Wireless Engineering and Technology, 2021, 12, 53-82
https://www.scirp.org/journal/wet
ISSN Online: 2152-2308
ISSN Print: 2152-2294

Central Force Optimization with Gravity < 0, Elitism, and Dynamic Threshold Optimization: An Antenna Application, 6-Element Yagi-Uda Arrays

Richard A. Formato

Consulting Engineer & Registered Patent Attorney of Counsel, Emeritus Cataldo & Fisher, LLC Woburn, MA, USA
Email: rf2@ieee.org

How to cite this paper: Formato, R.A. (2021) Central Force Optimization with Gravity < 0, Elitism, and Dynamic Threshold Optimization: An Antenna Application, 6-Element Yagi-Uda Arrays. *Wireless Engineering and Technology*, **12**, 53-82.
https://doi.org/10.4236/wet.2021.124004

Received: September 18, 2021
Accepted: October 26, 2021
Published: October 29, 2021

Abstract

This paper investigates the effect of adding three extensions to Central Force Optimization when it is used as the Global Search and Optimization method for the design and optimization of 6-elementYagi-Uda arrays. Those extensions are *Negative Gravity*, *Elitism*, and *Dynamic Threshold Optimization*. The basic CFO heuristic does not include any of these, but adding them substantially improves the algorithm's performance. This paper extends the work reported in a previous paper that considered only negative gravity and which showed a significant performance improvement over a range of optimized arrays. Still better results are obtained by adding to the mix *Elitism* and *DTO*. An overall improvement in best fitness of 19.16% is achieved by doing so. While the work reported here was limited to the design/optimization of 6-element Yagis, the reasonable inference based on these data is that any antenna design/optimization problem, indeed any Global Search and Optimization problem, antenna or not, utilizing Central Force Optimization as the Global Search and Optimization engine will benefit by including all three extensions, probably substantially.

Keywords

Yagi, Yagi-Uda, Array, Antenna, Antenna Design, Optimization, Central Force, Central Force Optimization, CFO, CFO-GED, Negative Gravity, Elitism, Dynamic Threshold Optimization, DTO, Dynamic, Threshold, Metaheuristic, Evolutionary Computation

1. Introduction

In a previous paper, [1], six element Yagi-Uda arrays (Yagis) were optimally designed using Central Force Optimization (CFO) with a small measure of pseudo randomly injected negative gravity (see [1] for details and array geometry). CFO searches for the maxima of an objective function, not its minima. The effect of negative gravity is to improve CFO's exploration of the decision space (DS) by causing probes that otherwise would coalesce around discovered maxima to disperse away from those maxima and sample under-sampled regions or perhaps regions that were not sampled at all. Based on the results reported in [1] there is no question that all CFO implementations and applications likely will benefit from some measure of negative gravity. At a minimum the algorithm's performance should be investigated with this added extension.

Negative gravity, however, is not the only improvement that can be made. This note reports the results of adding to CFO with negative gravity two additional extensions: *Elitism* [2] and *Dynamic Threshold Optimization* (*DTO*) [3], each of which improves CFO's performance still further. CFO is an evolutionary algorithm (EA) that explores DS in a series of "time steps" using "probes" to sample the objective function. *Elitism* comes in two flavours: 1) *Step-by-step Elitism* that involves in a given run storing step-by-step, the DS coordinates for a specified percentage of the best probes' locations and inserting those data into the algorithm's next step, thereby preserving best fitness information at each step. For the work reported various percentages of the best fitnesses' data from each step were pseudo randomly inserted into the next step. 2) *Run-to-Run Elitism*, also referred to as *Seed Probe Elitism*, involves passing the coordinates of the best fitness probe(s) from a previous run to the current CFO run. For the work reported here the coordinates of the best fitness probe in the previous run are saved in an external "seed probe" data file and inserted into the last of CFO's probes in the current run.

Dynamic Threshold Optimization is a purely geometrical method that modifies the objective function's topology on successive runs of *any* global search and optimization program (GSO, referring to both a program or to the heuristic, context permitting). *DTO* compresses the objective function's "landscape" from below by filtering out all local maxima that are below a *threshold* value that has been established at the run's start. *DTO* can be applied to *any* GSO algorithm, even different algorithms on successive runs. As the *DTO* threshold rises, more and more local maxima are removed so that the landscape becomes "flatter" and flatter. This, in turn, may (likely will) make it harder to explore DS because there is less topological information to guide the search. One way to address this issue is to increase the number of sample points during successive applications of the algorithm (in CFO's case, the number of probes). Besides step*-by-step elitism* within a CFO run, in this paper *DTO* is implemented with *run-to-run elitism* as well. That is, the best global maximum returned by a CFO run is passed on to the next CFO run within the *DTO* shell by pseudo randomly inserting it into the next run's initial probe distribution (IPD). As with *step-by-step elitism*, *run-by-run elitism* prevents loss of information that may be, almost certainly will be, useful in searching for maxima.

It is important to note that both *elitism* and *DTO* do not preserve the best fitness step-to-step or run-to-run. The previous best fitness information, specifically its coordinates in DS, are used only to guide the search. At any given step or at the end of any run the best fitness may actually be lower than the value inserted using *elitism* or *DTO*. Of course, preservation of the global best fitness is possible, but that approach was not taken for the results reported here because doing so may actually impede the GSO's exploration.

2. Methodology

2.1. CFO-GED Algorithm

CFO is a deterministic metaheuristic. It returns the same results every time it is run with the same setup. This characteristic is a major advantage when trying to develop either a truly useful objective function, or appropriate runtime parameters for real-world problems like Yagi array design. By contrast, if a stochastic optimizer is used, it is difficult, maybe impossible, to determine for sure whether or not a change in the objective function or a change in runtime parameters accounts for the different, sometimes wildly different, results on successive runs [4].

However, even though CFO is deterministic it does benefit from the addition of a pseudo random component because doing so improves its exploration. Both in [1] and in the work reported here pseudo randomness was injected using BBP-generated π-fractions. Calculation and use of the π-fractions are discussed in [5], and details of CFO theory and its implementation are contained in [6] (note important discussion on equations of motion in §VI-B of [6]). The essential characteristics of pseudo random variables (prv's) are that they are uncorrelated with the fitness function's topology, they are uniformly distributed, and they are uncorrelated among themselves. π-fractions meet these requirements. A prv is specified by enumeration or by calculation, and thus always remains deterministic. By contrast, a true random variable (rv) is *a priori* unknowable because it must be calculated from a probability distribution. This difference is fundamental and important, especially when it comes to solving real-world problems with metaheuristic methods. Thus, by using prv's CFO remains deterministic at every step while including some "randomness" that improves its exploration.

