

A New Method of Solving the Bernoulli Quadrisection Problem: Diagrams

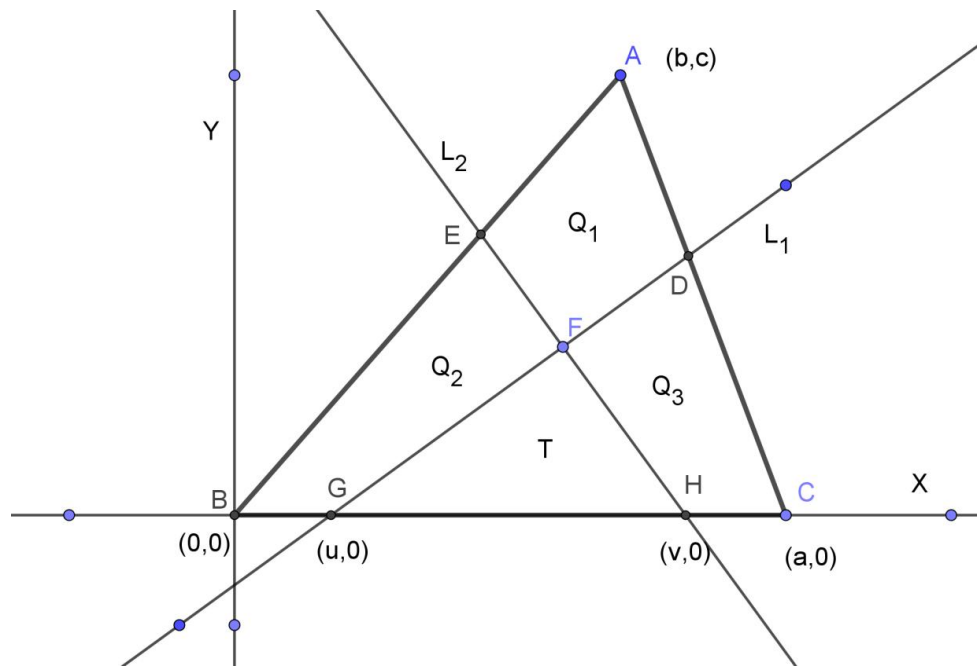


Figure 1

	A	B	C	D	E
1	Bernoulli Quadrisection Problem Using Particle Swarm Optimisation				
2					
3	Inputs				
4	Population Size =	40			
5	Cut-off Error =	0.00000001			
6	Maximum Position =	10			
7	Maximum Velocity =	0.2			
8	Maximum Iterations =	1000			
9	Initial Inertia Weight =	0.9			
10					
11	Outputs				
12	Iterations =	659			
13	Final Error =	8.87E-09			
14	u =	0.86196958032996			
15	v =	7.63913848328345			
16	m =	0.58391091828018			
17	T =	9.9999999079320			
18	Q1 =	9.9999999797039			
19	Q2 =	10.00000002139910			
20	Q3 =	9.99999998983728			

