Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 7361092, 8 pages
https://doi.org/10.1155/2020/7361092

Research Article

On a Dual Direct Cosine Simplex Type Algorithm and Its Computational Behavior

Elsayed Badr [1,2] **and Sultan Almotairi** [3]

[1]*Scientific Computing Department, Faculty of Computers & Artificial Intelligence, Benha University, Benha, Egypt*
[2]*Higher Technological Institute, 10th of Ramadan City, Egypt*
[3]*Department of Natural and Applied Sciences, Community College Majmaah University, Al-Majmaah 11952, Saudi Arabia*

Correspondence should be addressed to Sultan Almotairi; almotairi@mu.edu.sa

Received 1 February 2020; Revised 4 April 2020; Accepted 6 April 2020; Published 11 May 2020

Academic Editor: Giuseppe D'Aniello

The goal of this paper is to propose a dual version of the direct cosine simplex algorithm (DDCA) for general linear problems. The proposed method has not artificial variables, so it is different from both the two-phase method and big-M method. Our technique solves the dual Klee–Minty problem via two iterations and solves the dual Clausen problem via four iterations. The power of the proposed algorithm is evident from the extensive experimental results on benchmark problems adapted from NETLIB. Preliminary results indicate that this dual direct cosine simplex algorithm (DDCA) reduces the number of iterations of the two-phase method.

1. Introduction

Linear programming plays an important role in the optimization theory. Many real-world problems can be formulated as linear or nonlinear mathematical models. The simplex method is the common tool for solving linear programs. It is an iterative method that was developed by Dantzig [1–3].

There are many pivot rules for the simplex-type algorithm such as the exterior point simplex algorithm [4–6], primal-dual exterior point algorithm [7], and max-out-in pivot rule [8]. It is known that the application of the simplex algorithm requires at least one basic feasible solution. On the other hand, the common techniques that are used for determining an initial feasible basis are the two-phase and big-M methods. The main drawback of these techniques lies in requiring the introduction of artificial variables and increasing the dimension of the problem. Corley et al. [9] introduced the cosine simplex algorithm for solving linear programs. Yeh and Corley [10] proposed a simple direct cosine simplex algorithm (DCA) which solves the Klee–Minty Problem [11] via two iterations. They deduced that their algorithm reduced the number of iterations of simplex in most cases in their experimental results. Li and Li [12] explained the relationship between the cosine pivot rule and the most-obtuse-angle pivot rule, proposed by Pan [13]. In this paper, we propose a dual version of a simple direct cosine simplex algorithm (DDCA) which solves the dual Klee–Minty class of problems via two iterations while the two-phase method solves this class in n+1 iterations where n is the size of the problem. Our technique also solves Clausen class of problems via four iterations, but the two-phase method solves this class in $2n$-1 iterations where n is the size of the problem. Our technique does not require the introduction of artificial variables.

The rest of the paper is organized as follows. Section 2 describes the proposed DDCA algorithm and its characteristics. Benchmark problems "Klee–Minty and Clausen problems" are presented in Section 3. In Section 4, we introduce illustrations of the proposed method with two examples. Computational experiments are proposed in Section 5. Finally, conclusions and future work are proposed in Section 6.

2. Dual Cosine Simplex Algorithm (DDCA)

We consider the linear programming (LP) problem in standard form:

$(P)\ \max\{b^T y\colon A^T y = c;\ y \geq 0\}$, where A is an $m \times n$ matrix, x and c are n-dimensional vectors, and T denotes transposition. The dual of (P) is the problem.

$(D)\ \min\{c^T x\colon Ax \geq b\}$,where y is an m-dimensional vector.

For constraint i of (D), define $\cos\theta_i = (\sum_{j\in N}(a_{ij}c_j)^2/\sum_{j\in N}(a_{ij})^2)$ as the cosine of angle θ_i between the constraints i and the objective function where $b_i < 0$ and N is the index set of the nonbasic variables (Algorithm1).

Remark 1. There is no proof for the correctness of the above cosine criteria. Hence, it is not true for ever.

