_{k-1}$ has been inserted in both numerator and denominator of (6) and (7) while referring on the study of [11]. The new CG coefficient is defined as below,

$$ß_k^{\mathrm{MTT}} = \frac{g_k^T(s_{k-1}^T y_{k-1} g_{k-1} - t||s_{k-1}||^2 g_{k-1})}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}}, \tag{8}$$

$$\delta_k^{\mathrm{MTT}} = \frac{||s_{k-1}||^2 g_k^T g_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}}. \tag{9}$$

According to [13], $\bar{t} = 0.1$ is an appropriate choice. Therefore, $t = \max\{0.1, \frac{||y_k||^2 g_{k-1}^T s_k}{z_k}\}$ is assigned.

Algorithm 1: Modified three-term(MTT) PRP-DL conjugate gradient method

Step 1: Let $k = 0$. Choose a starting point $x_0 \in R^n$. Obtain $g(x_0)$ and assign $d_0 = -g_0$.
Step 2: If $||g_k|| \leq \varepsilon, \ \varepsilon = 10^{-6}$, then stop, otherwise proceed to the next step.
Step 3: Determine the step size α_k along the direction d_k by using the Strong Wolfe-Powell line search stated in (2) and (3).
Step 4: Let $x_{k+1} = x_k + \alpha_k d_k$ to compute the new iterative point.
Step 5: Calculate the search direction d_k by using:

$$d_k^{\mathrm{MTT}} = \begin{cases} -g_k, & \text{if } k = 0, \\ -g_k + ß_k^{\mathrm{MTT}} s_{k-1} - \delta_k^{\mathrm{MTT}} y_{k-1}, & \text{if } k \geq 1, \end{cases}$$

where $s_{k-1} = \alpha_{k-1} d_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$.

Step 6: Set $k = k + 1$ and repeat Step 1.

3 CONVERGENCE ANALYSIS

In the upcoming discussion, clarification will be established that Algorithm 1 possesses a sufficient descent property, regardless of the line search technique employed.

Lemma 1. *Algorithm 1 generated the sequence $\{d_k^{MTT}\}$ independent on any line search, and it always holds that:*

$$g_k^T d_k^{\mathrm{MTT}} \leq -||g_k||^2, \forall k \geq 0. \tag{10}$$

Proof. When $k = 0$, then $d_0 = -g_0$, and it holds that $g_0^T = -||g_0||^2$. For $k \geq 1$, the subsequent inequality obtained according to the definition of d_k^{MTT}:

$$\begin{aligned}
g_k^T d_k^{\text{MTT}} &= -||g_k||^2 + ß_k^{\text{MTT}} g_k^T s_{k-1} - \delta_k^{\text{MTT}} d_k^T y_{k-1} \\
&= -||g_k||^2 + \frac{g_k^T (s_{k-1}^T y_{k-1} g_{k-1} - t||s_{k-1}||^2 g_{k-1})}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} g_k^T s_{k-1} \\
&- \frac{||s_{k-1}||^2 g_k^T g_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} g_k^T y_{k-1} \\
&= -||g_k||^2 + \frac{g_k^T ||s_{k-1}||^2 y_{k-1} g_{k-1} g_k^T}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} \\
&- \frac{t||s_{k-1}||^2 g_{k-1} g_k^T s_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} \\
&- \frac{||s_{k-1}||^2 g_k^T g_{k-1} g_k^T y_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} \\
&= -||g_k||^2 - \frac{t||s_{k-1}||^2 g_{k-1} g_k^T s_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} \\
&\le -||g_k||^2,
\end{aligned}$$

where the last inequality holds when $t \ge 0$. Then, equation (10) holds. This completes the proof. □

Lemma (1) indicates that, regardless of the line search method, the new direction satisfies the sufficient descent property. On top of that, a conjugate condition plays an essential role to numerical performance. For MTT, by the design of the direction d_k^{MTT},

$$\begin{aligned}
(d_k^{\text{MTT}})^T y_{k-1} &= -g_k^T y_{k-1} + \frac{g_k^T (s_{k-1}^T y_{k-1} g_{k-1} - t||s_{k-1}||^2 g_{k-1})}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} y_{k-1}^T s_{k-1} \\
&- \frac{||s_{k-1}||^2 g_k^T g_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} ||y_{k-1}||^2 \\
&= -g_k^T y_{k-1} + \frac{||s_{k-1}||^2 y_{k-1} g_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} g_k^T y_{k-1}^T \\
&- \frac{y_{k-1}^T (t||s_{k-1}||^2 g_{k-1})}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} g_k^T s_{k-1} \\
&- \frac{||y_{k-1}||^2 (s_{k-1} g_{k-1})}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} g_k^T s_{k-1} \\
&\le -\frac{t y_{k-1}^T ||s_{k-1}||^2 g_{k-1} + ||y_{k-1}||^2 s_{k-1} g_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} g_k^T s_{k-1}.
\end{aligned} \tag{11}$$

From (11), it holds that the new direction d_k^{MTT} satisfies DL conjugate condition, in an extent form in which $(d_k^{\text{MTT}})^T y_{k-1} \le -t_1 g_k^T s_{k-1}$ where

$t_1 = -\frac{t y_{k-1}^T ||s_{k-1}||^2 g_{k-1} + ||y_{k-1}||^2 s_{k-1} g_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}}$. In fact, if we adopt the line search technique which results in $||s_{k-1}||^2 g_{k-1}^T y_{k-1} \geq 0$, then it holds that $t_1 = -- \frac{t y_{k-1}^T ||s_{k-1}||^2 g_{k-1} + ||y_{k-1}||^2 s_{k-1} g_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} > 0$.

In this part, the convergence characteristics of the $ß_k^{\mathrm{MTT}}$ will be examined and investigated. Assume that for all values of k, $g_k \neq 0$. If g_k is equal to zero, it indicates the presence of a stationary point. The convergence of nonlinear conjugate gradient algorithms is frequently demonstrated based on the subsequent assumptions.