Figure 2

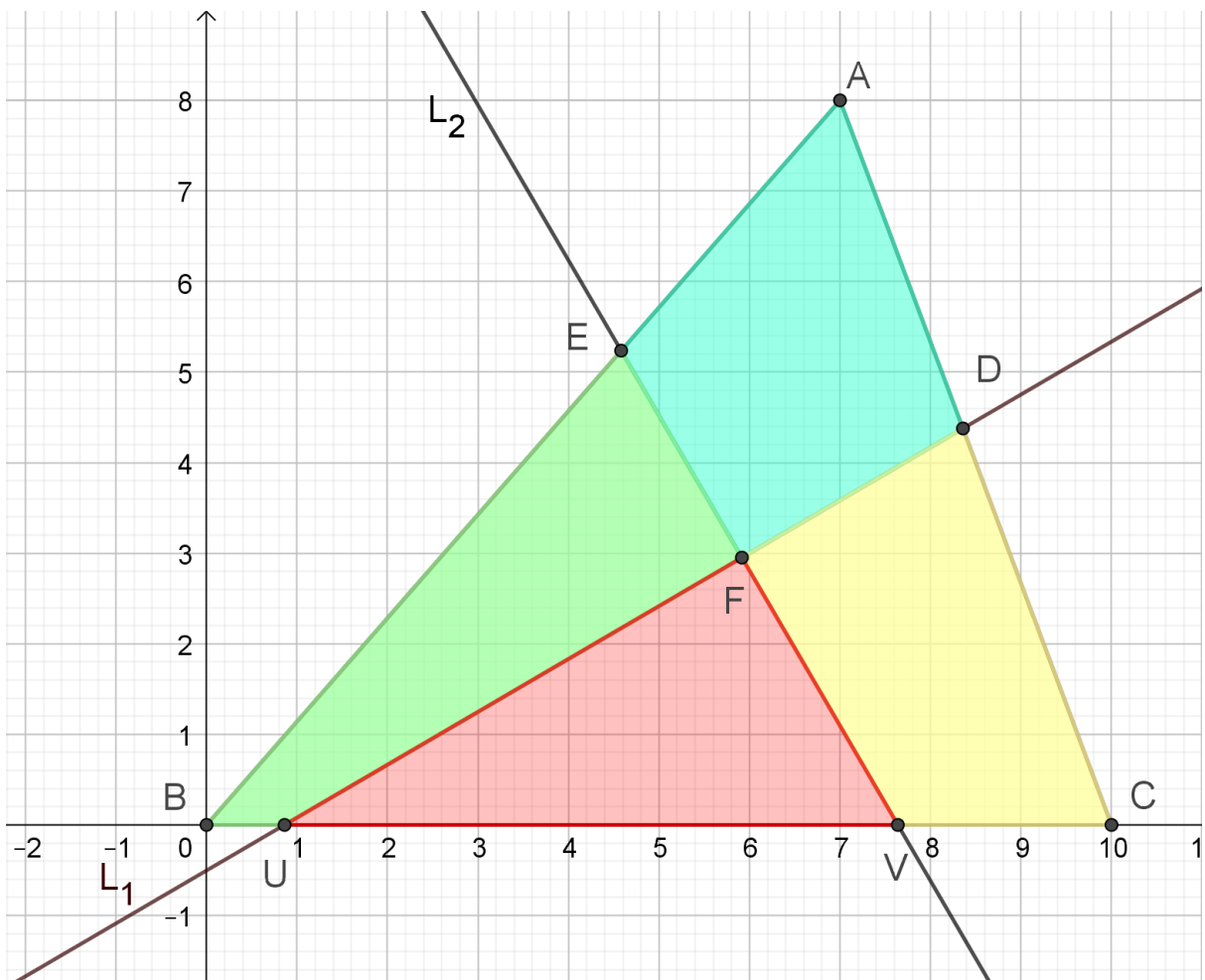


Figure 3 was drawn using the dynamic geometry software program Geogebra. It shows the quadrisection generated by the results obtained in **Figure 2** for a triangle whose vertices are (7,8), (0,0) and (10,0), and whose area is 40 units.