3. Benchmark Problems

In this section, we present two well-known classes of linear programming problems, Klee–Minty class of problems [11] is the first problem and the other is Clausen class of problems [14] as illustrated in the following models:

$$\begin{aligned} &\max && \sum_{j=1}^{n} 10^{n-j}x_j \\ &\text{subject to} && 2\sum_{j=1}^{i-1} 10^{i-j}x_j + x_i \leq 100^{i-1} \\ &&& x_j \geq 0, \quad i = 1, 2, \ldots n, \\ &&& \text{Klee–Minty problem,} \end{aligned}$$

$$\begin{aligned} &\max && \sum_{j=1}^{n} \left(\frac{4}{5}\right)^{j} x_j, \quad x_1 \leq 1 \\ &\text{subject to} && 2\sum_{j=1}^{i-1} \left(\frac{5}{4}\right)^{i-j} x_j + x_i \leq 5^{i-1} \\ &&& x_j \geq 0, \quad i = 2, \ldots n, \\ &&& \text{Clausen problem.} \end{aligned} \tag{1}$$

Klee and Minty [11, 14] proved that the simplex algorithm is an exponential algorithm in 1972 (for the worst case). An interesting result is that the dual simplex method solves the Klee–Minty problem in a polynomial number of iterations. Another challenging exponential example is shown in [14]. The advantage of Clausen's example [14] is that the dual simplex is exponential on the dual problem, whereas the primal simplex method is exponential on the primal problem.

The following examples show the superiority of our technique over the two-phase method. Example 1 shows that the two-phase method requires 6 tableaus while our technique requires 3 iterations only, without including the initial one.

4. Illustrative Examples

Example 1. Consider the following Random Linear Programming Problem:

$$\begin{aligned} &\min && w = 4x_1 + x_2 \\ &\text{subject to:} && 3x_1 + x_2 \leq 3;\ 3x_1 + x_2 \geq 3;\ 4x_1 + 3x_2 \geq 6;\ x_1 \\ &&& +2x_2 \leq 4x_1, \quad x_2 \geq 0. \end{aligned} \tag{2}$$

The variables x_3 and x_6 are slack variables, but the variables x_4 and x_5 are the surplus variables for the corresponding constraints. We calculate $\cos\theta_i$ for every $i = 2, 3$ (in the first iteration) as follows:

$$\begin{aligned} &\cos\theta_1 = \#, \\ &\cos\theta_2 = \frac{[(-3)\times(-4) + (-1)(-1)]^2}{(-3)^2 + (-1)^2} = \frac{169}{10} = 16.9, \\ &\cos\theta_3 = \frac{[(-4)\times(-4) + (-3)(-1)]^2}{(-4)^2 + (-3)^2} = \frac{361}{25} = 14.44, \\ &\cos\theta_4 = \#. \end{aligned} \tag{3}$$

The value of $\cos\theta_2$ is bigger than the value of $\cos\theta_3$. Therefore, the variable x_4 is the leaving variable. From STEP 2, the entering variable is calculated as follows.

$\min\{|b_i/a_{ij}|\colon a_{ij} < 0 \text{ and } j \in N\} = \{|-3/-3|, |-3/-1|\} = 1$, and therefore, the element x_1 is the entering variable. We can construct a new simplex table by the pivoting operations (i.e., Step 3) as shown in Iteration 1 in Table 1. We repeat all above operations until all coefficients in Row 0 are nonpositive in Iteration 3, and $x_3 = 0$, $x_4 = 2/5$, $x_5 = 9/5$, and $x_6 = 1$ are optimal with $z = 17/5$ in the original problem.

Furthermore, the two-phase method requires 6 iterations, as shown in Table 2, without including the initial one.

Example 2. Dual Klee–Minty problem.