Assumption 1. *The level set $T := \{x \in R^n : f(x) \leq f(x_0)\}$ is bounded where x_0 is the initial point, then it means there exist a constant $X > 0$ in such a way that:*

$$||x|| \leq X, \quad \forall x \in T. \tag{12}$$

Assumption 2. *In some neighborhood N of T, the gradient of function $f(x)$ and $g(x)$ known as Lipschitz continuous, which means there exists a constant $L > 0$ such that:*

$$||g(x) - g(y)|| \leq L||x - y||, \quad \forall x, y \in N. \tag{13}$$

It should be noted that, according to Assumption 1 and Assumption 2, there is a positive constant G that satisfies the following condition:

$$||g(x)|| \leq G, \quad \forall x \in T. \tag{14}$$

In the following analysis, the sequence d_k^{MTT} produced by Algorithm 1 is bounded will be demonstrated.

Lemma 2. *Consider the condition $0 < t \leq T$, and assume that both Assumption 1 and Assumption 2 are satisfied. For any line search technique, consider the sequence $\{d_k^{MTT}\}$ generated by Algorithm 1. If the objective function f exhibits uniform convexity on the set T, it can be concluded that $||d_k^{MTT}||$ is bounded.*

Proof. Given that the function f exhibits uniform convexity on the set N, it follows that for any x, $y \in N$, the following inequality holds:

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq \tilde{u}||x - y||^2,$$

where $\tilde{u} > 0$ is the uniform convexity parameter. In particular, when assigning $x = x_k$ and $y = x_{k-1}$, the following equation is true:

$$||s_{k-1}||^2 g_{k-1}^T y_{k-1} \geq \tilde{u}||s_{k-1}||^2 > 0.$$

In the subsequent analysis, the boundedness of the parameters $ß_k^{\mathrm{MTT}}$ and δ_k^{MTT} will be proved.

According to their respective definitions, observe that:

$$
\begin{aligned}
|ß_k^{\mathrm{MTT}}| &= \left| \frac{g_k^T(s_{k-1}^T y_{k-1} g_{k-1} - t||s_{k-1}||^2 g_{k-1})}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} \right| \\
&\le \frac{||g_k||(||s_{k-1}^T|| ||y_{k-1}|| ||g_{k-1}|| + t||s_{k-1}||^2 ||g_{k-1}||)}{|\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}|} \\
&\le \frac{||g_k||(||s_{k-1}^T|| ||y_{k-1}|| ||g_{k-1}|| + t||s_{k-1}||^2 ||g_{k-1}||)}{||s_{k-1}||^2 g_{k-1}^T y_{k-1}} \\
&\le \frac{(L+T)||s_{k-1}||}{\tilde{u}||s_{k-1}||^2} ||g_k|| \\
&= \frac{(L+T)}{\tilde{u}} \frac{||g_k||}{||s_{k-1}||}.
\end{aligned}
$$

$$
\begin{aligned}
|\delta_k^{\mathrm{MTT}}| &= \left| \frac{||s_{k-1}||^2 g_k^T g_{k-1}}{\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}} \right| \\
&\le \frac{||s_{k-1}||^2 ||g_k|| ||g_{k-1}||}{|\max\{||s_{k-1}||^2 g_{k-1}^T y_{k-1}, ||g_{k-1}||^2 g_{k-1}^T s_{k-1}\}|} \\
&\le \frac{||s_{k-1}||^2 ||g_k|| ||g_{k-1}||}{||s_{k-1}||^2 g_{k-1}^T y_{k-1}} \\
&\le \frac{||s_{k-1}||^2 ||g_k|| ||g_{k-1}||}{\tilde{u}||s_{k-1}||^2} \\
&= \frac{1}{\tilde{u}} ||g_k|| ||g_{k-1}||.
\end{aligned}
$$

Hence, according to the definition of d_k^{MTT} :

$$
\begin{aligned}
||d_k^{\mathrm{MTT}}|| &= \left|\left| -g_k + ß_k^{\mathrm{MTT}} s_{k-1} + \delta_k^{\mathrm{MTT}} y_{k-1} \right|\right| \\
&\le ||g_k|| + |ß_k^{\mathrm{MTT}}| ||s_{k-1}|| + |\delta_k^{\mathrm{MTT}}| ||y_{k-1}|| \\
&\le ||g_k|| + \frac{(L+T)}{\tilde{u}} \frac{||g_k||}{||s_{k-1}||} ||s_{k-1}|| + \frac{1}{\tilde{u}} ||g_k|| ||g_{k-1}|| ||y_{k-1}|| \\
&\le ||g_k|| + \frac{(L+T)}{\tilde{u}} ||g_k|| + \frac{L}{\tilde{u}} ||g_k|| \\
&= \left(1 + \frac{2L+T}{\tilde{u}} + \frac{L}{\tilde{u}}\right) ||g_k|| \\
&\le \left(1 + \frac{2L+T}{\tilde{u}} + \frac{L}{\tilde{u}}\right) G.
\end{aligned}
$$

where the last inequality is satisfied by (14). Then, this completes the proof. □

The subsequent Lemma presented serves as a crucial role in the global convergence theorem of the proposed method.

Lemma 3. *Suppose that Assumption 1 and Assumption 2 are satisfied. Consider iterative method represented by equation 1, where d_k fulfils the sufficient descent condition and α_k is established using the strong Wolfe-Powell line search stated in (2) and (3). According to [9], if the aforementioned relationship holds:*

$$\sum_{k\geq 0} \frac{1}{||d_k||^2} = +\infty, \tag{15}$$

then, the method exhibits global convergence as such:

$$\lim_{k\to+\infty} \inf ||g_k|| = 0. \tag{16}$$

A proof that Algorithm 1 is globally converge for uniformly convex objective functions will be presented in the next discussion.

