Asia-Pacific Journal of Information Technology and Multimedia 15(1) 2026: 211 – 223
https://doi.org/10.17576/apjitm-2026-1501-12

Sufficient - Descent Scaled Conjugate Gradient Direction for Large-Scale Unconstrained Optimization

Arah Kecerunan Konjugat Berskala Berpenurunan Mencukupi untuk Pengoptimuman Tanpa Kekangan Berskala Besar

Sulaiman Mohammed Ibrahim[1,], Lawal Muhammad[2], Habibulla Ahadkulov[3], Mohd Kamal M. Nawawi[4]*

[1]Department of Basic and Applied Sciences, College of Applied and Health Sciences, A'Sharqiyah University, Oman
[2]Department of Mathematics, Faculty of Science, Northwest University Kano, Kano, Nigeria
[3]Department of Higher and Applied Mathematics Mathematics, Tashkent State University of Economics, Uzbekistan
[4]School of Quantitative Sciences, Universiti Utara Malaysia, UUM Sintok, 06010, Kedah, Malaysia

**Corresponding author: sulaimancga@gmail.com*

Received 17 September 2025
Accepted 6 May 2026, Available online 30 June 2026

ABSTRACT

Conjugate gradient (CG) methods are extensively utilized for solving large-scale unconstrained optimization problems due to their low memory requirements and strong practical performance. However, achieving descency property and maintaining efficiency across diverse optimization landscapes remains challenging. This study addresses these limitations by introducing a novel scaled conjugate gradient direction that integrates a self-scaling parameter into the classical framework. The proposed methodology derives from quasi-Newton principles, ensuring the sufficient descent condition independent of line search precision. Numerical experiments on 120 benchmark problems from the CUTEr library, spanning dimensions from 2 to 600,000 variables, demonstrate the robustness and efficiency of the scaled method.

Keywords: Scaled conjugate gradient method, Sufficient descent properties, Benchmark problems.

ABSTRAK

Kaedah kecerunan konjugat (CG) digunakan secara meluas untuk menyelesaikan masalah pengoptimuman tanpa kekangan berskala besar kerana keperluan memori yang rendah serta prestasi praktikal yang kukuh. Namun, mencapai sifat penurunan (descent property) dan mengekalkan kecekapan merentasi pelbagai landskap pengoptimuman masih menjadi cabaran. Kajian ini menangani kekangan tersebut dengan memperkenalkan satu arah kecerunan konjugat berskala baharu yang mengintegrasikan parameter penskalaan kendiri ke dalam rangka kerja klasik. Kaedah yang dicadangkan ini diperoleh daripada prinsip quasi-Newton, yang menjamin syarat penurunan mencukupi tanpa bergantung kepada ketepatan carian garisan. Eksperimen berangka ke atas 120 masalah penanda aras daripada perpustakaan CUTEr, yang merangkumi dimensi daripada 2 hingga 600,000 pemboleh ubah, menunjukkan keteguhan dan kecekapan kaedah berskala tersebut.

Kata Kunci: Kaedah kecerunan konjugat berskala, Sifat penurunan mencukupi, Masalah penanda aras.

INTRODUCTION

This work focuses on the unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f:\mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, and its gradient $\nabla f(x)$ is available (Sulaiman *et al.*, 2019).

The classical second-order methods, including Newton's method, compute their search direction by utilizing gradient and Hessian information, thereby attributing to their fast local convergence. In large-scale problems, however, it is impractical to compute and store the Hessian matrix. The quasi-Newton methods reduce this burden by approximating the Hessian but still require updates of the matrix which could be expensive in high dimensions *(Bojari & Eslahchi, 2020).* The conjugate gradient (CG) method, however, can be applied in a scalable manner which requires only the first-order information without having to store the matrix *(Awwal et al., 2023; Salihu et al., 2023; Malik et al., 2020a).* The CG algorithm generates a sequence of iterative points using:

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

where $\alpha_k > 0$ is a step size determined by a line search, and d_k is the search direction defined recursively by

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -g_k + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{3}$$

with $g_k = \nabla f(x_k)$ and β_k being the conjugate gradient parameter (Omesa *et al.*, 2025; Mamat *et al*., 2020).

RELATED WORK

The robustness of the CG algorithms is greatly influence by the selection of β_k, leading to a variety of variants and modifications *(Mohammed et al., 2015; Malik et al., 2020b).* For example, the method of *Polak & Ribiere (1969)* and *Polyak (1969)* and its variants have proven to be promising in practice, although their theoretical convergence is not guaranteed to hold under some line search techniques. One notable enhancement is the restart-based RMIL formula proposed by Rivaie *et al.* (2012a):

$$\beta_k^{RMIL} = \frac{g_k^T y_{k-1}}{\|d_{k-1}\|^2}, \tag{4}$$

where $y_{k-1} = g_k - g_{k-1}$. This method has been shown to converge globally under suitable assumption. However, Dai (2016) pointed out that inequality (3.3) used in Mamat Rivaie et al. (2012) is incorrect. Consequently, several theorems and lemmas whose proofs rely on this inequality are not valid. The necessary corrections and modifications to address this issue were subsequently suggested in Dai (2016). For more references on related studies (see; Yuan & Zhang, 2015; Al-Baali 1985; Touti-Ahmed & Storey, 1990; Zhang *et al*., 2006; Cheng, 2007; Mohammed *et al*., 2024; Salihu *et al*., 2023).