Consider the following dual Klee–Minty problem of size $n = 3$:

Require: infeasible basis
While $b_i < 0$
Step 1 (dual feasibility condition): let N is the index set of the nonbasic variables. The leaving variable, x_i, is the basic variable having the maximum $\cos\theta_i$ for minimization problem, where $\cos\theta_i = (\sum_{j\in N}(a_{ij}c_j)^2/\sum_{j\in N}(a_{ij})^2)$ is the angle between the constraint i and the objective function. The tie is broken by choosing the most negative value in the right hand side.
Step 2 (dual optimality condition): given that x_i is the leaving variable, the entering variable is the nonbasic variable $a_{ij} < 0$ that corresponds to $\min\{|b_i/a_{ij}|: a_{ij} < 0 \text{ and } j \in N\}$ The ties are broken arbitrary. If $a_{ij} \geq 0$ for all nonbasic variables, then the problem has no feasible solution.
Step 3: apply a pivoting
End while
The current basis is feasible
Apply the simplex algorithm.

Algorithm 1: Dual cosine simplex method (DCSM).

Table 1: The solution of Example 1 by the proposed DCSM.

Iteration		x_1	x_2	x_3	x_4	x_5	x_6	R.H.S
0	Z	−4	−1	0	0	0	0	0
	x_3	3	1	1	0	0	0	3
	x_4	−3	−1	0	1	0	0	−3
	x_5	−4	−3	**0**	0	1	0	−6
	x_6	1	2	**0**	0	0	1	4
1	Z	0	2	0	0	−1	0	6
	x_3	0	0	1	1	0	0	0
	x_4	1	1/3	0	−1/3	0	0	1
	x_5	0	−5/3	0	−4/3	1	0	−2
	x_6	0	5/3	0	1/3	0	1	3
2	Z	0	0	0	−8/3	1/5	0	18/5
	x_3	0	0	1	1	0	0	0
	x_4	1	0	0	−3/5	1/5	0	3/5
	x_5	0	1	0	4/5	−3/5	0	6/5
	x_6	0	0	0	−1	1	1	1
3	Z	0	0	0	−7/5	0	−1/5	17/5
	x_3	0	0	1	1	0	0	0
	x_4	1	0	0	−2/5	0	−1/5	2/5
	x_5	0	1	0	1/3	0	−3/5	9/5
	x_6	0	0	0	−1	1	1	1

Table 2: The solution of Example 1 by the two-phase method.

Iteration		x_1	x_2	x_3	x_4	x_5	R_1	R_2	x_6	R.H.S
0 Phase1	Z′	0	0	0	0	0	−1	−1	0	0
	x_5	3	1	0	0	1	0	0	0	3
	R_1	3	1	−1	0	0	1	0	0	3
	R_2	4	3	**0**	−1	0	0	1	0	6
	x_6	1	2	**0**	0	0	0	0	1	4
1 Phase1	Z′	7	4	−1	−1	0	0	0	0	9
	x_5	3	1	0	0	1	0	0	0	3
	R_1	3	1	−1	0	0	1	0	0	3
	R_2	4	3	0	−1	0	0	1	0	6
	x_6	1	2	0	0	0	0	0	1	4
2 Phase1	Z′	0	1.67	−1	−1	−2.33	0	0	0	2
	x_1	1	0.33	0	0	0.33	0	0	0	1
	R_1	0	0	−1	**0**	−1	1	0	0	0
	R_2	0	1.67	0	−1	−1.33	0	1	0	2
	x_6	0	0	0	0	−0.33	0	0	1	3

Table 2: Continued.