A major advantage that CFO provides is its repeatability. Stochastic GSO's generate results that vary from one run to the next, and there is no way of knowing why they are different. Was it a change in the run parameters? Was is a change in the objective function? Or was it the program's inherent randomness? Real-world problems do not have built-in objective functions. A suitable function must be developed for each new problem, and doing so can be quite difficult if the GSO's results are completely unpredictable. CFO by contrast permits rapid evaluation of changes in run parameters or in the objective function itself because every run with the same setup returns the same results. Any change in CFO's output is a result of changes in the program setup or in the fitness function.

The GSO algorithm used throughout this paper is basic CFO augmented with the three extensions not contained in that basic algorithm: 1) *Negative Gravity*; 2) *Elitism*; and 3) *Dynamic Threshold Optimization*. The new algorithm is called ***CFO-GED***, and its pseudo code appears in **Figure 1**. The effect of *negative gravity*, denoted $G < 0$ where G is CFO's "gravitational constant," is discussed extensively in [1]. *Elitism* involves storing the best fitness(es) from a previous GSO step or run and inserting those data into the next step or run. As discussed above, the first is referred to as *step-by-step* and the second as *run-by-run*, and it is important to note their differences because they produce different results. Both are used in CFO-GED. In *run-by-run* the *single* best fitness/coordinate data are passed on to the next run, whereas in *step-by-step* elitism a *user-specified number* of best fitnesses and locations are passed one step to the next (details in **Table 1**).

```
Alg: CFO-GED
DTO Shell
For # DTO Passes
   (a)Set $T_k^*$
   (b)Set $N_p$, $N_t$
   (c)Call GSO** (in this case CFO)
CFO Segment
      Initialization (Step 0)*
      (i)     Initialize π-fraction index (← 2)
      (ii)    Compute IPD***
      (iii)   Elitism (run-by-run): insert previous run best
              fitness coordinates into probe #1
      (iv)    Compute Initial Fitness Matrix
      (v)     Sort fitnesses with corresponding probes #'s
      (vi)    Initial Probe Accelerations (← 0)
      (vii)   Initialize Repositioning Factor Frep
      For Steps [1 - #Steps]
      (i)     Negative Gravity: Pseudo randomly assign $G<0$, $G>0$
      (ii)    Compute Probe Position Vectors
      (iii)   Elitism (step-by-step): pseudo randomly inject (#) Elitist
              probes from previous step into (#) probes this step
      (iv)    Retrieve Errant Probes
      (v)     Compute Fitness Matrix Current Probe Distribution
      (vi)    Sort fitnesses/probe #'s at this step
      (vii)   Compute Accelerations from Current Probe
              Distribution/Fitnesses
      (viii)  Increment Frep
      Next
Next

* see Appendix   ** any GSO, not necessarily same one each call
*** all pseudo random variables are computed using π-fractions
```

Figure 1. CFO-GED pseudo code.

Table 1. CFO-GED Yagi fitnesses with $G < 0$ and elitism (no DTO).

Yagi	N_p	% $G<0$		For Comparison to Data Below-Best Yagi from Reference [1], $G < 0$ Only			
		Target	Actual	F_{max}	g_{max}	Z_0	S ($VSWR//Z_0$)
[1]	20	5	6.0	49.189	11.92 dBi	59.80 Ω	1.49/2.01

Data with $G < 0$ & Two Types of Elitism Only (no DTO)

Yagi #	N_p	% $G<0$		*Elitism: Step-by-Step Only*						*Elitism: Step-by-Step & Seed Probe*			
		Target	Actual	% N_p	# *pr*	F_{max}	g_{max}	VZ_0	S	F_{max}	g_{max}	VZ_0	S
1	20	5	5.6	5	1	52.164	12.17	50.15	1.27/1.82	52.704	12.55	69.51	1.31/1.93
2	20	5	6.1	10	2	46.806	11.30	51.69	1.29/1.93	52.068	12.79	56.19	1.18/1.92
3	20	5	5.4	15	3	45.433	11.52	42.79	1.66/2.30	51.621	12.42	54.49	1.52/1.64
4	20	5	4.8	20	4	44.912	11.59	45.04	1.58/2.36	50.941	11.77	50.88	1.38/2.06
5	50	5	5.4	2	1	52.481	12.78	77.32	1.39/2.48	53.910	12.63	62.59	1.18/1.63
6	50	5	5.8	4	2	54.466	12.31	46.67	1.04/1.63	54.676	12.31	47.54	1.02/1.53
7	50	5	6.3	6	3	52.331	11.71	57.83	1.03/1.43	52.476	11.80	54.30	1.06/1.53
8	50	5	5.2	10	5	50.370	12.20	45.38	1.70/2.11	53.898	12.61	56.74	1.07/1.50
9	50	5	4.1	16	8	50.358	12.21	55.91	1.36/2.20	53.639	11.95	42.40	1.08/1.36
10	50	5	5.3	20	10	44.473	11.17	50.33	1.84/2.75	52.013	12.00	45.52	1.42/1.84

Notes: Num elitist probes, # pr; F_{max} = best fitness; g_{max} = max gain (dBi); VZ_0 = Var Z_0 feed impedance (Ω); $S = VSWR//VZ_0$ (min/max).

2.2. Dynamic Threshold Optimization

While *Elitism* has been extensively used in, for example, most multiobjective algorithms, ant colony algorithms, scatter search, ($\mu + \lambda$) algorithms, and many others [2], *DTO* has not. *Dynamic Threshold Optimization* is not a GSO heuristic *per se*. Instead it is a geometric approach to modifying the objective function's topology to remove local maxima. *DTO* can be used with *any* GSO algorithm. In this paper that GSO program happens to be CFO, but any other search/optimization program(s) could have been used instead, alone or in combination.