Recent research has sought to bridge CG methods with the underlying principles of quasi-Newton updates and Barzilai–Borwein (BB) scaling strategies (Sabi'u *et al.*, 2024). These efforts aim to retain the low memory footprint of CG while incorporating curvature information more effectively. In this spirit, the present work proposes a new class of scaled CG method, inspired by the mRMIL direction and quasi-Newton structure, but designed to operate in a fully matrix-free manner.

The main contributions of this study are summarized as follows:

- A new scaled conjugate gradient (CG) method that incorporates a quasi-Newton direction without requiring matrix operations is developed.
- It is proved that the proposed method satisfies the sufficient descent condition regardless of the accuracy of the line search.
- The effectiveness of the proposed algorithm is demonstrated through extensive numerical experiments on a collection of standard test problems.

The rest of the paper is organized as follows. Section 2 describes the proposed scaling strategy and further provides a theoretical analysis establishing the sufficient descent property. Section 3 presents numerical results to evaluate the proposed methods, and Section 4 concludes the paper.

METHODOLOGY

This section presents the scaled conjugate gradient method for solving unconstrained optimization problems. The mRMIL CG parameter by Rivaie *et al*., (2012b) is given by:

$$\beta_k^{mRMIL} = \frac{g_k^T y_{k-1}}{d_{k-1}^T(d_{k-1} - g_k)}. \tag{5}$$

Scaling strategy has been an effective procedure for enhancing theoretical and performances of the CG algorithms (Amini et al., 2018; Andrie, 2007; Oren & Luenberger, 1974) . In line with these developments, this study proposed a scale mRMIL conjugate gradient parameter as follows:

$$\beta_k^{smRMIL} = \mu_k \beta_k^{mRMIL}, \tag{6}$$

such that $|\mu_k| \leq 1$. Gilbert & Nocedal (1992) defined a new CG formula and showed that the FR CG method with parameter $\beta_k = \mu_k \beta_k^{FR}$ in which $|\mu_k| \leq 1$, satisfies the sufficient descent condition (Mirhoseini & Babaie-Kafaki 2022). Inspired by this, the contribution of this study is determining the value of the scalar μ_k at each iteration.

Determining μ_k via the quasi-Newton direction
We intend to determine an optimal scaling choice parameter μ_k at each iteration via quasi-Newton approach. Recall the quasi-Newton equation which is defined as

$$y_k = U_{k+1}s_k, \quad (7)$$

where U_{k+1} is a positive definite and symmetric Hessian approximate. From the fact that $U_{k+1} = U_{k+1}^T$. Thus (7) can be written as

$$y_k^T = s_k^T U_{k+1}. \quad (8)$$

Again, assuming that V_{k+1} is the inverse of U_{k+1}, the quasi-Newton direction is as follows:

$$d_{k+1} = -V_{k+1}g_{k+1}. \quad (9)$$

Now, using the general CG direction, the scaled search direction is define as

$$d_{k+1} = -g_{k+1} + \mu_k\beta_k d_k, \quad k = 0,1,\cdots. \quad (10)$$

where β_k is defined as (5). Since quasi-Newton approaches use the real Hessian approximations, hence, from (9) and (10), we can incorporate the Hessian approximation information in the quasi-Newton direction into our approach as well. Additionally, by virtue of the equality relation, we obtain

$$-V_{k+1}g_{k+1} = -g_{k+1} + \mu_k\beta_k^{mRMIL}d_k. \quad (11)$$

Multiplying (11) by $-s_k^T U_{k+1}$ will produce

$$s_k^T g_{k+1} = s_k^T U_{k+1}g_{k+1} - \mu_k\beta_k^{mRMIL}s_k^T U_{k+1}d_k. \quad (12)$$

Using the modified quasi-Newton equation (8), we further eliminate the matrix U_{k+1} in (12) to obtain

$$s_k^T g_{k+1} = y_k^T g_{k+1} - \mu_k\beta_k^{mRMIL}y_k^T d_k. \quad (13)$$

To clarify the derivation of μ_k, recall that $s_k = x_{k+1} - x_k$and $y_k = g_{k+1} - g_k$, which are commonly used in quasi-Newton methods to capture curvature information of the objective function. Since U_{k+1}represents a symmetric positive definite approximation of the Hessian matrix, the quasi-Newton relation $y_k = U_{k+1}s_k$ is utilized to incorporate second-order information without explicitly forming the Hessian matrix. The algebraic manipulation from (11) to (13) exploits this relation to eliminate the matrix U_{k+1}, thereby obtaining an expression that depends only on the vectors s_k, y_k, and the search direction d_k. This substitution significantly reduces computational complexity while preserving curvature information required to determine the scaling parameter μ_k.