Iteration		x_1	x_2	x_3	x_4	x_5	R_1	R_2	x_6	R.H.S
3 Phase1	Z′	0	0	−1	0	−1	0	−1	0	0
	x_1	1	0	0	0.2	0.6	1	−0.2	0	0.6
	R_1	0	0	−1	0	−1	0	0	0	0
	x_2	0	1	0	−0.6	−0.8	0	0.6	0	1.2
	x_6	0	0	0	1	1	0	−1	1	1
4 Phase2	Z′	0	0	0	0.2	1.6	Blocked	Blocked	0	3.6
	x_1	1	0	0	0.2	0.6	0	−0.2	0	0.6
	R_1	0	0	−1	0	−1	1	0	0	0
	x_2	0	1	0	−0.6	−0.8	0	0.6	0	1.2
	x_6	0	0	0	1	1	0	−1	1	1
5 Phase 2	Z′	0	0	−1.6	0.2	0	Blocked	Blocked	0	3.6
	x_1	1	0	−0.6	0.2	0	0	−0.2	0	0.6
	x_5	0	0	1	0	1	1	0	0	0
	x_2	0	1	0.8	0.6	0	0	0.6	0	1.2
	x_6	0	0	−1	1	0	0	−1	1	1
6 Phase 2	Z′	0	0	−1.4	0	0	Blocked	Blocked	−0.2	3.4
	x_1	1	0	−0.4	0	0	0	−0.2	−0.2	0.4
	x_5	0	0	1	0	1	1	0	0	0
	x_2	0	1	0.2	1	0	0	0.6	0.6	1.8
	x_4	0	0	−1	0	0	0	−1	1	1

Table 3: The solution of Example 2 by the proposed DCSM.

Iteration		x_1	x_2	x_3	x_4	x_5	x_6	R.H.S
0	Z	−1	−10	−10000	0	0	0	0
	x_4	−1	−20	−200	1	0	0	−100
	x_5	0	−1	−20	0	1	0	−10
	x_6	0	0	**(−1)**	0	0	1	−1
1	Z	−1	−10	0	0	0	−10000	10000
	x_4	−1	−20	0	1	0	200	100
	x_5	0	−1	0	0	1	20	10
	x_3	0	0	1	0	0	−1	1

$$\begin{aligned} &\min && w = x_1 + 100x_2 + 10000x_3 \\ &\text{subject to :} && x_1 + 20x_2 + 200x_3 \geq 100;\ x_2 + 20x_3 \geq 10;\ 4x_3 \\ &&& \geq 1,\ x_1, x_2, \quad x_3 \geq 0. \end{aligned} \tag{4}$$

The variables x_4, x_5, and x_6 are the surplus variables for the corresponding constraints. We calculate the corresponding $\cos\theta_i$ in the Iteration 0 for every $i = 1, 2, 3$ as follows:

$$\begin{aligned} \cos\theta_1 &= \frac{[(-1)\times(-1) + (-20)(-100) + (-200)\times(-10000)]^2}{(-1)^2 + (-20)^2 + (-200)^2} = \frac{4.0008 \times 10^{12}}{40401} = 99027351.81, \\ \cos\theta_2 &= \frac{[(0)\times(-1) + (-1)(-100) + (-20)\times(-10000)]^2}{(0)^2 + (-1)^2 + (-20)^2} = \frac{4.004001 \times 10^{10}}{401} = 99850399, \\ \cos\theta_3 &= \frac{[(0)\times(-1) + (0)(-100) + (-1)\times(-10000)]^2}{(0)^2 + (0)^2 + (-1)^2} = \frac{10^8}{1} = 10^8. \end{aligned} \tag{5}$$

The value of $\cos\theta_3$ is bigger than the values of $\cos\theta_1$ and $\cos\theta_2$. We choose x_6 as the leaving variable. The entering variable is calculated as follows (STEP 2):

$$\min\left\{\left|\frac{b_i}{a_{ij}}\right| : a_{ij} < 0 \text{ and } j \in N\right\} = \left\{\left|\frac{-100}{-1}\right|, \left|\frac{-100}{-20}\right|, \left|\frac{-100}{-200}\right|\right\} = \frac{1}{2}. \tag{6}$$

Table 4: The solution of Example 2 by the two-phase method.