DTO operates by compressing the fitness function's landscape from below. It comprises a series of passes that successively increase the objective function's "floor" or threshold so as to filter out any local maxima below the threshold value. This is accomplished by using the auxiliary function $g(\vec{x}) = \left[f(\vec{x}) - T_k\right] \cdot U\left[f(\vec{x}) - T_k\right] + T_k$ where $f(\vec{x})$ is the objective function, T_k the threshold value on the k^{th} pass, and $U[\cdot]$ the Unit Step function, viz., $U(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$. Thus, for $f(\vec{x}) \geq T_k$, $g(\vec{x}) = f(\vec{x})$, whereas for $f(\vec{x}) < T_k$, $g(\vec{x}) = T_k$. On each successive pass, *DTO changes the topology* of the objective function by *redefining* it using auxiliary function $g(\vec{x})$. A more detailed discussion of *DTO* with examples appears in the Appendix.

2.3. Yagi Fitness Function

There are many ways to measure an antenna's performance, for example, as representative parameters: maximum directive gain, gain bandwidth, radiation pattern, pattern bandwidth, polarization, radiation efficiency, input impedance, impedance bandwidth, directionality, beamwidth, maximum sidelobe levels/directions, specific absorption rate, physical size, fabrication cost/time. In a real-world problem the antenna engineer must decide which of these parameters will be used and how they will be combined in an objective function to be maximized. Should they be added/subtracted? Multiplied? Made the arguments of some other mathematical manipulation, say, exponentiation? And so on, and so on. The possibilities are endless, yet some objective functions work much better than others, both in terms of reflecting the desired balance between antenna parameters and in being "searchable," that is, amenable to investigation by GSO. Some, unfortunately, are pathological, but not obviously so, and as a result they are difficult or impossible to deal with, especially when the pathology is hidden [4] [5] [6].

The work reported here uses the same (very simple) fitness function that is used in [1], *viz.*,

$$F = c_1 \cdot \boldsymbol{g}_L + c_3 \cdot \boldsymbol{g}_M + c_5 \cdot \boldsymbol{g}_U - c_2 \cdot \boldsymbol{S}_L - c_4 \cdot \boldsymbol{S}_M - c_6 \cdot \boldsymbol{S}_U \quad (1)$$

where subscripts L, M, and U denote lower, mid and upper frequencies at which the Yagi's power gain (directivity multiplied by radiation efficiency), $\boldsymbol{g}$, and feedpoint voltage standing wave ratio, $\boldsymbol{S}$, are computed. This objective function is simple, and it is anticipated to be well-behaved and amenable to GSO while reflecting the desired balance between antenna parameters. The values of the weighting coefficients c_i are: $c_1 = c_2 = c_5 = c_6 = 1$ and $c_3 = c_4 = 3$. They were chosen for simplicity while slightly favoring midband performance with L = 204.8 MHz, M = 299.8 MHz, U = 304.8 MHz. VSWR (Voltage Standing Wave Ratio) is computed relative to the feed point impedance Z_0 and is denoted VSWR//Z_0. While Z_0 usually is a fixed, user-supplied parameter, typically the industry standard value of 50 Ω resistive, Variable Z_0 technology (VZ_0), which was used here, treats it the same as any other decision space variable. This approach embraces array current distributions that otherwise would be excluded because they fail to adequately match the predetermined value of Z_0. Whether or not the CFO-returned value is feasible and desirable is an engineering and economic judgment, but more often than not impedance-matching the "non-standard" Z_0 is worthwhile because the antenna's performance is better, often much better [7]. Variable Z_0 technology is now available for public use without limitation [8]. At each of the three frequencies L, M, U, the Method of Moments code NEC-4 [9] computed the Yagi's maximum gain and feedpoint Z_0 from which VSWR//Z_0 was computed for use in the fitness function.

2.4. Yagi Array Prior Results

In [1] the best CFO-returned fitness occurs with 6% negative gravity (6% $G < 0$

where G is CFO's "gravitational constant"). Its value of 49.1892 corresponds to a maximum array gain of 11.92 dBi and feedpoint impedance Z_0 = 59.8 ohms with VSWR//59.8 ranging from 1.49 to 2.01 over the optimization frequency range 294.8 to 304.8 MHz. The reference array in that paper, that is, the CFO-optimized array with zero $G < 0$, has a best fitness of 47.8932 with a gain of 11.28 dBi and VSWR//65.56 ranging from 1.25 to 1.61. Injecting a small amount of negative gravity into CFO resulted in discovering a range of array designs whose fitnesses were better than or very similar to the reference array's. This improvement is attributable to enhanced DS exploration because the effect of negative gravity is to cause CFO's probes to fly away from each other, apparently into regions of DS that have been under-sampled or perhaps not sampled at all. In this paper CFO is further enhanced by including *Elitism* and *Dynamic Threshold Optimization*.

3. Results

3.1. Negative Gravity & Elitism

Table 1 shows the fitnesses and corresponding Yagi parameters for CFO with Negative Gravity ("$G < 0$") and with Elitism, but not with *DTO*. The results from [1] are included for comparison (the CFO runs in [1] used $G < 0$ but not Elitism). While the run in [1] was made with 550 steps, all the entries in Table 1 were made with 1000 using the same CFO setup parameters shown in Table 1 and Table 2 of [1]. The best **CFO-GED** results with only $G < 0$ and Elitism are highlighted in **bold red** text.

Table 1 shows the number of probes, N_p, the target and actual values of $G < 0$, and the amount of Elitism as a percentage of N_p with the corresponding number of elitist probes *# pr*. Step-by-step elitism was employed, so the number of elitist probes shown in the table was pseudo randomly injected into the following step. Because CFO-GED is completely deterministic, even with π-fraction prv's, every

Table 2. Fitness Data with $G < 0$, *Elitism* and *DTO*. [Target 5% $G < 0$; *Elitism*, step-by-step 4 best probes within each run, single best probe run-to-run, no seed probe.].

Yagi #	*DTO Thresholds*	N_p	N_t	Best Fitness	g_{max} (dBi)	Z_0 (Ω)	*VSWR//Z_0* Min/Max
11	0/20/25/35/45	10/15/30/50/200	5/20/30/100/200	54.945	12.82	51.61	1.02/1.83
12	0/20/25/35/45	10/15/30/100/300	5/20/30/100/150	55.117	**12.87**	48.01	1.19/1.65
13	0/25/35/45/50	5/10/20/50/350	5/20/30/50/200	50.000	3.20	30.00	161.3/196.5
14	0/20/25/35/45	5/10/20/50/450	5/20/30/100/200	55.603	12.53	42.75	1.08/1.41
15	0/20/25/35/45	5/10/20/100/500	5/20/30/100/250	54.949	12.30	46.91	**1.03/1.35**
16	0/20/25/35/45	5/10/20/100/500	5/20/30/100/5000	**56.641**	12.84	33.69	1.01/1.68
17	0/20/25/35/45	5/10/20/50/1000	5/20/30/50/100	55.588	12.54	48.98	1.04/1.61
18*	0/20/25/35/45	5/10/20/50/450	5/20/30/100/200	45.000	0.92	30.29	36.51/38.44

* Re-run of Yagi #14 without *Elitism*, no seed probe, but with $G < 0$ (5% target, 3.6% actual) and *DTO* only.

run in Table 1 can be precisely recovered simply by running it again. If any change is made in the setup then a different fitness result is a consequence of that change, likewise for changing the objective function. Deterministic GSO programs permit the (relatively) quick comparison of how well different run setups or fitness functions perform. This flexibility is essential for solving real-world problems, but it is not provided by stochastic GSO's (see [6] for some specific examples).