Now, solving for μ_k in (13) to get

$$\mu_k = \frac{(y_k - s_k)^T g_{k+1}}{\beta_k^{mRMIL}y_k^T d_k}. \quad (14)$$

Furthermore, we defined the following modified version of (14) in order to achieve $|\mu_k| \leq 1$ and to benefit from the nonnegative restriction of mRMIL.

$$\mu_k^* = \min\{1, |\mu_k|\}. \tag{15}$$

The bounded choice $\mu_k^* = \min(1, |\mu_k|)$ is introduced to ensure that the scaling parameter remains within a stable range. In practice, the unconstrained value of μ_k obtained from (14) may occasionally exceed unity, which could lead to an excessively large scaling of the search direction. Restricting the parameter using $\min(1, |\mu_k|)$ prevents overly aggressive updates and helps maintain the stability of the generated search direction while preserving the descent property of the algorithm.

More so, to ensure the sufficient descent condition is satisfied, we present the new scaled mRMIL CG direction as follows:

$$d_{k+1} = -\eta_k g_{k+1} + \mu_k^* \beta_k^{mRMIL} d_k, \quad k = 0,1,\cdots, \tag{16}$$

where

$$\eta_k = 1 + \mu_k^* \frac{\beta_k^{mRMIL} g_{k+1}^T d_k}{\|g_{k+1}\|^2} \tag{17}$$

Remark 2.1: The direction (16) satisfies the SDC.

To justify the descent property of the proposed direction, consider the inner product between the gradient g_{k+1} and the search direction d_{k+1}. By multiplying (16) with g_{k+1}^T and substituting the definition of η_k given in (17), it can be shown that the additional scaling term cancels out. Consequently, the resulting expression reduces to a strictly negative quantity, which guarantees that the generated direction satisfies the sufficient descent condition.
Multiplying (16) by g_{k+1}^T to get

$$g_{k+1}^T d_{k+1} = -\eta_k \|g_{k+1}\|^2 + \mu_k^* \beta_k^{mRMIL} g_{k+1}^T d_k, \quad k = 0,1,\cdots. \tag{18}$$

Substituting the value of η_k in (18)

$$g_{k+1}^T d_{k+1} = -\left(1 + \mu_k^* \frac{\beta_k^{mRMIL} g_{k+1}^T d_k}{\|g_{k+1}\|^2}\right) \|g_{k+1}\|^2 + \mu_k^* \beta_k^{mRMIL} g_{k+1}^T d_k, \quad k = 0,1,\cdots. \tag{19}$$

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 - \mu_k^* \beta_k^{mRMIL} g_{k+1}^T d_k + \mu_k^* \beta_k^{mRMIL} g_{k+1}^T d_k, \quad k = 0,1,\cdots. \tag{20}$$

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2. \tag{21}$$

The fact that exact line searches both seem to be expensive and impractical, inexact line searches are mostly applied. The Wolfe approach is frequently used to find the step length α_k in the CG methods. The standard Wolfe line search is as follows.

$$f(x_k + \alpha_k d_k)^T \leq f(x_k) + \tau_0 \alpha_k g_k^T d_k \tag{22}$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \tau_1 g_k^T d_k \tag{23}$$

where $0 < \tau_0 < \tau_1 < 1$.

The following algorithm detailed out the computational procedure of the proposed smRMIL.

Algorithm 1: A Modified Scaled smRMIL CG Algorithm
1: **Input:** $x_0 \in \mathbb{R}^n$, $\epsilon > 0$, $\tau_0 \in (0,1)$, $\tau_1 \in (0,1)$
2: **Initialize:** $k = 0$, compute $d_0 = -g_0$
3: **while** $\|g_{k+1}\| \leq 10^{-6}$ **do**
4: Compute step size α_k using conditions in (22) and (23)
5: Compute new direction d_{k+1} using (16)
6: Update $x_{k+1} = x_k + \alpha_k d_k$
7: Set $k = k + 1$
8: **end while**

RESULTS AND DISCUSSION

This section presents a comprehensive performance analysis of the proposed CG algorithm (smRMIL) across various problem instances. The numerical experiments conducted assess the robustness, computational efficiency, and convergence behavior of the new methods across different problem sizes and characteristics. The study compares the performance of the new method against other classical modifications of CG algorithms to evaluate its efficacy in terms of number of iterations (NOI), convergence speed (CPUT, and number of function evaluation (FEV).

Unconstrained optimization

To evaluate the performance of the proposed methods, the study conducted a numerical test on a collection of 120 unconstrained optimization test problems, most of which are from CUTEr library (Gould *et al*., 2003). For each testing problem, a range of dimensions which comprise small-scale as low as 2 and large-scale dimensions up to 600,000 variables are considered. The accuracy of the computed solutions is assessed by comparing them against reference solutions obtained from the following conjugate gradient methods:

- Classical RMIL formula (4) by Rivaie et al., (2012a)
- A modified mRMIL method (5) by Rivaie et al., (2012b)

All algorithms are implemented in MATLAB R2023b standard numerical computing environment, utilizing double-precision arithmetic for accurate computations while the experiment was performed on a Windows 11 operating system personal computer whose specifications include a 10th Gen Intel Core i7-1065G7 processor, 16GB of RAM, and 512GB of SSD1234. The line search considered for the numerical computation is the Wolfe line search with the parameter values $\tau_0 = 0.01$, $\tau_1 = 0.1$. The termination criterion for all algorithms is set as

$$\| g(\mathrm{x}_k) \| < 10^{-6},$$

of if any of the following holds:

- The iteration number reaches 5000.
- The code fails to execute due to low memory.