Figure 3

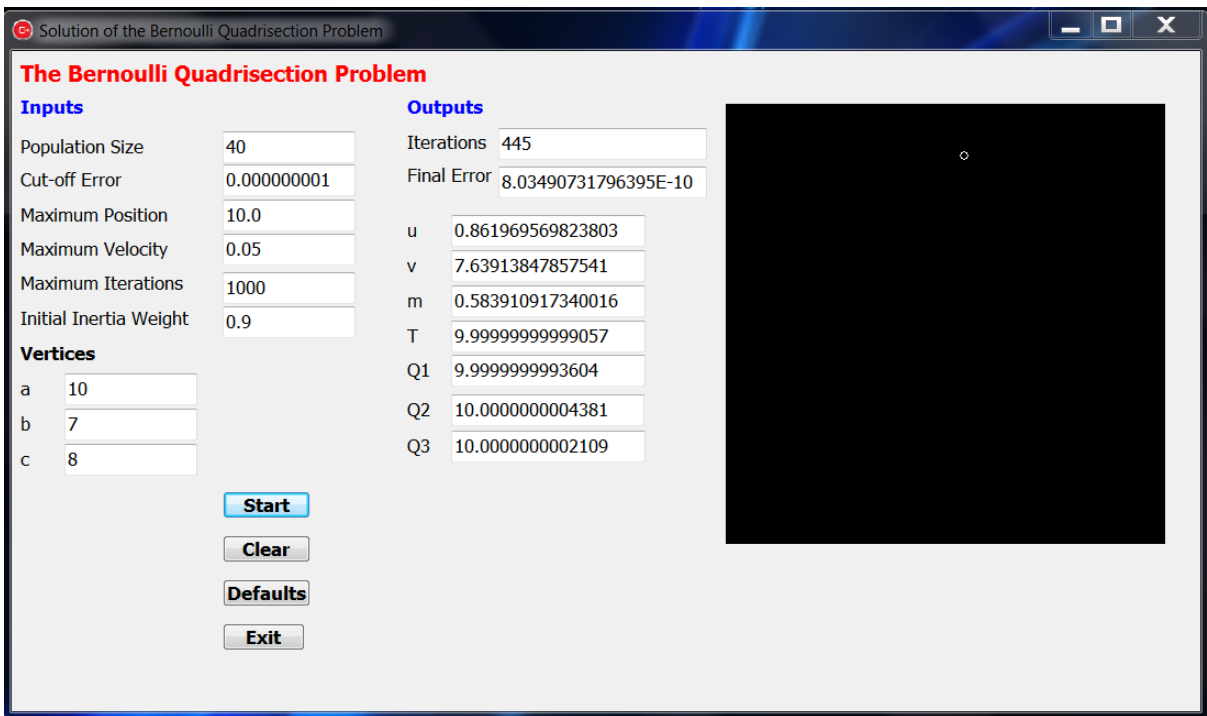


Figure 4

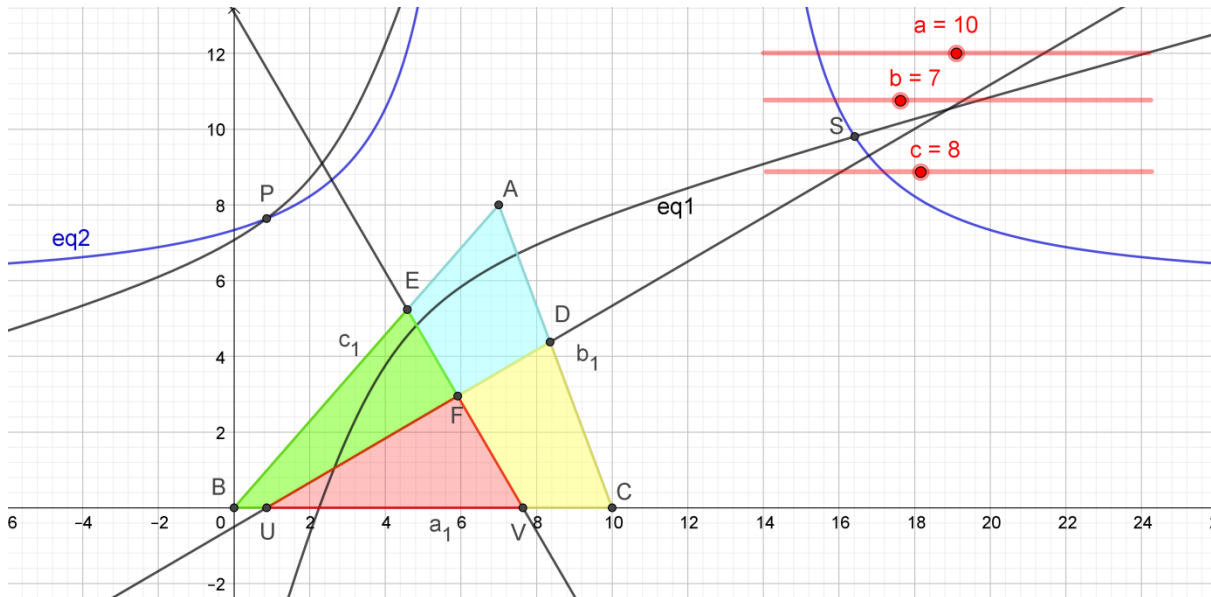


Figure 5

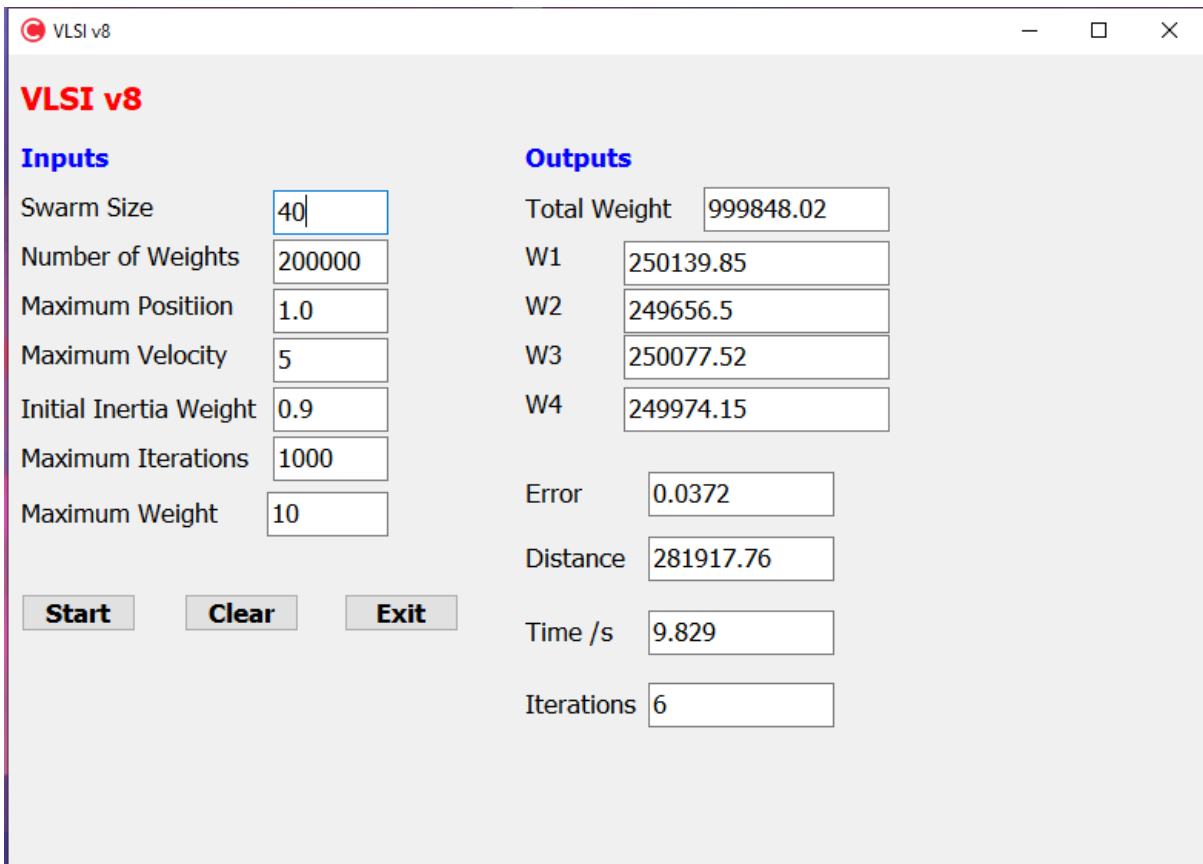


Figure 6

# Weights	Time	Iterations	Error	Distance
20000	1.4811	10	0.0353	29514.66
40000	3.0626	10	0.0399	58900.98
60000	4.4438	10	0.0368	87986.05
80000	5.2032	8	0.0310	116233.30
100000	6.5437	8	0.0316	145273.32
120000	7.8562	8	0.0275	174716.35
140000	8.5593	8	0.0328	203034.79
160000	9.5345	7	0.0325	230432.22
180000	10.2125	7	0.0351	257866.14
200000	11.9561	7	0.0272	288373.60