Turning to the effect of adding *Elitism* to CFO along with $G < 0$, the data in Table 1 provide convincing evidence that supplementing $G < 0$ with *Elitism* improves the discovered fitnesses by quite a bit. The best fitness of 54.466 is nearly 11% better than the best fitness in [1] which used 6% $G < 0$ without *Elitism*. Regardless of the specific problem being worked, the obvious conclusion is that all CFO runs should be augmented with both $G < 0$ and *Elitism* because in all likelihood the results will be better.

3.2. Negative Gravity, Elitism & DTO

The next question is whether or not including *DTO* improves CFO's performance even more. Recall from Figure 1 that the *DTO* shell comprises a series of passes with progressively increasing thresholds (objective function "floors"). Setting the sequence of thresholds can be tricky, quite tricky in some cases, because the problem's landscape becomes sparser and sparser as the floor rises and consequently adequate exploration may be impeded. *DTO* appears to work quite well with highly multimodal objective functions such as the Schfwefel's Problem 2.26 (see Appendix) because there is sufficient topology at every threshold to guide the search. When the landscape is too flat, however, *DTO* may miss the remaining maxima and return the threshold value itself as its best fitness. For the Schwefel 2.26 the very high level of multimodality is evident from the 2D plot in Figure A3, pass #1, so *DTO* is expected to, and in fact does, perform very well. But no such plot is available to assess how multimodal is the Yagi problem. In the problem's twelve dimensions its landscape might resemble the Schewfel 2D's, but then again it might not. This conundrum is inherent in *DTO*, and it requires consideration in setting up the *DTO* shell. Section A3 of this paper discusses this issue in greater detail, and provides some examples of setting thresholds by calculation.

For the Yagi problem, however, the first approach is setting the thresholds by enumeration because the fitness function is readily bounded, approximately, even though its maximum value is unknown. The best VSWR value of course is 1, and a reasonable maximum gain is, say, 15 dBi midband and 10 dBi at the band edges, these values being based on experience with this type of array. Inserting them into Equation (1) yields a maximum fitness of $F_{\max} = 60$. The CFO runs with $G < 0$ and *Elitism* suggest that reasonable best fitnesses probably are in the range 55 - 60, so that the highest threshold should be placed far enough below this range that CFO can still effectively explore DS. Again, because CFO is deterministic it is straightforward to run test cases to determine what is a suitable threshold sequence, whereas a stochastic GSO might make this impossible from a practical point of view.

Table 2 shows results for runs with all three added extensions, $G < 0$, *Elitism*, and *DTO*. *Elitism* was introduced in two ways, *step-by-step* and *run-to-run*. For *step-by-step* the four best probes at each step were pseudo randomly injected into the next step, whereas for *run-by-run* the single best probe in each run was injected as the last probe in the next run. Five *DTO* passes were made. The table shows the enumerated thresholds and the number of probes and time steps at each threshold. These run parameters are followed by the best CFO-returned fitness, and the Yagi's maximum gain, *Variable* Z_0 feedpoint impedance, and the corresponding VSWR range.

The data in Table 2 show that adding *DTO* for the most part substantially increased the best fitness, from 54.466 to 56.641, a change of 2.175 or about 4%. The data also show the pitfalls of choosing thresholds poorly, specifically Yagis #13 and 18. Yagi #13's performance is quite poor because CFO was unable to locate any maxima that were not on the threshold itself (value of 50). In this case the last *DTO* threshold of 50 was simply too high, which depressed CFO's exploration to the point of its not discovering any other maxima. Yagi #18 was run to investigate the effect of removing *Elitism* entirely. In this case the threshold values and number of probes and time steps used in Yagi #14 were replicated, but the *DTO* passes were made only with $G < 0$, no *Elitism*. As in the previous case, CFO was unable to locate any maxima above the threshold value of 45, and the result was an array with extremely poor performance.

Table 3 summarizes the best fitness data and run statistics for the eighteen Yagi designs that used negative gravity, elitism and *DTO*, but no seed probe. *DTO* was not included for the first twelve runs through Yagi #10, but it was included for Yagis #11 through 18. The averages fitness for the first set was 49.710. Including *DTO* improved that statistic substantially to 55.476 for the second. The total number of NEC runs is included as a measure of computational effort and ranges from a low just over twenty thousand to a high slightly above two and one half million. While the longest run did provided the best overall fitness, 56.461, its length raises the important question of how much fitness improvement is worth the additional computational effort. NEC runs averaged 0.037129 sec for the work reported here.

Adding a seed probe improves the best fitness even more as shown in Table 4. Each run was seeded with a probe having the coordinates of the best fitness probe from the previous run. For example, the seed probe for Yagi #25 was the best probe located by Run #24. Yagi #19 was seeded with a probe from a run that did not use negative gravity, elitism or *DTO*. Except for the shortest run, Yagi #21, the best fitness increased monotonically using seeded runs. The benefit of starting a search in the vicinity of a previous global maximum is apparent from these data, and would be a recommended approach for any GSO.

Table 3. Best fitness summary. [Target 5% $G < 0$; *Elitism*, step-by-step 4 best probes within each run, single best probe run-to-run, no seed probe.].