A detailed description of the numerical results is presented in Tables 1 – 3.

TABLE 1. Numerical performance based on NOI, FEV, and CPU time

				smRMIL			mRMIL			RMIL	
S/N	**Test Functions**	**Dimm**	**NOI**	**CPUT**	**FEV**	**NOI**	**CPUT**	**FEV**	**NOI**	**CPUT**	**FEV**
F1	cosine	6000	252	0.17878	305	231	0.123035	283	311	0.167812	365
F2	cosine	20000	405	0.535451	453	419	0.547394	475	687	0.96332	745
F3	cosine	50000	731	2.240261	785	1113	10.44084	1170	617	5.769405	669
F4	dixmaana	6000	515	1.136462	565	62	0.229965	120	70	0.24634	127
F5	dixmaana	30000	97	1.357141	157	118	5.061837	171	128	5.46166	189
F6	dixmaana	90000	192	9.214281	246	153	17.5168	206	173	19.3836	230
F7	dixmaanb	12000	93	0.599638	149	71	0.467598	129	200	0.945478	266
F8	dixmaanb	24000	92	1.04878	153	110	4.001797	170	194	6.089348	259
F9	dixmaanb	48000	118	2.412608	177	118	8.09351	178	138	8.820641	197
F10	dixmaanc	900	44	0.070461	107	43	0.033443	100	55	0.033248	112
F11	dixmaanc	2700	44	0.105628	99	93	0.140959	151	68	0.144613	126
F12	dixmaanc	27000	117	1.347627	176	510	14.50921	567	98	4.037752	155
F13	dixmaand	9000	60	0.359398	119	72	0.361466	137	84	0.387063	142
F14	dixmaand	12000	265	1.151503	323	84	0.493918	143	95	0.878871	155
F15	dixmaand	90000	145	5.465989	204	153	17.83424	211	199	21.44068	260
F16	dixmaane	30	79	0.028886	130	302	0.016467	363	424	0.020152	479
F17	dixmaane	90	1564	0.103747	1621	199	0.022506	267	**	**	**
F18	dixmaane	150	1012	0.094468	1079	338	0.036222	401	652	0.060789	704
F19	dixmaang	600	1717	0.531627	1793	**	**	**	**	**	**
F20	dixmaanh	30	363	0.036985	423	488	0.045675	542	1133	0.079864	1192
F21	dixmaanh	300	388	0.077184	465	**	**	**	**	**	**
F22	dixmaanh	600	1006	0.266639	1078	**	**	**	**	**	**
F23	dixmaanj	6	140	0.026665	203	117	0.008801	171	135	0.008808	186
F24	dixmaank	6	229	0.039832	298	229	0.018836	281	95	0.006843	164
F25	dixmaanl	6	843	0.054151	905	623	0.02247	686	664	0.023328	728
F26	dqdrtic	9000	555	0.08294	618	456	0.052553	521	**	**	**
F27	fletchcr	10	512	0.021898	566	306	0.008416	362	**	**	**
F28	fletchcr	100	248	0.005075	303	**	**	**	696	0.015405	753
F29	himmelbg	70000	372	1.29	383	372	1.267494	383	373	3.357096	384
F30	himmelbg	240000	686	13.11888	697	686	23.90996	697	687	22.89849	698
F31	bdexp	5000	26	0.047812	37	26	0.030635	37	27	0.024665	38
F32	bdexp	50000	78	0.500303	89	78	0.481172	79	78	0.545194	89
F33	bdexp	500000	239	52.19194	250	239	15.92289	250	240	15.69513	251
F34	biggsb1	4	47	0.009272	94	77	0.002525	124	163	0.004917	209
F35	biggsb1	10	132	0.002976	182	121	0.002816	160	**	**	**
F36	raydan1	1000	747	0.045758	806	363	0.019167	416	1387	0.088232	1442
F37	raydan1	5000	1090	0.304105	1151	**	**	**	**	**	**
F38	raydan2	1000	30	0.016197	75	29	0.005736	76	29	0.005394	76
F39	raydan2	2000	33	0.009218	73	36	0.011383	83	34	0.008956	74
F40	raydan2	5000	47	0.033441	91	53	0.03793	101	47	0.033239	91
F41	raydan2	10000	64	0.077916	111	**	**	**	63	0.070673	109
F42	raydan2	20000	72	0.16842	119	**	**	**	75	0.172633	119
F43	raydan2	50000	125	0.602705	172	123	0.609758	167	**	**	**
F44	trid	100	109	0.025706	172	93	0.014414	146	132	0.016232	198
F45	trid	500	298	0.287011	354	1003	0.773515	1069	721	0.571948	779
F46	trid	2000	1777	16.37644	1836	167	1.990932	224	126	1.828563	197
F47	trid	5000	59	6.333548	122	204	13.38705	266	118	9.547304	191
F48	trid	8000	141	26.86329	209	125	23.51158	187	96	21.60514	163
F49	DRCAV1LQ	6000	55	0.034916	109	51	0.009186	96	52	0.009358	98
F50	DRCAV1LQ	20000	90	0.045565	1140	87	0.040643	135	88	0.037161	136