Theorem 3.1. *Suppose that Assumption 1 and Assumption 2 are satisfied wherein α_k is established using the strong Wolfe-Powell line search stated in (2) and (3). If the objective function f exhibits uniform convexity on the set N, it can be concluded that Algorithm 1 achieves global convergence in a way that:*

$$\lim_{k\to+\infty} ||g_k|| = 0. \tag{17}$$

Proof. According to Lemma 1, it can be concluded that the direction d_k^{MTT} exhibits the sufficient descent property with a constant value of $c = 1$. According to the inequality stated in equation (2), it can be observed that the sequence $\{f(x_k)\}_{k\geq 0}$ is monotonically decreasing, and $\{x_k\}_{k\geq 0}$ belongs to the set of natural numbers, N. The validity of equation (15) can be established by utilizing the boundedness property of d_k^{MTT} as stated in Lemma 2. Subsequently, equation (16) satisfies. As f is uniformly convex, (17) holds. The proof is now complete. □

4 RESULTS AND DISCUSSIONS

This section focuses on the numerical performance of Algorithm 1 and its comparison with the HTTCGSC by [9], MZZ by [12], MTTHS by [14] and HCGM by [11]. The parameters for each of the aforementioned methods are taken into account and being used in this study.

The tests have been conducted on a Personal Computer DELL (Intel Core i5-6440HQ CPU @ 2.60GHz, with 8.00 GB RAM, Windows 10). All the problems listed in Appendix (Table 3) have been resolved using MATLAB R2023a. The parameters used are $\delta = 0.0001$ and $\sigma = 0.009$.

The numerical results are evaluated by comparing the number of iterations(NOI) and computational(CPU) time. The testing terminated if either the total number of iterations exceeds 10,000 or CPU times took longer than 120 seconds.

138 test problems with various initial points and dimensions are considered in this study. The numerical results of Algorithm 1 along with other compared existing methods are shown in Table 1 and Table 2. The numerical performances are depicted in Figure 1 and Figure 2, correspondingly, utilizing the performance profile method developed by [15].

Table 1 : Numerical results for MTT with HTTCGSC and MZZ in terms of NOI and CPU Time

No	Functions	**MTT**		HTTCGSC		MZZ	
		NOI	CPU	NOI	CPU	NOI	CPU
1	Extended White Holst	9	1.4627	-	-	20	2.9973
2	Extended White Holst	9	2.2906	-	-	16	3.7468
3	Extended White Holst	10	46.1012	-	-	13	49.6204
4	Extended Rosenbrock	14	0.6426	-	-	21	0.7261
5	Extended Rosenbrock	14	1.352	-	-	21	1.553
6	Extended Rosenbrock	14	23.5483	-	-	20	22.0262
7	Extended Freudenstein Roth	10	0.0507	-	-	17	0.0356
8	Extended Freudenstein Roth	-	-	-	-	-	-
9	Extended Freudenstein Roth	10	2.3928	-	-	-	-
10	Extended Beale	10	0.0445	995	1.7941	13	0.0451
11	Extended Beale	10	1.2099	-	-	14	1.3205
12	Extended Beale	10	2.6315	-	-	15	2.8063
13	Raydan 1	24	0.0068	18	0.0058	20	0.0064
14	Raydan 1	48	0.0053	56	0.0062	58	0.0071
15	Raydan 1	68	0.0211	295	0.0475	68	0.0172
16	Extended Tridiagonal 1	11	0.0093	23	0.0078	7	0.0085
17	Extended Tridiagonal 1	11	0.0031	21	0.0045	7	0.0055
18	Extended Tridiagonal 1	11	0.0053	24	0.0074	9	0.0046
19	Diagonal 4	2	0.0074	465	0.3049	3	0.0085
20	Diagonal 4	2	0.0374	479	1.2495	3	0.0263
21	Diagonal 4	2	0.0603	192	6.2568	3	0.088
22	Extended Himmelblau	7	0.0171	28	0.0338	11	0.0204
23	Extended Himmelblau	7	0.2593	29	0.7804	12	0.4062
24	Extended Himmelblau	7	0.5279	30	1.687	11	0.7477
25	FLETCHCR	33	0.016	119	0.0286	44	0.0181
26	FLETCHCR	40	0.1885	154	0.4831	38	0.1436
27	FLETCHCR	40	1.4865	147	5.4633	37	1.2617
28	NONSCOMP	11	0.0211	676	0.0251	20	0.0126
29	NONSCOMP	271	0.017	-	-	231	0.0144
30	Extended DENSCHNB	5	0.0135	9	0.0196	7	0.0144
31	Extended DENSCHNB	5	0.1668	10	0.2764	7	0.1976
32	Extended DENSCHNB	5	0.335	10	0.5641	7	0.402
33	Extended Penalty Function U52	10	0.0102	73	0.0102	11	0.0083
34	Extended Penalty Function U52	11	0.0007021	40	0.0086	15	0.000817
35	Extended Penalty Function U52	17	0.0032	382	0.0311	10	0.0047
36	Hager	9	0.0055	11	0.0052	9	0.0054
37	Hager	12	0.0005922	11	0.0011	13	0.000548
38	Hager	19	0.0033	31	0.0036	20	0.0041
39	Cube	25	0.0151	-	-	12	0.2249
40	Extended Maratos	14	0.0084	5753	0.188	25	0.0091
41	Extended Maratos	14	0.0033	-	-	25	0.006

continued on next page

Table 1: Numerical results for MTT with HTTCGSC and MZZ in terms of NOI and CPU Time (Continued)