Yagi #	# NEC Runs	*Total Steps*	*Steps to F_{max}*	*DTO?*	Best Fitness
1	20,020	1000	691		52.164
2	20,020	1000	995		46.806
3	20,020	1000	835		45.433
4	20,020	1000	974		44.912
5	50,050	1000	941		52.481
6	50,050	1000	972	↑ No	54.466
6(i)	137,137	1000	668	Avg: 49.710	51.666
6(ii)	136,773	500	75	↓	51.057
7	50,050	1000	696		52.331
8	50,050	1000	860		50.370
9	50,050	1000	967		50.358
10	50,050	1000	967		44.473
11	46,555	355	347		54.954
12	56,705	305	284		55.117
13	73,760	305	106	↑ Yes	50.000
14	91,360	355	352	Avg:	55.603
15	136,460	405	314	55.476 ↓	54.949
16	2,511,460	5155	2693		56.641
17	104,410	355	204		55.588
18	96,360	355	156	n/a	45.000

Note 1: Case 6(i): $N_p = 137$, $N_t = 1000$; Case 6(ii): $N_p = 273$, $N_t = 500$. Note 2: For averaging, Yagis #13 & #18 excluded because DTO threshold was too high. Note 3: n/a → not applicable.

Table 4. Run data using seed probe each run.

Yagi #	% $G < 0$	Elitist Probes #/N_p	# NEC Runs	Best Fitness	g_{max} (dBi)	Z_0 (Ω)	*VSWR//Z_0* Min/Max
19	4.4	1/405	61105	56.483	13.12	42.66	1.01/1.84
20	4.8	1/1215	183820	56.826	13.11	39.61	1.00/1.82
21	6.2	4/320	6820	53.851	13.18	54.34	1.18/2.24
22				56.842	13.02	38.79	1.00/1.74
23				56.871	12.99	38.49	1.00/1.53
24	↑	↑	↑	56.906	12.98	37.80	1.00/1.52
25	5.6 ↓	2/512 ↓	204216 ↓	57.066	13.03	36.62	1.01/1.67
26				57.067	13.03	36.69	1.00/1.68
27				57.067	13.03	36.70	1.00/1.68

Note: For Yagis #23-27 Best Probe Coordinates from Previous Run Used as Seed Probe.

All *DTO* results presented for Yagis #11 through 18 were computed using specified thresholds as shown in Table 2. But that approach may not be the best. The thresholds for runs 19 through 27 were calculated, rather than enumerated. Table 5 shows threshold and best fitness data for Yagi #19 and for Yagi #27, the data for the other runs in Table 4 all being similar. Each run started with five probes, and the number of probes was doubled on each successive threshold. The thresholds were computed as $T_k = F_{\min}^{k-1} + C \cdot \left(F_{\max}^{k-1} - F_{\min}^{k-1} \right)$ where $F_{\max}$, $F_{\min}$ are maximum, minimum fitnesses, and $C = 0.2$ for $k = 1$, $C = 0.8$ for $k > 1$. $F_{\min}^{0}$ and $F_{\max}^{0}$ are the minimum and maximum fitnesses returned on the initialization run which is made with no threshold. The coefficient C was determined experimentally, which is easily accomplished with a deterministic GSO like CFO because otherwise using a stochastic GSO it is difficult or impossible to determine the effect of changing C. With a stochastic program different results may be a consequence of changing a coefficient, or of the algorithm's randomness, or both. A deterministic algorithm like CFO removes all doubt as to why its results are different.

As seen in Table 5(a) for Yagi #19 using five *DTO* thresholds, the best fitness increases significantly with increasing threshold. It ranges from a low of 37.8033 on a threshold of −1055.3856 to a high of 56.4300 on a threshold of 54.116. For Yagi #27, Table 5(b), using seven thresholds, the corresponding data are fitness of 57.0669 with a threshold of −1312.5739, and 57.0670 on a threshold of 57.0073. By this run the best fitness has essentially saturated as is seen in the data in Table 4. It is again apparent that a deterministic GSO allows the user to quickly converge on a run setup that minimizes computer usage while homing in on an optimized design. In this case, as soon as saturation of the best fitness is clear, then there is no point in doing additional calculations because they will not provide significantly better results.

Table 6 provides a summary of how the best fitness is improved by extending CFO with *Negative Gravity*, *Elitism* and *DTO*. The overall improvement compared to the reference run in [1] in which no extension was used is quite dramatic, just under 20%. It is clear that CFO performs better as a global search and optimization algorithm when these extensions are included.

4. Design or Optimization?

These observations raise a very important *practical question* about how to best design the "best" antenna, whether it be a Yagi array or any other antenna. Should it be "optimized" as was the sequence of Yagi designs discussed thus far, or should it be "designed" by specifying minimum performance criteria which, if met, terminate the design process. This activity is the "D" in global search "D/O," *design* and *optimization*. The fact is, again as a practical matter, it almost always will be quicker and will utilize fewer resources to set up the antenna problem as a design problem instead of as an optimization problem. And to that end, a deterministic design algorithm makes all the difference in the world because the antenna engineer has complete control of every aspect of the process, nothing being "left to chance."

As an example, a *design run* was made with the objectives of *midband* gain of at least 13 dBi and midband VSWR less than 1.5 relative to the *Variable* Z_0-computed feedpoint impedance. Rather than use the fitness function in Equation (1) the following fitness was used instead:

$$F = 1 - \left(g_t - g_M / S_M\right) / \left(g_t + g_M / S_M\right) \tag{2}$$

Table 5. (a). Yagi #19; (b). Yagi #27.

(a)