Figure 7

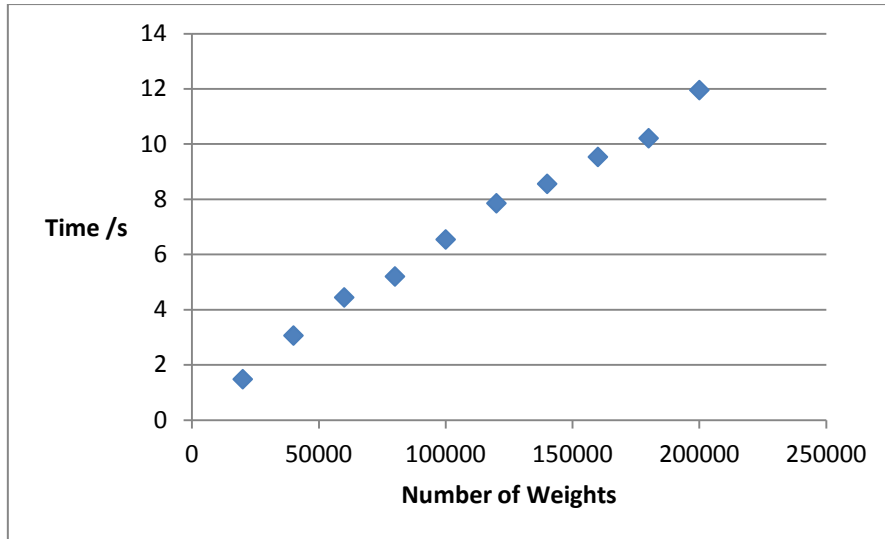


Figure 8

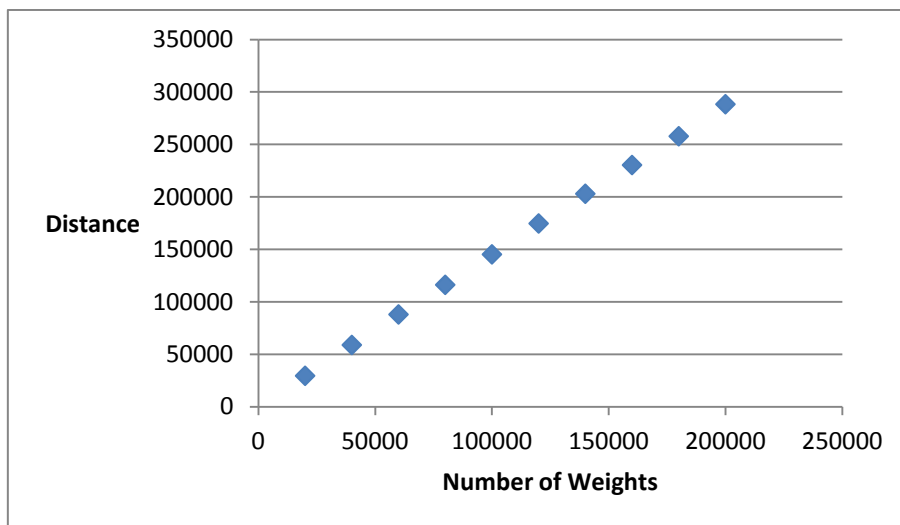


Figure 9

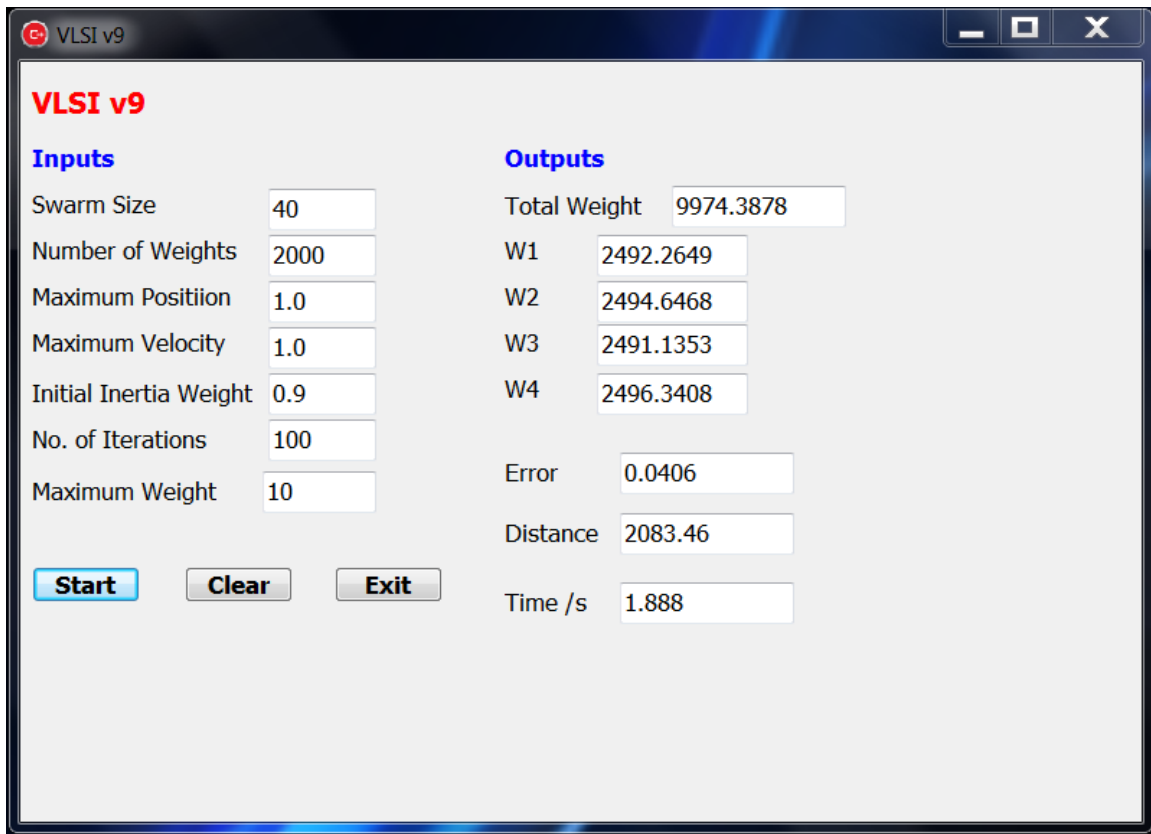


Figure 10

x	y
0.866125	0.696543
-0.362531	-0.0908504
-0.546766	0.842492
-0.566875	0.427543
0.427945	-0.330454
0.353755	0.608591
0.706225	0.53765
-0.518675	0.0150212
-0.822896	-0.575611
-0.64001	-0.662542
0.714867	0.458283
-0.681581	-0.545982
0.950956	0.557209
-0.554324	0.552609
0.544606	0.519229
0.891398	-0.033471
0.602659	-0.922454
-0.533678	-0.566488
-0.516178	-0.563881
-0.515878	0.715676