42	Extended Maratos	14	0.0044	3916	0.6855	21	0.0051
43	Six Hump Camel	5	0.0247	10	0.0054	8	0.0058
44	Six Hump Camel	8	0.011	10	0.0016	8	0.0011
45	Three Hump Camel	27	0.0155	-	-	-	-
46	Booth	2	0.0051	33	0.0062	3	0.0053
47	Booth	2	0.0001685	10	0.0003998	3	0.000214
48	Trecanni	1	0.0039	1	0.0036	1	0.0043
49	Trecanni	4	0.0011	9	0.0012	10	0.0011
50	Zettl	1	0.0046	2	0.0047	2	0.0057
51	Zettl	11	0.0011	108	0.0037	19	0.000687
52	Shallow	3	0.0113	18	0.0236	3	0.013
53	Shallow	4	0.1468	61	1.6884	3	0.0976
54	Shallow	4	0.2769	68	3.6071	3	0.1973
55	Generalized Quartic	9	0.0147	5	0.0082	-	-
56	Generalized Quartic	8	0.0844	9	0.0315	-	-
57	Generalized Quartic	9	0.2161	9	0.0635	18	0.1552
58	Quadratic QF2	26	0.0084	77	0.0093	28	0.0073
59	Quadratic QF2	102	0.0185	685	0.0862	102	0.0175
60	Quadratic QF2	342	0.2856	7382	9.028	305	0.2179
61	Leon	17	0.0055	-	-	39	0.0172
62	Leon	14	0.0008994	-	-	68	0.0044
63	Generalized Tridiagonal 1	20	0.0107	47	0.0103	22	0.0104
64	Generalized Tridiagonal 1	29	0.0023	59	0.0048	33	0.0026
65	Generalized Tridiagonal 1	36	0.0172	68	0.0208	34	0.0148
66	Generalized Tridiagonal 2	28	0.0101	65	0.0104	-	-
67	Generalized Tridiagonal 2	44	0.005	79	0.0063	-	-
68	Generalized Tridiagonal 2	46	0.0381	-	-	-	-
69	POWER	10	0.0052	785	0.0275	33	0.0062
70	POWER	66	0.0071	-	-	177	0.0153
71	POWER	773	0.3253	-	-	2249	0.8647
72	Quadratic QF1	56	0.0146	683	0.0872	63	0.0159
73	Quadratic QF1	187	0.1162	7056	9.3302	206	0.1651
74	Quadratic QF1	606	3.0397	-	-	665	4.2917
75	Extended Quadratic Penalty QP2	15	0.0012	-	-	19	0.0104
76	Extended Quadratic Penalty QP2	24	0.0183	-	-	37	0.0135
77	Extended Quadratic Penalty QP2	41	0.0832	-	-	53	0.0958
78	Extended Quadratic Penalty QP1	7	0.0094	60	0.0019	10	0.0096
79	Extended Quadratic Penalty QP1	7	0.0525	69	0.0115	10	0.05792
80	Extended Quadratic Penalty QP1	7	0.0026	-	-	11	0.0058
81	Quartic	544	0.0297	6496	0.1877	240	0.0166
82	Quartic	30	0.0015	3884	0.1083	72	0.0025
83	Matyas	1	0.0049	1	0.0062	1	0.0053
84	Matyas	1	0.000186	1	0.0001359	1	0.0001404
85	Colville	14	0.0088	4804	0.0974	35	0.0092

continued on next page

Table 1: Numerical results for MTT with HTTCGSC and MZZ in terms of NOI and CPU Time (Continued)

86	Colville	84	0.0026	8881	0.1864	74	0.0023
87	Dixon and Price	79	0.0644	-	-	90	0.0739
88	Dixon and Price	79	0.4333	-	-	90	0.4724
89	Dixon and Price	79	4.6986	-	-	90	4.5173
90	Sphere	1	0.0063	1	0.0068	1	0.0057
91	Sphere	1	0.0098	1	0.01	1	0.009
92	Sphere	1	0.067	1	0.0616	1	0.0625
93	Sum Squares	178	0.1577	6267	6.0348	178	0.1248
94	Sum Squares	578	2.8561	-	-	578	3.6725
95	Sum Squares	1310	30.7119	-	-	1310	47.4673
96	DENSCHNA	6	0.0205	30	0.053	8	0.0153
97	DENSCHNA	6	0.0918	34	0.443	8	0.1328
98	DENSCHNA	6	0.8686	38	6.6764	8	1.1302
99	DENSCHNB	8	0.0094	14	0.011	14	0.0106
100	DENSCHNB	8	0.0326	16	0.054	13	0.0483
101	DENSCHNB	8	0.2879	17	0.5244	12	0.3894
102	DENSCHNC	7	0.0131	125	0.0408	10	0.0132
103	DENSCHNC	7	0.0883	129	1.0174	11	0.1238
104	DENSCHNC	7	0.8096	134	14.8962	13	1.3674
105	DENSCHNF	11	0.0177	-	-	-	-
106	DENSCHNF	12	0.0762	-	-	-	-
107	DENSCHNF	13	0.7105	-	-	-	-
108	Extended Block-Diagonal BD1	6	0.0113	28	0.0139	8	0.0107
109	Extended Block-Diagonal BD1	7	0.0345	31	0.1044	8	0.0342
110	Extended Block-Diagonal BD1	7	0.2876	33	1.0896	8	0.2723
111	HIMMELBG	1	0.0109	2	0.0117	1	0.0095
112	HIMMELBG	1	0.0024	3	0.0019	1	0.0046
113	HIMMELBG	1	0.0031	3	0.0026	1	0.0012
114	HIMMELBH	4	0.0079	14	0.0067	4	0.0068
115	HIMMELBH	4	0.001	15	0.002	4	0.0028
116	HIMMELBH	4	0.0011	16	0.003	4	0.0012
117	Extended Hiebert	22	0.0437	-	-	43	0.0608
118	Extended Hiebert	22	0.2521	-	-	37	0.3296
119	Extended Hiebert	22	2.2975	-	-	46	5.0033
120	Linear Perturbed	54	0.0151	620	0.0805	54	0.0144
121	Linear Perturbed	406	1.0354	-	-	406	0.9988
122	Linear Perturbed	1310	45.8803	-	-	1310	48.7352
123	QUARTICM	42	0.2271	30	0.1585	35	0.1897
124	QUARTICM	109	51.2399	63	22.4039	87	26.093
125	Zirilli or Aluffi-Pentini's	4	0.0065	5	0.0112	4	0.0058
126	Zirilli or Aluffi-Pentini's	4	0.0002242	5	0.0002545	4	0.0001465
127	Extended Quadratic Penalty QP3	10	0.0115	15	0.0109	16	0.0101
128	Extended Quadratic Penalty QP3	10	0.0018	-	-	16	0.0016
129	Extended Quadratic Penalty QP3	15	0.0084	-	-	-	-

continued on next page

Table 1: Numerical results for MTT with HTTCGSC and MZZ in terms of NOI and CPU Time (Continued)