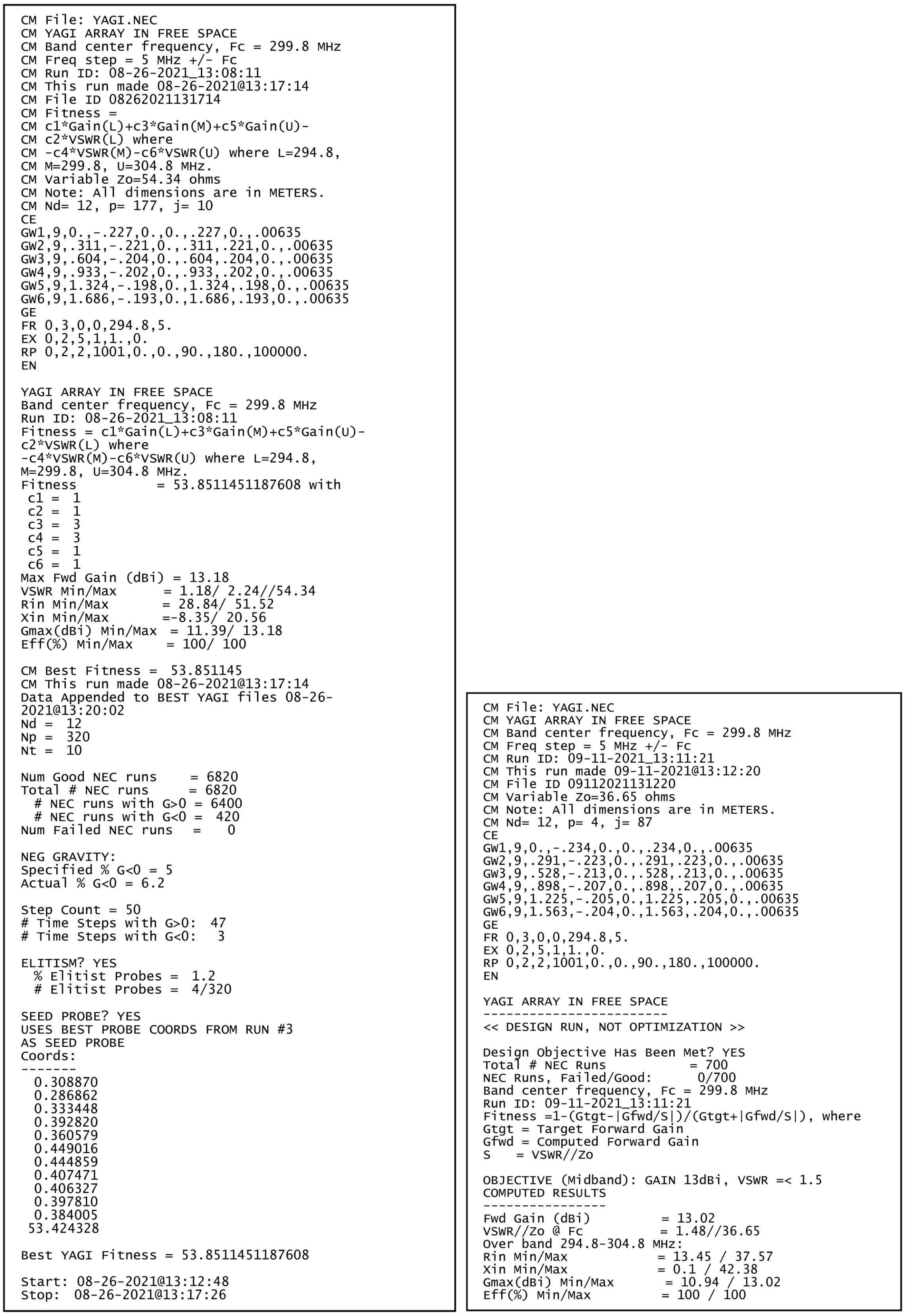
T#	T_k	F_{max}
1	−1055.3856	37.8033
2	−171.3621	49.6438
3	5.4426	49.6438
4	44.647	54.4482
5	54.116	56.4300

(b)

T#	T_k	F_{max}
1	−1312.5739	57.0669
2	−129.5613	57.0669
3	19.7413	57.0669
4	49.6019	57.0670
5	55.5740	57.0670
6	56.7684	57.0670
7	57.0073	57.0670

Table 6. CFO extension summary.

CFO Extension						
Type	*Elitism* Type			*DTO*	Best Fitness	Percent Improvement
	Step-by-Step	Run-to-Run	Seed Probe			
NONE	---	---	---	---	47.893	0
$G < 0$, *Elitism*	yes	no	no	no	54.466	13.72
$G < 0$, *Elitism*	yes	no	yes	no	54.676	14.16
$G < 0$, *Elitism*	yes	yes	no	yes (enumerated)	56.641	18.27
$G < 0$, *Elitism*	yes	yes	yes	yes (calculated)	57.067	19.16



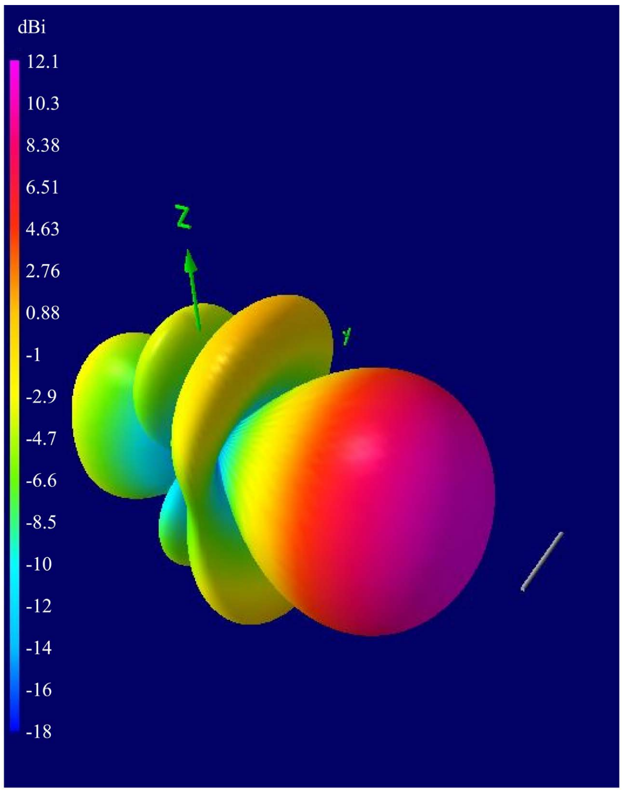
```
CM File: YAGI.NEC
CM YAGI ARRAY IN FREE SPACE
CM Band center frequency, Fc = 299.8 MHz
CM Freq step = 5 MHz +/- Fc
CM Run ID: 08-26-2021_13:08:11
CM This run made 08-26-2021@13:17:14
CM File ID 08262021131714
CM Fitness =
CM c1*Gain(L)+c3*Gain(M)+c5*Gain(U)-
CM c2*VSWR(L) where
CM -c4*VSWR(M)-c6*VSWR(U) where L=294.8,
CM M=299.8, U=304.8 MHz.
CM Variable Zo=54.34 ohms
CM Note: All dimensions are in METERS.
CM Nd= 12, p= 177, j= 10
CE
GW1,9,0.,-.227,0.,0.,.227,0.,.00635
GW2,9,.311,-.221,0.,.311,.221,0.,.00635
GW3,9,.604,-.204,0.,.604,.204,0.,.00635
GW4,9,.933,-.202,0.,.933,.202,0.,.00635
GW5,9,1.324,-.198,0.,1.324,.198,0.,.00635
GW6,9,1.686,-.193,0.,1.686,.193,0.,.00635
GE
FR 0,3,0,0,294.8,5.
EX 0,2,5,1,1.,0.
RP 0,2,2,1001,0.,0.,90.,180.,100000.
EN

YAGI ARRAY IN FREE SPACE
Band center frequency, Fc = 299.8 MHz
Run ID: 08-26-2021_13:08:11
Fitness = c1*Gain(L)+c3*Gain(M)+c5*Gain(U)-
c2*VSWR(L) where
-c4*VSWR(M)-c6*VSWR(U) where L=294.8,
M=299.8, U=304.8 MHz.
Fitness          = 53.8511451187608 with
 c1 =  1
 c2 =  1
 c3 =  3
 c4 =  3
 c5 =  1
 c6 =  1
Max Fwd Gain (dBi) = 13.18
VSWR Min/Max      = 1.18/ 2.24//54.34
Rin Min/Max       = 28.84/ 51.52
Xin Min/Max       =-8.35/ 20.56
Gmax(dBi) Min/Max  = 11.39/ 13.18
Eff(%) Min/Max     = 100/ 100

CM Best Fitness =  53.851145
CM This run made 08-26-2021@13:17:14
Data Appended to BEST YAGI files 08-26-
2021@13:20:02
Nd =  12
Np =  320
Nt =  10

Num Good NEC runs     = 6820
Total # NEC runs      = 6820
  # NEC runs with G>0 = 6400
  # NEC runs with G<0 =  420
Num Failed NEC runs   =    0

NEG GRAVITY:
Specified % G<0 = 5
Actual % G<0 = 6.2

Step Count = 50
# Time Steps with G>0:  47
# Time Steps with G<0:   3

ELITISM? YES
  % Elitist Probes =  1.2
  # Elitist Probes =  4/320

SEED PROBE? YES
USES BEST PROBE COORDS FROM RUN #3
AS SEED PROBE
Coords:
-------
  0.308870
  0.286862
  0.333448
  0.392820
  0.360579
  0.449016
  0.444859
  0.407471
  0.406327
  0.397810
  0.384005
 53.424328

Best YAGI Fitness = 53.8511451187608

Start: 08-26-2021@13:12:48
Stop:  08-26-2021@13:17:26
```