Iteration		x_1	x_2	x_3	x_4	x_5	x_6	R_1	R_2	R_3	R.H.S
0 Phase1	Z'	0	0	0	0	0	0	−1	−1	−1	0
	R_1	1	20	200	−1	0	0	1	0	0	100
	R_2	0	−1	−20	0	−1	0	0	1	0	10
	R_3	0	0	**(−1)**	0	0	−1	0	0	1	1
1 Phase1	Z'	1	21	221	−1	−1	−1	0	0	0	111
	R_1	1	20	200	−1	0	0	1	0	0	100
	R_2	0	1	20	0	−1	0	0	1	0	10
	R_3	0	0	1	0	0	−1	0	0	1	1
2 Phase1	Z'	−0.11	−1.10	0	0.11	−1	−1	−1.11	0	0	0.50
	x_3	0.01	0.10	1	−0.01	0	0	0.01	0	0	0.50
	R_2	−0.10	−1	0	**(0.10)**	−1	0	−0.10	1	0	0
	R_3	−0.01	−0.10	0	0.01	0	−1	−0.01	0	1	0.50
3 Phase1	Z'	0	−0.05	0	0	0.05	−1	−1	−1.05	0	0.50
	x_3	0	0.05	1	0	−0.05	0	0	0	0	0.50
	x_4	−1	−10	0	1	−10	0	−1	10	0	0
	R_3	0	−0.05	0	0	0.05	−1	0	−0.05	1	0.50
4 Phase1	Z'	0	0	0	0	0	0	−1	−1	−1	0
	x_3	0	0	1	0	0	−1	0	0	1	1
	x_4	−1	−20	0	1	0	−200	−1	0	200	100
	R_3	0	−1	0	0	1	−20	0	−1	20	10
5 Phase 2	Z'	−1	−100	0	0	0	-10^4	Blocked	Blocked	Blocked	10000
	x_3	0	0	1	0	0	−1	0	0	1	1
	x_4	−1	−20	0	1	0	−200	−1	0	200	100
	x_5	0	−1	0	0	1	−20	0	−1	20	10

Table 5: The Tableau obtained from the dual cosine and two-phases.

Size	Dual Klee–Minty problem		Dual Clausen problem	
	Dual cosine DDCA	Two-phase method	Dual cosine DDCA	Two-phase method
1	2	1	4	3
2	2	3	4	4
3	2	4	4	5
4	2	5	4	7
5	2	6	4	9
6	2	7	4	11
7	2	8	4	13
8	2	9	4	15
9	2	10	4	17
10	2	11	4	19

Table 6: Properties of 33 NETLIB problems.

Problem name	No. of nonzeros	Density	New number of constraints	New number of variables	Number of variables	Number of "≤" constrains	Number of "≥" constrains	Number of "=" constrains
adlittle	465	0.0856	56	97	97	40	1	15
afiro	88	0.10185	27	32	32	19	0	8
bandm	2659	0.01847	305	472	472	0	0	305
beaconfd	3476	0.07669	173	262	262	33	0	140
brandy	2150	0.03925	220	249	249	54	0	166
etamacro	2489	0.00547	400	688	688	183	125	354
fit1d	14,430	0.0134	24	1026	1026	1038	11	1
fit1p	10,894	0.00633	627	1677	1677	399	0	627
grow15	5665	0.00976	300	645	645	600	0	300
grow22	8318	0.00666	440	946	946	880	0	440
grow7	2633	0.02083	140	301	301	280	0	140
kb2	291	0.13649	43	41	41	21	15	16
lotfi	1086	0.02305	153	308	308	42	16	95

Table 6: Continued.