130	DIAG-AUP1	4	0.1897	15	0.124	26	0.1993
131	Strait	17	0.0282	790	0.5186	55	0.056
132	Strait	16	1.1067	809	59.9595	38	3.0691
133	Strait	24	16.2049	-	-	50	47.1011
134	Perturbed Quadratic	2	0.0093	10	0.0097	3	0.0077
135	Perturbed Quadratic	2	0.0001905	11	0.0023	4	0.0001688
136	Perturbed Quadratic	2	0.0001502	12	0.0013	4	0.0001942
137	Diagonal 2	9	0.0113	7	0.0105	7	0.0111
138	Diagonal 2	97	0.0063	18	0.0018	19	0.0195

Table 2 : Numerical results for MTT with MTTHS and HCGM in terms of NOI and CPU Time

No	Functions	**MTT**		MTTHS		HCGM	
		NOI	CPU	NOI	CPU	NOI	CPU
1	Extended White Holst	9	1.4627	18	2.1081	-	-
2	Extended White Holst	9	2.2906	13	3.9432	-	-
3	Extended White Holst	10	46.1012	17	32.0173	-	-
4	Extended Rosenbrock	14	0.6426	22	0.7138	5429	221.1279
5	Extended Rosenbrock	14	1.352	19	1.3355	-	-
6	Extended Rosenbrock	14	23.5483	24	28.3459	-	-
7	Extended Freudenstein Roth	10	0.0507	-	-	-	-
8	Extended Freudenstein Roth	-	-	34	1.7817	-	-
9	Extended Freudenstein Roth	10	2.3928	83	8.9313	-	-
10	Extended Beale	10	0.0445	12	0.0453	1089	2.1014
11	Extended Beale	10	1.2099	14	1.2874	-	-
12	Extended Beale	10	2.6315	16	3.9427	-	-
13	Raydan 1	24	0.0068	18	0.0066	55	0.0067
14	Raydan 1	48	0.0053	59	0.0057	283	0.0225
15	Raydan 1	68	0.0211	68	0.017	363	0.0481
16	Extended Tridiagonal 1	11	0.0093	8	0.0083	2133	0.0999
17	Extended Tridiagonal 1	11	0.0031	10	0.0032	-	-
18	Extended Tridiagonal 1	11	0.0053	14	0.0061	-	-
19	Diagonal 4	2	0.0074	3	0.0087	111	0.0809
20	Diagonal 4	2	0.0374	4	0.2801	257	0.659
21	Diagonal 4	2	0.0603	5	0.154	340	12.9754
22	Extended Himmelblau	7	0.0171	7	0.0159	30	0.0389
23	Extended Himmelblau	7	0.2593	8	0.2615	39	1.62
24	Extended Himmelblau	7	0.5279	8	0.5204	-	-
25	FLETCHCR	33	0.016	43	0.0175	-	-
26	FLETCHCR	40	0.1885	49	0.194	-	-
27	FLETCHCR	40	1.4865	36	1.3561	-	-
28	NONSCOMP	11	0.0211	13	0.0204	1974	0.0507

continued on next page

Table 2: Numerical results for MTT with MTTHS and HCGM in terms of NOI and CPU Time (Continued)

29	NONSCOMP	271	0.017	135	0.0198	-	-
30	Extended DENSCHNB	5	0.0135	6	0.0131	17	0.027
31	Extended DENSCHNB	5	0.1668	6	0.1726	-	-
32	Extended DENSCHNB	5	0.335	6	0.349	12	0.9354
33	Extended Penalty Function U52	10	0.0102	6	0.0084	79	0.01
34	Extended Penalty Function U52	11	0.0007021	13	0.0006407	109	0.0048
35	Extended Penalty Function U52	17	0.0032	10	0.0018	381	0.03
36	Hager	9	0.0055	9	0.0071	10	0.0051
37	Hager	12	0.0005922	13	0.0005144	20	0.0008025
38	Hager	19	0.0033	20	0.003	49	0.0049
39	Cube	25	0.0151	30	0.0226	-	-
40	Extended Maratos	14	0.0084	26	0.0079	209	0.0138
41	Extended Maratos	14	0.0033	16	0.0023	-	-
42	Extended Maratos	14	0.0044	20	0.0046	-	-
43	Six Hump Camel	5	0.0247	6	0.0058	10	0.0048
44	Six Hump Camel	8	0.011	7	0.0021	25	0.0036
45	Three Hump Camel	27	0.0155	6	0.0077	-	-
46	Booth	2	0.0051	3	0.005	21	0.0055
47	Booth	2	0.0001685	3	0.0001341	69	0.0013
48	Trecanni	1	0.0039	1	0.0039	1	0.0036
49	Trecanni	4	0.0011	7	0.0012	28	0.0015
50	Zettl	1	0.0046	2	0.0048	2	0.0045
51	Zettl	11	0.0011	17	0.0005669	176	0.0058
52	Shallow	3	0.0113	3	0.0037	11	0.0194
53	Shallow	4	0.1468	3	0.0983	9	0.2905
54	Shallow	4	0.2769	3	0.2086	7	0.4699
55	Generalized Quartic	9	0.0147	5	0.0089	5	0.0013
56	Generalized Quartic	8	0.0844	5	0.0197	6	0.0244
57	Generalized Quartic	9	0.2161	5	0.0409	7	0.052
58	Quadratic QF2	26	0.0084	28	0.0088	83	0.0089
59	Quadratic QF2	102	0.0185	87	0.015	651	0.0796
60	Quadratic QF2	342	0.2856	327	0.2348	6928	7.1401
61	Leon	17	0.0055	53	0.0121	-	-
62	Leon	14	0.0008994	59	0.0033	-	-
63	Generalized Tridiagonal 1	20	0.0107	21	0.0093	50	0.0026
64	Generalized Tridiagonal 1	29	0.0023	36	0.0026	70	0.0044
65	Generalized Tridiagonal 1	36	0.0172	-	-	75	0.0253
66	Generalized Tridiagonal 2	28	0.0101	28	0.009	127	0.0039
67	Generalized Tridiagonal 2	44	0.005	46	0.0048	211	0.018
68	Generalized Tridiagonal 2	46	0.0381	48	0.0231	-	-
69	POWER	10	0.0052	46	0.007	500	0.014
70	POWER	66	0.0071	266	0.0214	-	-
71	POWER	773	0.3253	3133	1.3455	-	-
72	Quadratic QF1	56	0.0146	63	0.0156	510	0.0699

continued on next page

Table 2: Numerical results for MTT with MTTHS and HCGM in terms of NOI and CPU Time (Continued)