TABLE 2. Numerical performance based on NOI, FEV, and CPU time

				smRMIL			mRMIL			RMIL	
S/N	**Test Functions**	**Dimm**	**NOI**	**CPUT**	**FEV**	**NOI**	**CPUT**	**FEV**	**NOI**	**CPUT**	**FEV**
F51	DRCAV1LQ	100000	178	0.372662	231	178	0.262181	230	176	0.261117	228
F52	DRCAV2LQ	2	22	0.008262	66	22	0.001562	66	22	0.00108	66
F53	DRCAV2LQ	4	178	0.00785	230	789	0.014579	842	1905	0.031443	1954
F54	DRCAV2LQ	6	272	0.004859	322	225	0.005736	275	889	0.013122	939
F55	DRCAV2LQ	8	302	0.005441	353	257	0.004985	308	907	0.014176	960
F56	penalty1	4000	29	1.009906	94	27	0.855176	80	**	**	**
F57	dqdrtic	9000	555	0.085038	618	456	0.064723	521	**	**	**
F58	exdenschnf	90000	140	0.359014	213	146	0.32218	217	334	0.506249	401
F59	exdenschnf	280000	220	1.811706	290	668	4.414156	746	265	1.757803	340
F60	exdenschnf	600000	839	13.19716	916	312	4.751451	389	392	5.574966	468
F61	exdenschnb	6000	86	0.037827	146	92	0.015741	152	109	0.0199	165
F62	exdenschnb	8000	101	0.022301	152	93	0.017031	150	288	0.036044	341
F63	exdenschnb	80000	188	0.243163	245	188	0.210835	242	302	0.27579	355
F64	nonscomp	30000	656	0.452831	729	343	0.202942	411	270	0.140561	340
F65	nonscomp	80000	553	0.788967	618	**	**	**	795	0.967237	865
F66	genquartic	6000	67	0.040088	123	100	0.018789	161	59	0.012575	114
F67	genquartic	9000	78	0.020152	140	80	0.023548	137	73	0.026857	134
F68	genquartic	30000	88	0.110328	147	220	0.212807	273	107	0.151543	168
F69	ie	100	133	0.695862	180	27	0.236694	76	64	0.387318	123
F70	ie	300	21	2.089927	71	57	2.757017	105	345	10.64109	402
F71	ie	500	40	6.682922	90	23	4.91479	66	57	7.71099	103
F72	lin	100	27	0.072409	80	27	0.024419	80	23	0.0221	69
F73	lin	100	27	0.023414	80	27	0.027552	80	23	0.030576	69
F74	lin	500	43	0.232564	92	42	0.226649	89	42	0.250228	90
F75	NCB2	1000	663	0.055781	745	**	**	**	532	0.021948	608
F76	NCB3	2	476	0.01557	530	239	0.008995	280	111	0.002651	157
F77	NCB4	2	1312	0.033991	1358	162	0.003767	213	566	0.00921	619
F78	NCB5	10000	676	0.154745	741	745	0.134996	816	739	0.13039	790
F79	NCB5	20000	1114	0.450964	1170	1178	0.454531	1247	911	0.345726	978
F80	MSQRTBLS	10	15	0.015448	55	15	0.002444	55	15	0.001632	58
F81	MSQRTBLS	100	21	0.001712	69	14	0.001561	51	14	0.001109	51
F82	NCB20B	5000	303	0.039051	370	1125	0.100758	1176	314	0.031069	369
F83	NCB20B	10000	476	0.100097	527	221	0.047165	295	804	0.173644	861
F84	NCB20B	20000	523	0.286465	586	825	0.327047	879	203	0.087804	270
F85	EIGENBLS	10000	44	0.060293	94	45	0.044913	94	44	0.038577	92
F86	EIGENBLS	20000	56	0.101091	109	57	0.094614	111	55	0.089634	106
F87	EIGENALS	100	28	0.012084	79	27	0.004378	78	27	0.003263	75
F88	EIGENCLS	5000	287	0.14163	339	276	0.120749	323	277	0.14321	331
F89	EIGENCLS	10000	406	0.323955	460	382	0.287451	434	382	0.29966	434
F90	EIGENCLS	20000	545	0.98275	600	564	0.973281	635	543	0.869026	598
F91	OSCIPATH	2	17	0.008732	77	17	0.001977	74	21	0.001361	75
F92	MSQRTALS	1000	35	0.013842	81	35	0.003121	82	**	**	**
F93	VARDIM	1000	40	0.03621	87	57	0.042882	109	98	0.043618	148
F94	VARDIM	5000	74	0.202299	136	103	0.269343	163	95	0.250412	152
F95	VARDIM	10000	95	0.470685	149	184	0.756874	248	108	0.54277	169
F96	CURLY30	100	433	0.046025	479	55	0.009463	116	194	0.015525	247
F97	CURLY30	500	138	0.050828	199	82	0.03465	138	138	0.041157	192
F98	dixon3dq	4	387	0.014505	442	627	0.016146	677	736	0.013231	777
F99	edensch	10	99	0.014936	164	111	0.006219	169	179	0.007932	240
F100	edensch	1000	370	0.119126	432	**	**	**	**	**	**