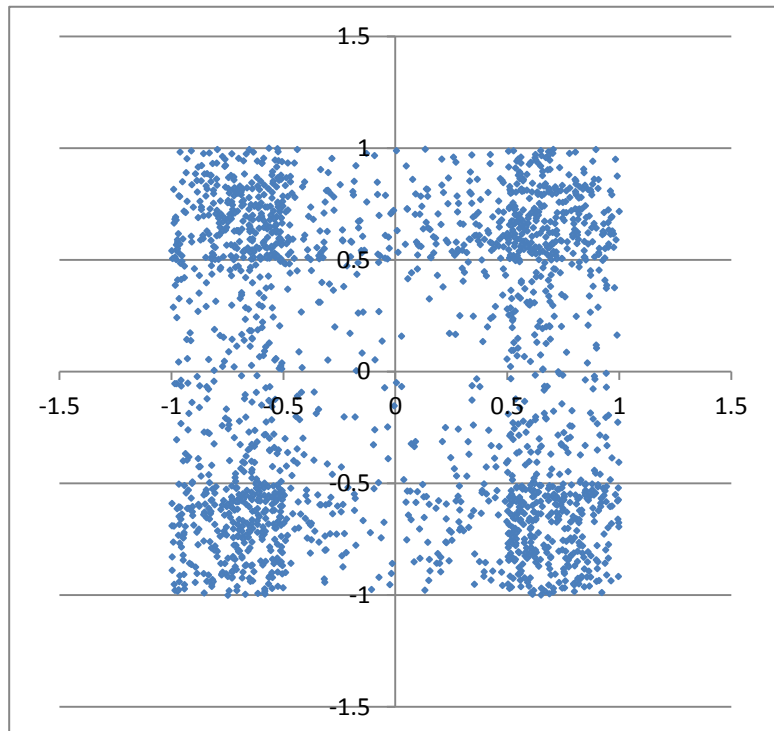


Figure 11

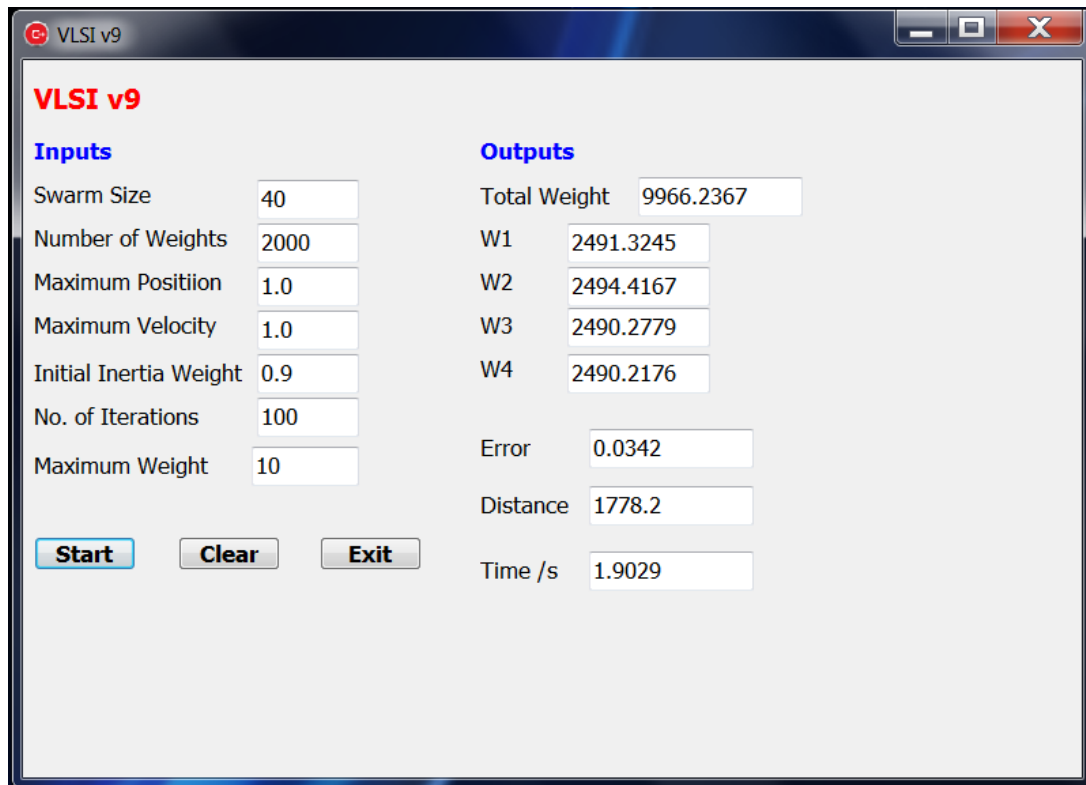


Figure 12

x	y
0.554093	0.215005
0.235447	-0.285745
0.732627	0.265285
-0.520411	-0.0418468
-0.296392	-0.428785
0.502777	0.367115
-0.827124	-0.306607
0.484136	-0.455505
-0.548789	-0.788108
-0.370283	-0.251725
-0.417125	-0.0320054
-0.396068	0.0127269
-0.547771	-0.0782005
0.705542	0.0198003
-0.567009	0.279903
0.293016	-0.555232