73	Quadratic QF1	187	0.1162	211	0.1643	7049	8.5846
74	Quadratic QF1	606	3.0397	744	4.732	-	-
75	Extended Quadratic Penalty QP2	15	0.0012	20	0.0103	-	-
76	Extended Quadratic Penalty QP2	24	0.0183	30	0.0076	-	-
77	Extended Quadratic Penalty QP2	41	0.0832	97	0.1428	-	-
78	Extended Quadratic Penalty QP1	7	0.0094	9	0.0094	53	0.0096
79	Extended Quadratic Penalty QP1	7	0.000525	9	0.0004641	64	0.0025
80	Extended Quadratic Penalty QP1	7	0.0026	12	0.0033	-	-
81	Quartic	544	0.0297	87	0.0115	6351	0.1725
82	Quartic	30	0.0015	122	0.0045	3084	0.0811
83	Matyas	1	0.0049	1	0.0047	2	0.0045
84	Matyas	1	0.000186	1	0.0001429	2	0.0001414
85	Colville	14	0.0088	34	0.0096	1045	0.0308
86	Colville	84	0.0026	48	0.0014	-	-
87	Dixon and Price	79	0.0644	64	0.0552	-	-
88	Dixon and Price	79	0.4333	64	0.3669	-	-
89	Dixon and Price	79	4.6986	64	3.6279	-	-
90	Sphere	1	0.0063	1	0.0058	1	0.0055
91	Sphere	1	0.0098	1	0.0088	1	0.0093
92	Sphere	1	0.067	1	0.059	1	0.0602
93	Sum Squares	178	0.1577	179	0.1357	5588	5.7609
94	Sum Squares	578	2.8561	588	3.6287	-	-
95	Sum Squares	1310	30.7119	1363	49.4167	-	-
96	DENSCHNA	6	0.0205	7	0.0234	53	0.0874
97	DENSCHNA	6	0.0918	7	0.1119	54	0.7111
98	DENSCHNA	6	0.8686	8	1.1948	22	3.8829
99	DENSCHNB	8	0.0094	7	0.0099	18	0.0123
100	DENSCHNB	8	0.0326	9	0.0544	-	-
101	DENSCHNB	8	0.2879	9	0.3183	-	-
102	DENSCHNC	7	0.0131	8	0.0142	119	0.0421
103	DENSCHNC	7	0.0883	8	0.093	115	0.953
104	DENSCHNC	7	0.8096	10	0.9925	138	18.1472
105	DENSCHNF	11	0.0177	-	-	-	-
106	DENSCHNF	12	0.0762	-	-	-	-
107	DENSCHNF	13	0.7105	-	-	-	-
108	Extended Block-Diagonal BD1	6	0.0113	8	0.0115	23	0.0138
109	Extended Block-Diagonal BD1	7	0.0345	8	0.0374	31	0.1184
110	Extended Block-Diagonal BD1	7	0.2876	9	0.287	13	0.6905
111	HIMMELBG	1	0.0109	3	0.0119	7	0.0121
112	HIMMELBG	1	0.0024	3	0.0021	7	0.0033
113	HIMMELBG	1	0.0031	3	0.0036	7	0.0045
114	HIMMELBH	4	0.0079	4	0.0074	15	0.0072
115	HIMMELBH	4	0.001	4	0.0006591	19	0.0029
116	HIMMELBH	4	0.0011	4	0.0013	22	0.0043

continued on next page

Table 2: Numerical results for MTT with MTTHS and HCGM in terms of NOI and CPU Time (Continued)

117	Extended Hiebert	22	0.0437	36	0.0472	-	-
118	Extended Hiebert	22	0.2521	58	0.4567	-	-
119	Extended Hiebert	22	2.2975	112	8.044	-	-
120	Linear Perturbed	54	0.0151	54	0.0145	629	0.0772
121	Linear Perturbed	406	1.0354	410	1.0398	-	-
122	Linear Perturbed	1310	45.8803	1363	49.7695	-	-
123	QUARTICM	42	0.2271	13	0.086	-	-
124	QUARTICM	109	51.2399	18	10.5822	-	-
125	Zirilli or Aluffi-Pentini's	4	0.0065	5	0.0066	9	0.0073
126	Zirilli or Aluffi-Pentini's	4	0.0002242	4	0.000123	5	0.000211
127	Extended Quadratic Penalty QP3	10	0.0115	14	0.0098	-	-
128	Extended Quadratic Penalty QP3	10	0.0018	-	-	-	-
129	Extended Quadratic Penalty QP3	15	0.0084	-	-	-	-
130	DIAG-AUP1	4	0.1897	14	0.1192	16	0.1582
131	Strait	17	0.0282	36	0.0423	449	0.3066
132	Strait	16	1.1067	34	2.1641	-	-
133	Strait	24	16.2049	50	47.4095	-	-
134	Perturbed Quadratic	2	0.0093	3	0.0001826	14	0.0078
135	Perturbed Quadratic	2	0.0001905	4	0.0001576	9	0.0002933
136	Perturbed Quadratic	2	0.0001502	4	0.0001928	17	0.0006843
137	Diagonal 2	9	0.0113	7	0.0123	16	0.0104
138	Diagonal 2	97	0.0063	24	0.0052	61	0.0045

Figure 1 and Figure 2 showed the performance profiles based on the number of iterations and CPU time, respectively. The analysis of Figure 2 was conducted by considering the CPU time, measured in seconds. The analysis was performed in order to estimate the time required to generate search direction with a specific objective of executing a line search and convergence test.