TABLE 3. Numerical performance based on NOI, FEV, and CPU time

			smRMIL			mRMIL			RMIL		
S/N	**Test Functions**	**Dimm**	**NOI**	**CPUT**	**FEV**	**NOI**	**CPUT**	**FEV**	**NOI**	**CPUT**	**FEV**
F101	eg2	4	249	0.013227	306	200	0.005756	244	**	**	**
F102	liarwhd	2	41	0.014423	99	237	0.005813	293	72	0.002689	134
F103	liarwhd	4	62	0.005547	120	109	0.006508	170	96	0.005033	152
F104	quartc	100	222	0.028522	290	193	0.013644	270	249	0.014437	312
F105	quartc	500	1523	0.270191	1606	1523	0.251698	1618	1761	0.289223	1835
F106	tridia	4	178	0.011489	243	122	0.005717	180	279	0.0086	333
F107	tridia	10	418	0.011294	479	**	**	**	**	**	**
F108	sine	1000	64	0.024591	123	40	0.01033	98	42	0.009643	98
F109	sine	10000	134	0.185728	189	58	0.112945	113	126	0.153718	191
F110	sine	20000	159	0.35005	218	167	0.426948	228	**	**	**
F111	sine	50000	129	0.872729	201	329	1.479509	404	212	1.25924	276
F112	sine	100000	692	6.825628	798	239	3.11768	304	154	1.761807	207
F113	diagonal2	1000	906	0.115167	960	**	**	**	**	**	**
F114	diagonal3	2	42	0.010607	96	**	**	**	201	0.004365	256
F115	diagonal3	4	164	0.003371	216	59	0.002242	112	242	0.00455	295
F116	diagonal3	10	520	0.011219	577	278	0.005901	331	129	0.002936	186
F117	diagonal3	20	160	0.003784	218	279	0.005747	334	**	**	**
F118	bv	2	96	0.014769	146	446	0.015629	491	762	0.018572	807
F119	bv	4	334	0.011614	371	157	0.005304	210	878	0.018691	927
F120	pen2	2	851	0.043007	895	89	0.004203	137	125	0.004289	169

Accuracy

This section assesses the accuracy of the proposed algorithm by comparing the computed solutions against reference solutions obtained from other CG algorithms having similar characteristics. Table 4 summarizes the percentage of success and failure in the computed solutions for all the methods, demonstrating excellent agreement with the reference solutions.

TABLE 4. Percentage of success and failure for all algorithms

	smRMIL	mRMIL	RMIL
Success	100%	89.2%	86.7%
Failure	0 %	10.8%	13.3 %

Convergence speed

Using a performance profile tool introduced by Dolan and More (2002), the convergence behavior of the proposed CG algorithms is graphically illustrated for the considered set of unconstrained optimization problems. The performance profile tool is constructed to analyze the convergence behaviour of the CG methods for problems of different sizes and dimensions. Suppose S is the set of n_s solvers and P define the collections of n_p benchmark problems. This tool is therefore the performance metric (NOI, CPUT, or FEV) for solver $s \in S$ and problem $p \in P$, compared to those from other CG algorithms. The measure of convergence behavior and evaluation of performance is achieved using

$$r_{p,s} = \frac{f_{p,s}}{\min\{f_{p,s}: s \in S \text{ and } p \in P\}}.$$

The algorithm with the best performance will have its convergence curve lying at the top of the performance profile curve. The graphically presented results (see Figures 1 - 3) follow from the computational results demonstrated in Tables 1 - 3. The plot shows the reduction in the residual norm over iterations (3), CPU time (1), function evaluation (2) indicating rapid convergence to the solution.

From the left-hand side of Figure 1, the mRMIL method won 78% despite solving only solved 89.2% of the problems, followed by the RMIL method that wins 68% and solved 86.7% of the problems; however, the proposed method recorded 65% win despite solving all the problems successfully. The interpretation from Figure 2 show that the proposed smRMIL perform better on this metric with the win of about 88% followed by mRMIL with 80% respectively. On the other hand, RMIL is the least performer on this metric with 69% win. For the right-hand side of the graph, the results are similar and have been discussed above.

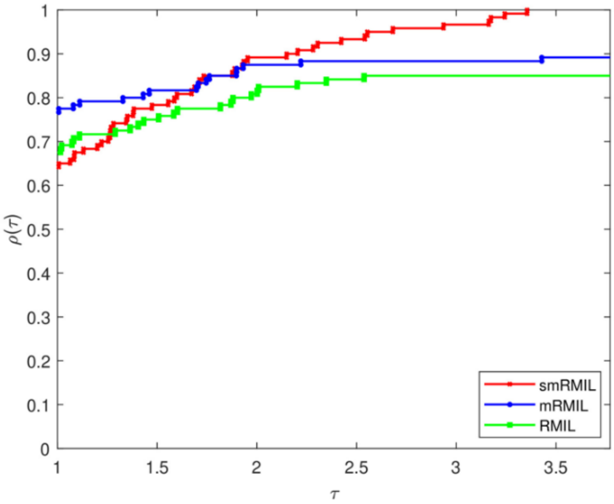

FIGURE 1. Performance profile showing comparison on CPU time in seconds of the test functions.