Problem name	No. of nonzeros	Density	New number of constraints	New number of variables	Number of variables	Number of "≤" constrains	Number of "≥" constrains	Number of "=" constrains
recipelp	752	0.0198	91	180	180	77	43	91
sc105	281	0.02598	105	103	103	60	0	45
sc205	552	0.01326	205	203	203	114	0	91
sc50a	131	0.05458	50	48	48	30	0	20
sc50b	119	0.04958	50	48	48	30	0	20
scagr25	2029	0.00862	471	500	500	146	25	300
scagr7	553	0.03062	129	140	140	38	7	84
scfxm1	2612	0.01732	330	457	457	143	0	187
scfxm2	5229	0.00867	660	914	914	286	0	374
scfxm3	7846	0.00578	990	1371	1371	429	0	561
scsd1	3148	0.05379	77	760	760	0	0	77
scsd6	5666	0.02855	147	1350	1350	0	0	147
sctap1	2052	0.01425	300	480	480	0	180	120
share1b	1182	0.0449	117	225	225	28	0	89
share2b	730	0.09626	96	79	79	83	0	13
shell	4900	0.00303	536	1775	1775	119	9	784
ship04l	8450	0.00992	402	2118	2118	40	8	354
ship04s	5810	0.00991	402	1458	1458	40	8	354
stair	3857	0.0186	356	467	467	153	0	698
stocfor1	474	0.0365	117	111	111	48	6	63
Sum	111,017	1.09377	8539	19,531	19,531	5453	454	7079
Average	3364.152	0.03315	258.758	591.849	591.849	165.242	13.7576	214.515
Max	14,430	0.13649	990	2118	2118	1038	180	784
Min	88	0.00303	24	32	32	0	0	1

Table 7: The classification of the benchmark problems according to the variable number range.

Variable number range	30–99	100–500	501–99	1000–1500	1501–1999	Over 2000
Number of problems	6	15	5	4	2	1

Table 8: A comparison between the two-phase method and the proposed DDCA.

Problem name	Iteration number						Difference in iteration number		
	DCA			Simplex					
	Phase I	Phase II	Phase I&II	Phase I	Phase II	Phase I&II	Phase I	Phase II	Phase I&II
adlittle	21	99	120	38	100	138	17	1	18
afiro	6	7	13	10	7	17	4	0	4
bandm	828	323	1151	1042	242	1284	214	−81	133
beaconfd	132	17	149	154	37	191	22	20	42
brandy	731	82	813	521	71	592	−210	−11	−221
etamacro	940	355	1295	944	423	1367	4	68	72
fit1d	52	1664	1716	94	1355	1449	42	−309	−267
fit1p	820	2288	3108	1441	1358	2799	621	−930	−309
grow15	285	205	490	303	485	788	18	280	298
grow22	425	245	670	443	704	1147	18	459	477
grow7	131	78	209	143	168	311	12	90	102
kb2	74	25	99	397	38	435	323	13	336
lotfi	208	164	372	126	77	203	−82	−87	−169
recipelp	300	6	306	299	28	327	−1	22	21
sc105	54	46	100	64	42	106	10	−4	6
sc205	118	110	228	128	115	243	10	5	15
sc50a	24	20	44	29	23	52	5	3	8
sc50b	32	14	46	37	21	58	5	7	12
scagr25	503	869	1372	639	218	857	136	−651	−515
scagr7	126	85	211	159	45	204	33	−40	−7
scfxm1	753	252	1005	802	211	1013	49	−41	8
scfxm2	1592	322	1914	1478	386	1864	−114	64	−50
scfxm3	1947	490	2437	2324	591	2915	377	1	378

Table 8: Continued.

Problem name	Iteration number						Difference in iteration number		
	DCA			Simplex					
	Phase I	Phase II	Phase I&II	Phase I	Phase II	Phase I&II	Phase I	Phase II	Phase I&II
scsd1	90	200	290	139	206	345	49	6	55
scsd6	216	184	400	170	447	617	−46	263	217
sctap1	453	161	614	705	163	868	252	2	254
share1b	352	224	576	363	158	521	11	−66	−55
share2b	125	50	175	112	27	139	−13	−23	−36
shell	795	264	1059	843	209	1052	48	−55	−7
ship04l	700	143	843	728	78	806	28	−65	−37
ship04s	488	106	594	499	58	557	11	−48	−37
stair	1019	323	1342	1203	265	1468	184	−58	126
stocfor1	81	12	93	90	29	119	9	17	26
Sum	14421	9433	23854	16467	8385	24852			
Average	437	285.848	722.848	499	254.091	753.091			
Max	1947	2288	3108	2324	1358	2915			
Min	6	6	13	10	7	17			

Therefore, the element x_3 is the entering variable. We can construct a new simplex table by the pivoting operations (i.e., STEP 3) as shown in Iteration 1 in Table 3. We repeat all above operations until all coefficients in Row 0 are nonpositive in Iteration 3, and $x_1 = 1$ and $x_2 = x_3 = 0$ are optimal with $z = 10^4$ in the original problem.