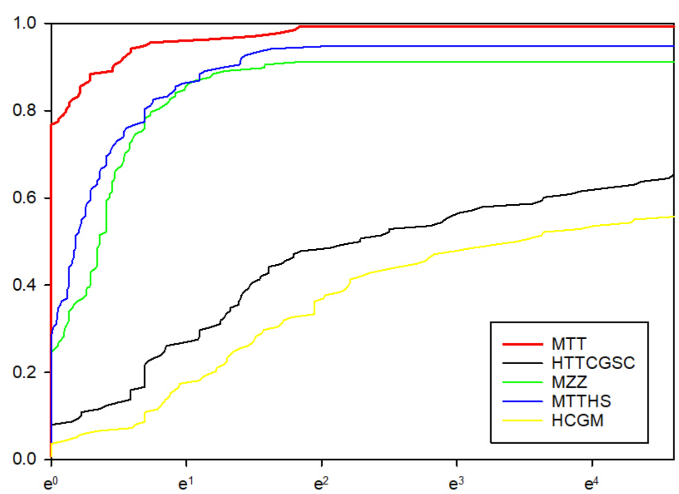

Figure 1 : Performance profile corresponding to the Number of Iterations

Figure 1 showed that the proposed method, MTT outperformed the HTTCGSC, MZZ, MTTHS, and HCGM in terms of number of iterations, as it exhibits less number of iterations. According to the data presented in Figure 1, the MTT method achieves a success rate of 99% in solving the test problems with the least iteration number. Meanwhile, it can be inferred that the MTTHS method at 94%, the MZZ method at 93%, the HTTCGSC method at 70%,, and the HCGM method at 60% in solving the test problems. As shown in Figure 1, it can be clearly seen that there is no competition between MTT with other methods. Furthermore, it is evident that the proposed method, MTT is the only method that approaches 1.0 the fastest. This indicates that MTT method performs the best and more robust than all other methods in terms of number of iterations.

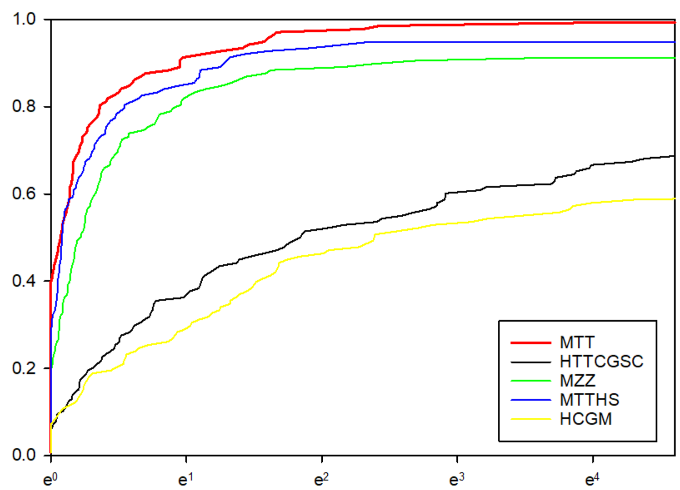

Figure 2 : Performance profile corresponding to the CPU time

On the other hand, according to Figure 2, it can be observed that MTT method exhibits a faster convergence time in comparison to HTTCGSC, MZZ, MTTHS, and HCGM. However, in the early stage, there is some competition between MTT and MTTHS. From analysis, we have that MTT method at 99%, MTTHS method at 94%, the MZZ method at 91%, the HTTCGSC at 70% and the HCGM method at 59% in solving all the test problems. This indicates that the MTT method achieves the best results in terms of the amount of CPU time. Therefore, from both Figure 1 and Figure 2, it can be concluded that MTT method outperformed other methods in terms of number of iterations and CPU time in solving all the test problems.

5 CONCLUSION

In this study, a modification of hybrid three-term conjugate gradient method has been presented. Subsequently, the new algorithm namely MTT has been utilized to solve large-scaled unconstrained optimization problems. The search direction of the algorithm always satisfies sufficient descent property regardless of any line search. In addition, the step size was obtained via strong Wolfe-Powell line search. Convergence of the algorithm was also analyzed under certain assumptions. In order to further support the convergence results, a numerical experiment was conducted by focusing on tackling the problem of large-scaled unconstrained optimization with 138 test problems. Finally, the results showed that the MTT method demonstrates superiority compared methods in terms of efficiency and robustness.

ACKNOWLEDGEMENT

The authors express their thanks to the editors and referees for their insightful comments and suggestions.

REFERENCES

[1] M. R. Hestenes, “The conjugate-gradient method for solving,” *Numerical analysis*, no. 6, p. 83, 1956.

[2] J. Wang, W. Wu, and J. M. Zurada, “Deterministic convergence of conjugate gradient method for feedforward neural networks,” *Neurocomputing*, vol. 74, no. 14-15, pp. 2368–2376, 2011.

[3] R. Fletcher and C. M. Reeves, “Function minimization by conjugate gradients,” *The computer journal*, vol. 7, no. 2, pp. 149–154, 1964.

[4] B. T. Polyak, “The conjugate gradient method in extremal problems,” *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 4, pp. 94–112, 1969.

[5] M. R. Hestenes, E. Stiefel *et al.*, “Methods of conjugate gradients for solving linear systems,” *Journal of research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[6] Y. Liu and C. Storey, “Efficient generalized conjugate gradient algorithms, part 1: theory,” *Journal of optimization theory and applications*, vol. 69, pp. 129–137, 1991.

[7] Y.-H. Dai and Y. Yuan, “A nonlinear conjugate gradient method with a strong global convergence property,” *SIAM Journal on optimization*, vol. 10, no. 1, pp. 177–182, 1999.

[8] R. Fletcher, *Practical methods of optimization.* John Wiley & Sons, 2000.

[9] Q. Tian, X. Wang, L. Pang, M. Zhang, and F. Meng, “A new hybrid three-term conjugate gradient algorithm for large-scale unconstrained problems,” *Mathematics*, vol. 9, no. 12, p. 1353, 2021.