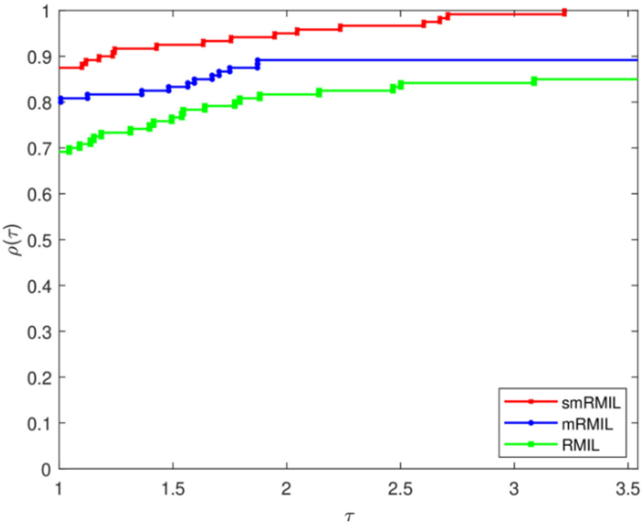

FIGURE 2. Performance profile showing comparison on number of function evaluation of the test functions.

On the other hand, the discussion on the right-hand side of Figure 3 follows from above and is thus omitted. However, based on the left-hand side of the plot, the smRMIL algorithm recorded

83% win followed by the mRMIL method with 75%. The other method, RMIL, is closely comparable in winning about 62% of the test problems, respectively.

Based on the summary of the results presented in Table 4, and the above discussion, it will suffice to say that the proposed smRMIL algorithm is not only efficient and robust but also very promising for the 120 benchmark problems considered.

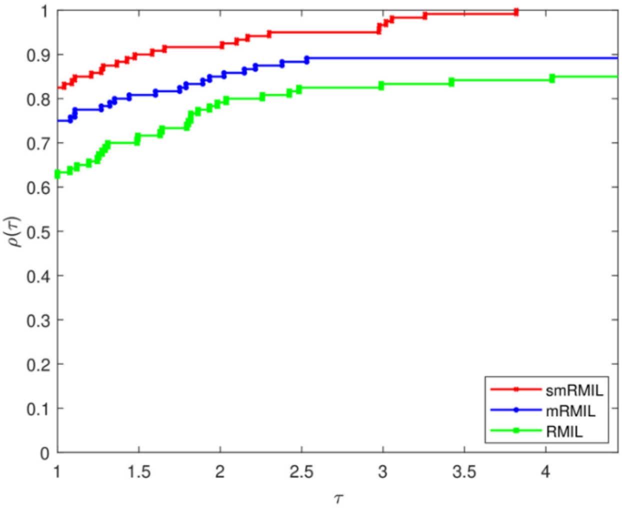

FIGURE 3. Performance profile showing comparison on iteration number of the test functions.

CONCLUSION

This study proposed a scaled conjugate gradient method (smRMIL) that integrates a quasi-Newton-derived scaling parameter to enforce sufficient descent without exact line searches. The matrix-free structure of the method ensures computational efficiency for large-scale unconstrained optimization, while theoretical guarantees under standard conditions enhance reliability. To validate the accuracy of smRMIL, 120 test problems are numerically tested, and the results confirm that smRMIL has a 100% convergence success rate, outperforming all the previous mRMIL and RMIL methods, and requires less iterations, function evaluations, and CPU time for high dimensional and nonconvex landscapes.

ACKNOWLEDGEMENT

The authors are grateful to the reviewers for their comments and suggestions which have improved the quality of the manuscript.

REFERENCES

Al-Baali, M. Descent property and global convergence of the Fletcher-Reeves method with inexact line search. *IMA J. Numer. Anal.,* 5 (1985), 121-124.

Amini K, Faramarzi P, & Pirfalah N.: A modified Hestenes-Stiefel conjugate gradient method with an optimal property, *Optimization Methods and Software,* (2018) DOI: 10.1080/10556788.2018.1457150.

Andrei N. Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization, *Optimization Methods and Software*, 22(4) (2007) pp. 561-571.

Awwal, A. M., Wang, L., Kumam, P., Sulaiman, M. I., Salisu, S., Salihu, N., & Yodjai, P. Generalized RMIL conjugate gradient method under the strong Wolfe line search with application in image processing. *Mathematical Methods in the Applied Sciences*, 46(16), 17544-17556 (2023).

Bojari, S., Eslahchi, M.R. Two families of scaled three-term conjugate gradient methods with sufficient descent property for nonconvex optimization. *Numer Algor* 83, 901–933 (2020).