On the other hand, the two-phase method requires 5 iterations, as shown in Table 4, without including the initial one.

5. Computational Experiments

In this section, we present the computational results of the dual cosine simplex algorithm (DDCA) and two-phase method for dual Klee–Minty and dual Clausen classes of problems. We compare the number of iterations of the dual cosine simplex algorithm (DDCA) with the two-phase method. We used different tolerances to reduce the number of iterations for each benchmark problem. We used the two-phase method [3, 15–18] to evaluate the effectiveness of the proposed method. On the other hand, the two-phase method was used for the problems contain "$\geq$" constraints and/or equality constraints.

The programming language used was MATLAB v7.01 SP2 with default options. All codes were run under 64-bit Window 8.1 Operating System having Core (TM)i5 CPU M 460 @2.53 GHz, 4.00 GB of memory.

It is clear that the basic difference between the dual cosine simplex method (DCSM) and the two-phase method is that our technique does not involve artificial variables. From Table 5, the contribution of the proposed algorithm is to solve the Klee–Minty problem and Clausen problem with 2 and 4 iterations, respectively. On the other hand, the simplex method with two-phase method spends $O(n)$ iterations for these problems.

Table 6 characterizes 33 NETLIB test problems [19] were used in comparison to evaluate the effectiveness of the proposed methods. For simplicity, we converted the bounded variables and free variables into constraints. The accuracy rates of the solution obtained from the proposed algorithms were tested by LINGO software.

Table 6 contains 6 categories of the problems according to the range of variable numbers as shown in Table 7.

Table 6 contains the largest nonzero number, density, number of variables (after transferring sign constraints), number of constraints (after transferring sign constraints), "$\leq$" constraint number, "$\geq$" constraint number, and "=" constraint number.

In general, from Table 8, the contribution of the proposed algorithm is that DDCA is generally better than the two-phase method (22 problems vs. 11 problems). The details of our results are as follows:

(a) Six problems with the variable numbers 30–99: DDCA is better than the two-phase method (5 problems vs. one problem)

(b) Fifteen problems with the variable numbers 100–500: DDCA is better than the two-phase method (10 problems vs. 5 problems)

(c) Five problems with the variable numbers 501–999: DDCA is better than the two-phase method (4 problems vs. one problem)

(d) Four problems with the variable numbers 1000–1500: DDCA and two-phase methods are equal (2 problems vs. 2 problems)

(e) Two problems with the variable numbers 1501–1999: the two-phase method is better than DDCA (0 problems vs. 2 problems)

(f) One problem with the variable numbers over 2000: the two-phase method is better than DDCA (0 problems vs. 1 problem)

6. Conclusions

We proposed a dual version of the direct cosine simplex algorithm (DDCA) for general linear problems. The proposed method has not artificial variables, so it is different

from both the two-phase method and big-M method. Our technique solved the dual Klee–Minty problem via two iterations and solved the dual Clausen problem via four iterations. The power of the proposed algorithm is evident from the extensive experimental results on benchmark problems adapted from NETLIB. Preliminary results indicate that this dual direct cosine simplex algorithm (DDCA) reduces the number of iterations of the two-phase method. In future work, we can improve this work by using different algorithms [20–24] with other combinations between them.

Data Availability

All data and methods generated or used during the study are available within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors extend their appreciation to the Deanship of Scientific Research at Majmaah University for funding this work under the project number RGP-2019-29.

References

[1] G. B. Dantzig, "Maximization of a linear function of variables subject to linear inequalities," in *Activity Analysis of Production and Allocation*, T. C. Koopmans, Ed., pp. 339–347, John Wiley, Hoboken, NJ, USA, 1951.

[2] G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, USA, 1963.

[3] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, John Wiley, Hoboken, NJ, USA, 3rd edition, 2004.