(a)

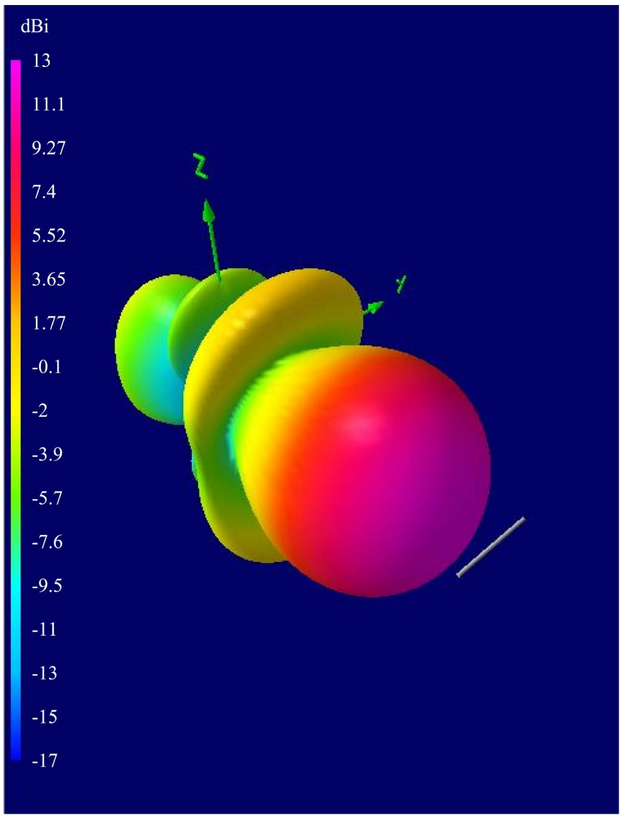
```
CM File: YAGI.NEC
CM YAGI ARRAY IN FREE SPACE
CM Band center frequency, Fc = 299.8 MHz
CM Freq step = 5 MHz +/- Fc
CM Run ID: 09-11-2021_13:11:21
CM This run made 09-11-2021@13:12:20
CM File ID 09112021131220
CM Variable Zo=36.65 ohms
CM Note: All dimensions are in METERS.
CM Nd= 12, p= 4, j= 87
CE
GW1,9,0.,-.234,0.,0.,.234,0.,.00635
GW2,9,.291,-.223,0.,.291,.223,0.,.00635
GW3,9,.528,-.213,0.,.528,.213,0.,.00635
GW4,9,.898,-.207,0.,.898,.207,0.,.00635
GW5,9,1.225,-.205,0.,1.225,.205,0.,.00635
GW6,9,1.563,-.204,0.,1.563,.204,0.,.00635
GE
FR 0,3,0,0,294.8,5.
EX 0,2,5,1,1.,0.
RP 0,2,2,1001,0.,0.,90.,180.,100000.
EN

YAGI ARRAY IN FREE SPACE
------------------------
<< DESIGN RUN, NOT OPTIMIZATION >>

Design Objective Has Been Met? YES
Total # NEC Runs           = 700
NEC Runs, Failed/Good:      0/700
Band center frequency, Fc = 299.8 MHz
Run ID: 09-11-2021_13:11:21
Fitness =1-(Gtgt-|Gfwd/S|)/(Gtgt+|Gfwd/S|), where
Gtgt = Target Forward Gain
Gfwd = Computed Forward Gain
S    = VSWR//Zo

OBJECTIVE (Midband): GAIN 13dBi, VSWR =< 1.5
COMPUTED RESULTS
----------------
Fwd Gain (dBi)         = 13.02
VSWR//Zo @ Fc          = 1.48//36.65
Over band 294.8-304.8 MHz:
Rin Min/Max           = 13.45 / 37.57
Xin Min/Max           = 0.1 / 42.38
Gmax(dBi) Min/Max      = 10.94 / 13.02
Eff(%) Min/Max         = 100 / 100
```

(b)

Figure 2. (a). NEC input file, yagi #21(4); (b). NEC input file, design run.



(a)



(b)

Figure 3. (a). Pattern yagi #21(4); (b). Pattern design run yagi.