-0.199166	-0.389906
-0.216125	0.330464
-0.210213	-0.409847
-0.426938	0.410077

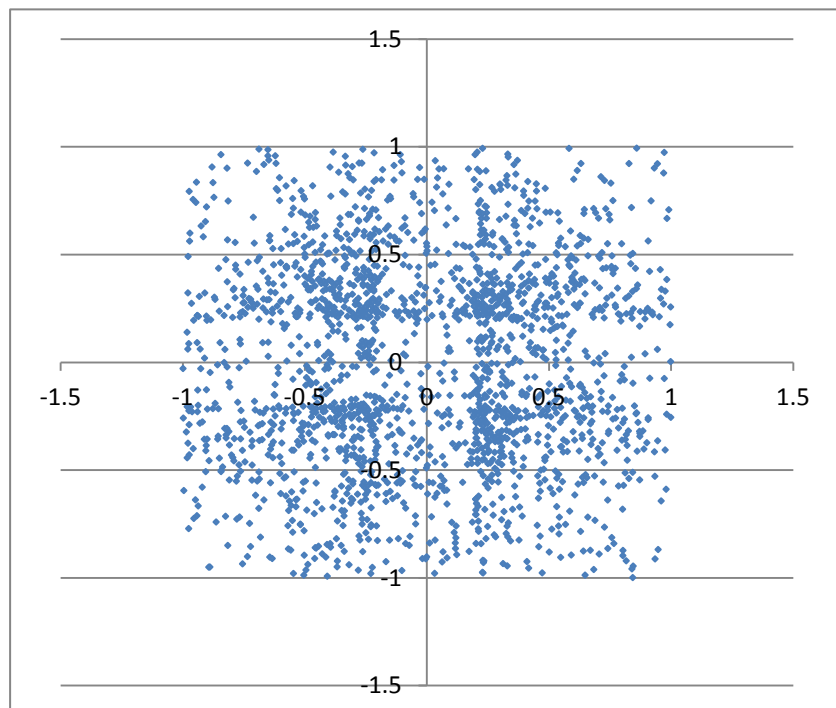


Figure 13

Schaffer F6 Function: VCSC Test

Inputs			Iterations
Population Size =	40		52
Cut-off Error =	1.00E-06		61
Maximum Position =	2		72
Maximum Velocity =	0.5		53
Maximum Iterations =	5000		56
Initial Inertia Weight =	0.9		54
			56
			74
Outputs			86
Iterations =	52		52
Error =	9.29E-07	Average =	62