[4] K. Paparrizos, "An exterior point simplex algorithm for (general) linear programming problems," *Annals of Operations Research*, vol. 46-47, no. 2, pp. 497–508, 1993.

[5] E. S. Badr, K. Paparrizos, N. Samaras, and A. Sifaleras, "On the basis inverse of the exterior point simplex algorithm," in *Proceedings of the 17th National Conference of Hellenic Operational Research Society (HELORS)*, pp. 677–687, Rio, Greece, June 2005.

[6] E. S. Badr, K. Paparrizos, B. Thanasis, and G. Varkas, "Some computational results on the efficiency of an exterior point algorithm," in *Proceedings of the 18th National Conference of Hellenic Operational Research Society (HELORS)*, pp. 1103–1115, Rio, Greece, June 2006.

[7] N. Samaras, A. Sifelaras, and C. Triantafyllidis, "A primal-dual exterior point algorithm for linear programming problems," *Yugoslav Journal of Operations Research*, vol. 19, no. 1, pp. 123–132, 2009.

[8] M. Tipawanna and K. Sinapiromsaran, "Max-out-in pivot rule with Dantzig's safeguarding rule for the simplex method," in *Proceedings of the 2nd International Conference on Mathematical Modeling in Physical Sciences*, Prague, Czech Republic, September 2013.

[9] H. W. Corley, J. Rosenberger, W. C. Yeh, and T. K. Sung, "The cosine simplex algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 27, no. 9-10, pp. 1047–1050, 2006.

[10] W.-C. Yeh and H. W. Corley, "A simple direct cosine simplex algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 178–186, 2009.

[11] V. Klee and G. Minty, "How good is the simplex algorithm?," in *Inequalities–III*, O. Shisha, Ed., pp. 159–175, Academic Press, Cambridge, MA, USA, 1972.

[12] W. Li and H. Li, "On simplex method with most-obtuse-angle rule and cosine rule," *Applied Mathematics and Computation*, vol. 217, no. 20, pp. 7867–7873, 2011.

[13] P.-Q. Pan, "Practical finite pivoting rules for the simplex method," *OR Spektrum*, vol. 12, no. 4, pp. 219–225, 1990.

[14] J. Clausen, "A tutorial note on the complexity of the simplex algorithm," Technicla Report NR79/16, DIKU, Copenhagen, Denmark, 1979.

[15] K. G. Murty, *Linear Programming*, John Wiley & Sons, Hoboken, NJ, USA, 1983.

[16] R. Vanderbei, *Linear Programming: Foundations and Extensions*, Kluwer Academic Publishers, Boston, MA, USA, 2nd edition, 2001.

[17] P. E. Gill, W. Murray, and M. H. Wright, *PracticalOptimization*, Academic Press, Cambridge, MA, USA, 1981.

[18] F. S. Hiller and G. J. Lieberman, *Introduction to Operations Research*, McGraw-Hill, New York, NY, USA, 6th edition, 1995.

[19] http://www.netlib.org/lp/data.

[20] E. M. Badr and M. I. Moussa, "An upper bound of radio k-coloring problem and its integer linear programming model," *Wireless Networks*, 2019.

[21] E. Badr and K. Aloufi, "A robot's response acceleration using the metric dimension problem," 2019, https://www.preprints.org/manuscript/201911.0194/v1.

[22] M. A. Fahmy, "Shape design sensitivity and optimization for two-temperature generalized magneto-thermoelastic problems using time-domain DRBEM," *Journal of Thermal Stresses*, vol. 41, no. 1, pp. 119–138, 2018.

[23] M. A. Fahmy, "Shape design sensitivity and optimization of anisotropic functionally graded smart structures using bicubic B-splines DRBEM," *Engineering Analysis with Boundary Elements*, vol. 87, pp. 27–35, 2018.

[24] E. M. Badr and H. elgendy, "A Hybrid water cycle - particle swarm optimization for solving the fuzzy underground water confined steady flow," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, 2020.