Cheng, W. A two-term PRP-based descent method Numer. *Funct. Anal. Optim*. 28 (2007) 1217 - 1230.

Dai, Z. (2016). Comments on a new class of nonlinear conjugate gradient coefficients with global convergence properties. *Applied Mathematics and Computation*, 276, 297-300.

Dolan ED, and More JJ. Benchmarking optimization software with performance profiles, *Math. Program*. 91 (2002), pp. 201-213.

Gilbert, J. C. and Nocedal, J. *Global convergence properties of conjugate gradient methods for optimization, SIAM J. Optim*., 2 (1992) 21- 42.

Gould, N. I., Orban, D., & Toint, P. L. (2003). CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software* (TOMS), 29(4), 373-394.

Malik, M., Abas, S. S., Mamat, M., & Mohammed, I. S. (2020b). A new hybrid conjugate gradient method with global convergence properties. *International Journal of Advanced Science and Technology*, *29*(5), 199-210.

Malik, M., Mamat, M., Abas, S. S., & Sulaiman, I. M. (2020a). A new spectral conjugate gradient method with descent condition and global convergence property for unconstrained optimization. *J. Math. Comput. Sci.*, *10*(5), 2053-2069.

Mamat, M., Sulaiman, I. M., Maulana, M., & Zakaria, Z. A. (2020). An efficient spectral conjugate gradient parameter with descent condition for unconstrained optimization. *Journal of Advanced Research in Dynamical and Control Systems*, *12*(2), 2487-2493.

Mirhoseini, N., Babaie-Kafaki, S., and Aminifard, Z. *A Nonmontone scale Fletcher-Reeves conjugate gradient method with application in imahe reconstruction.* Math. Sci. Soc. 45 (2022) 2885-2904.

Mohammad, H., Sulaiman, I. M., & Mamat, M. Two diagonal conjugate gradient like methods for unconstrained optimization. *Journal of Industrial and Management Optimization*, 20(1), 170-187, (2024).

Mohammed, I. S., Mamat, M., Abashar, A., Rivaie, M., & Salleh, Z. (2015). The global convergence properties of an improved conjugate gradient method. *Applied Mathematical Sciences*, *9*(38), 1857-1868.

Omesa, A. U., Ibrahim, S. M., Yunus, R. B., Moghrabi, I. A., Waziri, M. Y., & Sambas, A. (2025). A brief survey of line search methods for optimization problems. *Results in Control and Optimization*, *19*, 100550.

Oren SS, and Luenberger DG. Self-scaling variable metric (SSVM) algorithms, Part I: Criteria and sufficient conditions for scaling a class of algorithms, *Manage Sci*. 20(5) (1974), pp. 845-862.

Polak E, and Ribire G. Note sur la convergence de methods de directions conjugees, *Rev. Franc Inform Recher Opiration* 16 (1969), pp. 35-43.

Polyak BT. The conjugate gradient method in extremal problems, *U.S.S.R Comput. Math. Math. Phys*. 9 (1969), pp. 94-112.

Rivaie, M. Mamat, L.W. June, and I. Mohd. A new class of nonlinear conjugate gradient coefficients with global convergence properties. *Applied Mathematics and Computation*, 218(22):11323–11332, (2012a).

Rivaie, M., Mamat, M., Leong, W. J., & Ismail, M. (2012b). A new conjugate gradient coefficient for large scale nonlinear unconstrained optimization. *Int. Journal of Math. Analysis*, 6(23), 1131-1146.

Sabi'u, J., Sulaiman, I. M., Kaelo, P., Malik, M., & Kamaruddin, S. A. An optimal choice Dai-Liao conjugate gradient algorithm for unconstrained optimization and portfolio selection. *AIMS Mathematics*, 9(1), 642-664 (2024).

Salihu, N., Kumam, P., Awwal, A. M., Sulaiman, I. M., & Seangwattana, T. (2023). The global convergence of spectral RMIL conjugate gradient method for unconstrained optimization with applications to robotic model and image recovery. *Plos one*, 18(3), e0281250.

Salihu, N., Kumam, P., Sulaiman, I. M., & Seangwattana, T. An efficient spectral minimization of the Dai-Yuan method with application to image reconstruction. *AIMS Mathematics*, 8(12), 30940-30962 (2023).

Sulaiman, I. M., Supian, S., & Mamat, M. (2019). New class of hybrid conjugate gradient coefficients with guaranteed descent and efficient line search. In *IOP Conference Series: Materials Science and Engineering* (Vol. 621, No. 1, p. 012021). IOP Publishing.

Touati-Ahmed, D. and Storey, C. Efficient hybrid conjugate gradient techniques, *Journal of Optim. Theory Appl.*, 64 (1990), 379 - 397.

Yuan, G. and Zhang, M. A three-term Polak-Ribiere-Polyak conjugate gradient algorithm for large-scale nonlinear equations. *Journal of Computational and Applied Mathematics* 286 (2015), 186-195.

Zhang, L., Zhou, W., and Li, D.H. A descent modified PolakRibire Polyak conjugate gradient method and its global convergence, *IMA J. Numer. Anal*. 26 (2006) 629 -640.