The Schaffer F6 function is defined as $f(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2}$. Its global minimum value is 0 and this occurs at (0,0). All of the tests using the Schaffer F6 function shown in **Figure 16** and **Figure 18** have the same inputs as in **Figure 14**, with the exception of the cut-off error which can be either 1.00E-06 or 1.00E-10. The number of iterations required to achieve the specified accuracy for ten successive test runs are shown.

Figure 14

Griewank Function: VCSC Test

Inputs			Iterations
Population Size =	40		48
Cut-off Error =	1.00E-06		74
Maximum Position =	2		73
Maximum Velocity =	0.2		61
Maximum Iterations =	5000		64
Initial Inertia Weight =	0.9		70
			61
			72
Outputs			58
Iterations =	69		69
Error =	2.28E-07	Average =	65

The Griewank function in n dimensions is $f(x_1, \dots, x_n) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$. The global minimum is zero and occurs at the origin. We used the three-dimensional Griewank function in all of our tests. All of the tests using the Griewank function shown in **Figure 17** and **Figure 19** have the same inputs as in **Figure 15**, with the exception of the cut-off error which can be either 1.00E-06 or 1.00E-10.

Figure 15

Schafer F6 Function

Cut-off error = 1.0×10^{-6}

Test No.	PSO	PSO-SPSA	VCSC	VCSC-SPSA
1	1675	133	52	53
2	418	37	61	73
3	155	98	72	53
4	211	152	53	55
5	4705	94	56	68
6	647	74	54	73
7	214	97	56	60
8	3816	131	74	68
9	261	140	86	38
10	4211	135	52	65
Average =	1631	109	62	61

In this table the performance of the standard PSO algorithm is compared with that of the hybrid PSO-SPSA algorithm, the VCSC algorithm, and the hybrid VCSC-SPSA algorithm by recording the number of iterations required in ten successive test runs to achieve the desired accuracy. In each test run, the program is terminated if the value of the objective function is less than 1.0×10^{-6} .

Figure 16

Griewank Function

Cut-off error = 1.0×10^{-6}

Test No.	PSO	PSO-SPSA	VCSC	VCSC-SPSA
1	799	137	48	47
2	783	169	74	57
3	986	141	73	56
4	984	164	61	56
5	1576	46	64	65
6	1542	110	70	61
7	1332	90	61	64
8	1251	190	72	60
9	1543	176	58	65
10	1738	219	69	69
Average =	1253	144	65	60

In this table we compare the performance of the four algorithms using the Griewank function.

Figure 17

Schaffer F6 Function

Cut-off error = 1.0×10^{-10}

Test No.	PSO	PSO-SPSA	VCSC	VCSC-SPSA
1	>5000	1014	141	158
2	>5000	830	152	140
3	>5000	985	140	120
4	>5000	963	142	142
5	>5000	806	121	142
6	>5000	825	156	135
7	>5000	876	135	148
8	>5000	932	142	133
9	>5000	1031	124	157
10	>5000	882	121	139
Average =	>5000	914	137	141

In these tests, using the Schaffer F6 function, the standard PSO algorithm was unable to achieve the desired accuracy even once in less than 5000 iterations. In each test run, the program is terminated if the value of the objective function is less than 1.0×10^{-10} . The superior performance of the VCSC algorithm compared to the PSO-SPSA algorithm is even more evident than in **Figure 16** and **Figure 17**.

Figure 18

Griewank Function

Cut-off error = 1.0×10^{-10}

Test No.	PSO	PSO-SPSA	VCSC	VCSC-SPSA
1	>5000	1018	121	137
2	>5000	1341	115	137
3	>5000	1142	125	149
4	>5000	1014	113	141
5	>5000	1215	121	127
6	>5000	1173	122	134
7	>5000	987	111	125
8	>5000	1244	110	158
9	>5000	1074	118	153
10	>5000	1119	110	146
Average =	>5000	1133	117	141

In these tests, using the Griewank function, the VCSC algorithm performs best. We used the Schaffer F6 and Griewank functions for our tests because they are well-known standard test functions and are widely used for assessing the efficiency of optimisation algorithms.

Figure 19