[10] L. Zhang, W. Zhou, and D. Li, “Some descent three-term conjugate gradient methods and their global convergence,” *Optimisation Methods and Software*, vol. 22, no. 4, pp. 697–711, 2007.

[11] H. Wasi and M. A. Shiker, “A new hybrid cgm for unconstrained optimization problems,” in *Journal of Physics: Conference Series*, vol. 1664, no. 1. IOP Publishing, 2020, p. 012077.

[12] S. Babaie-Kafaki, “On optimality of the parameters of self-scaling memoryless quasi-newton updating formulae,” *Journal of Optimization Theory and Applications*, vol. 167, pp. 91–101, 2015.

[13] Y. H. Dai and L. Z. Liao, "New conjugacy conditions and related nonlinear conjugate gradient methods," *Applied Mathematics and optimization*, vol. 43, pp. 87–101, 2001.

[14] L. Zhang, W. Zhou, and D.-H. Li, "A descent modified polak–ribière–polyak conjugate gradient method and its global convergence," *IMA Journal of Numerical Analysis*, vol. 26, no. 4, pp. 629–640, 2006.

[15] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical programming*, vol. 91, pp. 201–213, 2002.

APPENDIX

Table 3 : List of Problems Functions

No	Functions	Dimension	No	Functions	Dimension
1	Extended White & Holst	50000	23	Extended Himmelblau	50000
2	Extended White & Holst	100000	24	Extended Himmelblau	100000
3	Extended White & Holst	1000000	25	FLETCHCR	100
4	Extended Rosenbrock	50000	26	FLETCHCR	5000
5	Extended Rosenbrock	100000	27	FLETCHCR	50000
6	Extended Rosenbrock	1000000	28	NONSCOMP	2
7	Extended Freudenstein & Roth	1000	29	NONSCOMP	4
8	Extended Freudenstein & Roth	50000	30	Extended DENSCHNB	1000
9	Extended Freudenstein & Roth	100000	31	Extended DENSCHNB	50000
10	Extended Beale	1000	32	Extended DENSCHNB	100000
11	Extended Beale	50000	33	Extended Penalty Function U52	5
12	Extended Beale	100000	34	Extended Penalty Function U52	10
13	Raydan 1	10	35	Extended Penalty Function U52	50
14	Raydan 1	50	36	Hager	5
15	Raydan 1	100	37	Hager	10
16	Extended Tridiagonal 1	10	38	Hager	50
17	Extended Tridiagonal 1	50	39	Cube	2
18	Extended Tridiagonal 1	100	40	Extended Maratos	10
19	Diagonal 4	1000	41	Extended Maratos	50
20	Diagonal 4	5000	42	Extended Maratos	100
21	Diagonal 4	50000	43	Six Hump Camel	2
22	Extended Himmelblau	1000	44	Six Hump Camel	2
45	Three Hump Camel	2	92	Sphere	100000
46	Booth	2	93	Sum Squares	1000
47	Booth	2	94	Sum Squares	10000
48	Trecanni	2	95	Sum Squares	50000
49	Trecanni	2	96	DENSCHNA	1000
50	Zettl	2	97	DENSCHNA	10000
51	Zettl	2	98	DENSCHNA	100000
52	Shallow	1000	99	DENSCHNB	100
53	Shallow	50000	100	DENSCHNB	5000
54	Shallow	100000	101	DENSCHNB	50000
55	Generalized Quartic	100	102	DENSCHNC	100
56	Generalized Quartic	5000	103	DENSCHNC	5000
57	Generalized Quartic	10000	104	DENSCHNC	50000
58	Quadratic QF2	10	105	DENSCHNF	100
59	Quadratic QF2	100	106	DENSCHNF	5000
60	Quadratic QF2	1000	107	DENSCHNF	50000
61	Leon	2	108	Extended Block-Diagonal BD1	100
62	Leon	2	109	Extended Block-Diagonal BD1	5000
63	Generalized Tridiagonal 1	5	110	Extended Block-Diagonal BD1	50000
64	Generalized Tridiagonal 1	10	111	HIMMELBG	10
65	Generalized Tridiagonal 1	100	112	HIMMELBG	50
66	Generalized Tridiagonal 2	10	113	HIMMELBG	100
67	Generalized Tridiagonal 2	50	114	HIMMELBH	10
68	Generalized Tridiagonal 2	500	115	HIMMELBH	50
69	POWER	10	116	HIMMELBH	100
70	POWER	50	117	Extended Hiebert	1000
71	POWER	500	118	Extended Hiebert	10000
72	Quadratic QF1	100	119	Extended Hiebert	100000

continued on next page

Table 3: List of Problem Functions (Continued)

73	Quadratic QF1	1000	120	Linear Perturbed	100
74	Quadratic QF1	10000	121	Linear Perturbed	5000
75	Extended Quadratic Penalty QP2	5	122	Linear Perturbed	50000
76	Extended Quadratic Penalty QP2	50	123	QUARTICM	1000
77	Extended Quadratic Penalty QP2	500	124	QUARTICM	50000
78	Extended Quadratic Penalty QP1	5	125	Zirilli or Aluffi-Pentini's	2
79	Extended Quadratic Penalty QP1	10	126	Zirilli or Aluffi-Pentini's	2
80	Extended Quadratic Penalty QP1	100	127	Extended Quadratic Penalty QP3	5
81	Quartic	4	128	Extended Quadratic Penalty QP3	10
82	Quartic	4	129	Extended Quadratic Penalty QP3	50
83	Matyas	2	130	DIAG-AUP1	10000
84	Matyas	2	131	Strait	1000
85	Colville	4	132	Strait	100000
86	Colville	4	133	Strait	1000000
87	Dixon and Price	1000	134	Perturbed Quadratic	2
88	Dixon and Price	10000	135	Perturbed Quadratic	2
89	Dixon and Price	100000	136	Perturbed Quadratic	2
90	Sphere	1000	137	Diagonal 2	2
91	Sphere	10000	138	Diagonal 2	10