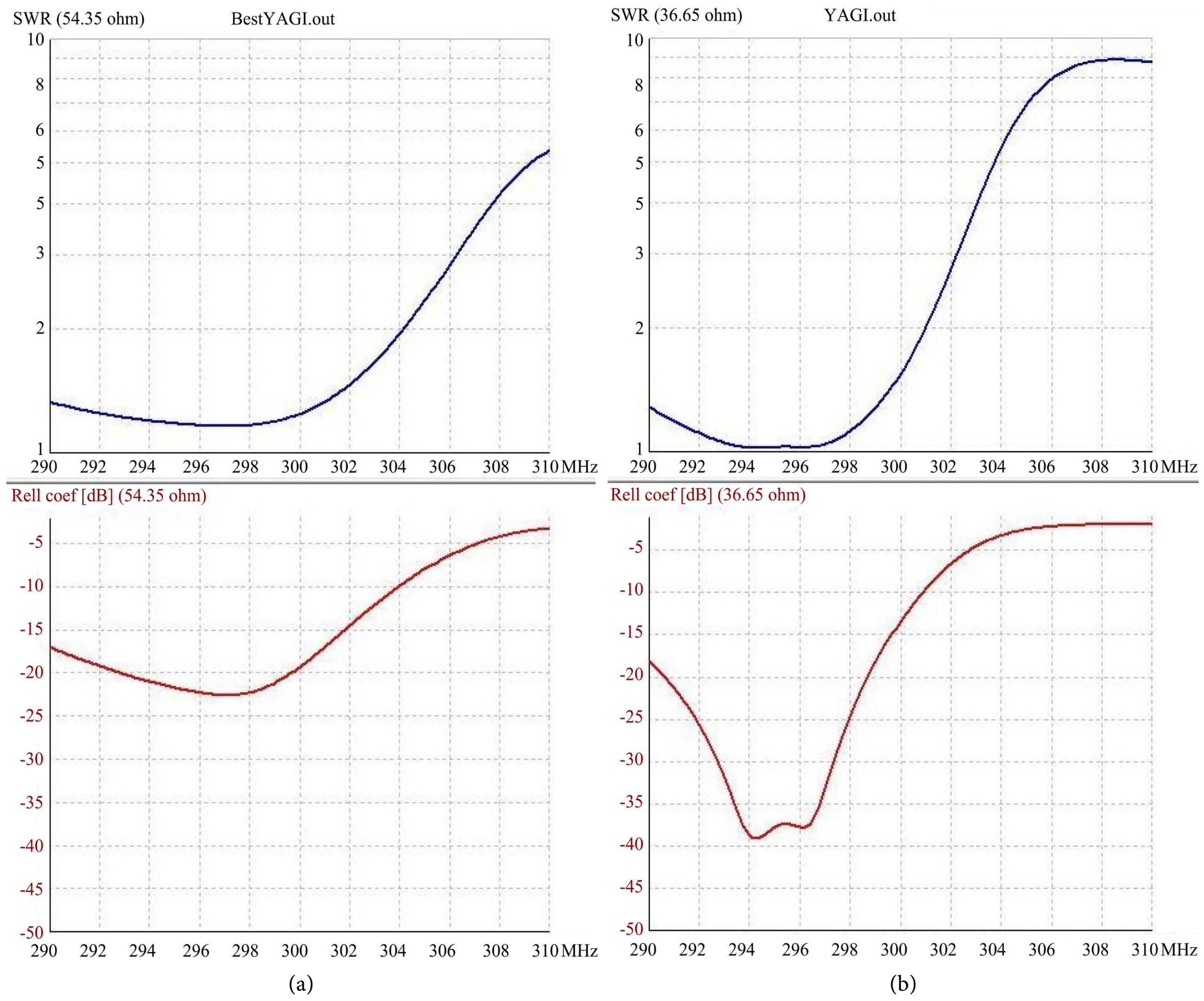


Figure 4. (a). VSWR//54.34 yagi #21(4); (b). VSWR//36.65 design run yagi.

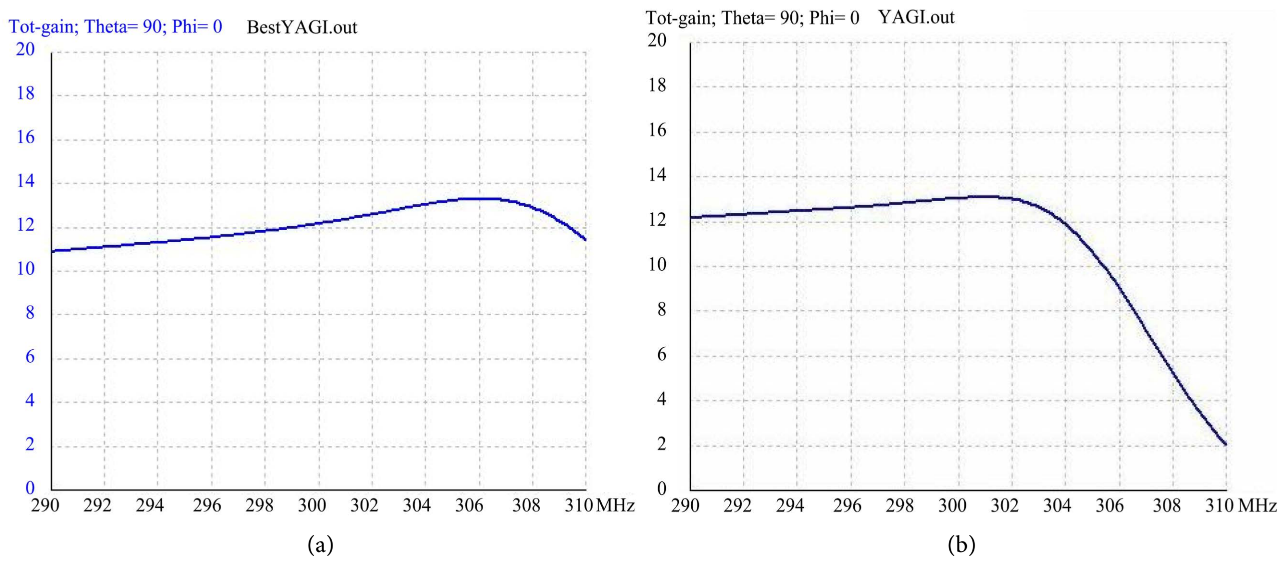


Figure 5. (a). Gain yagi #21(4); (b). Gain design run yagi.

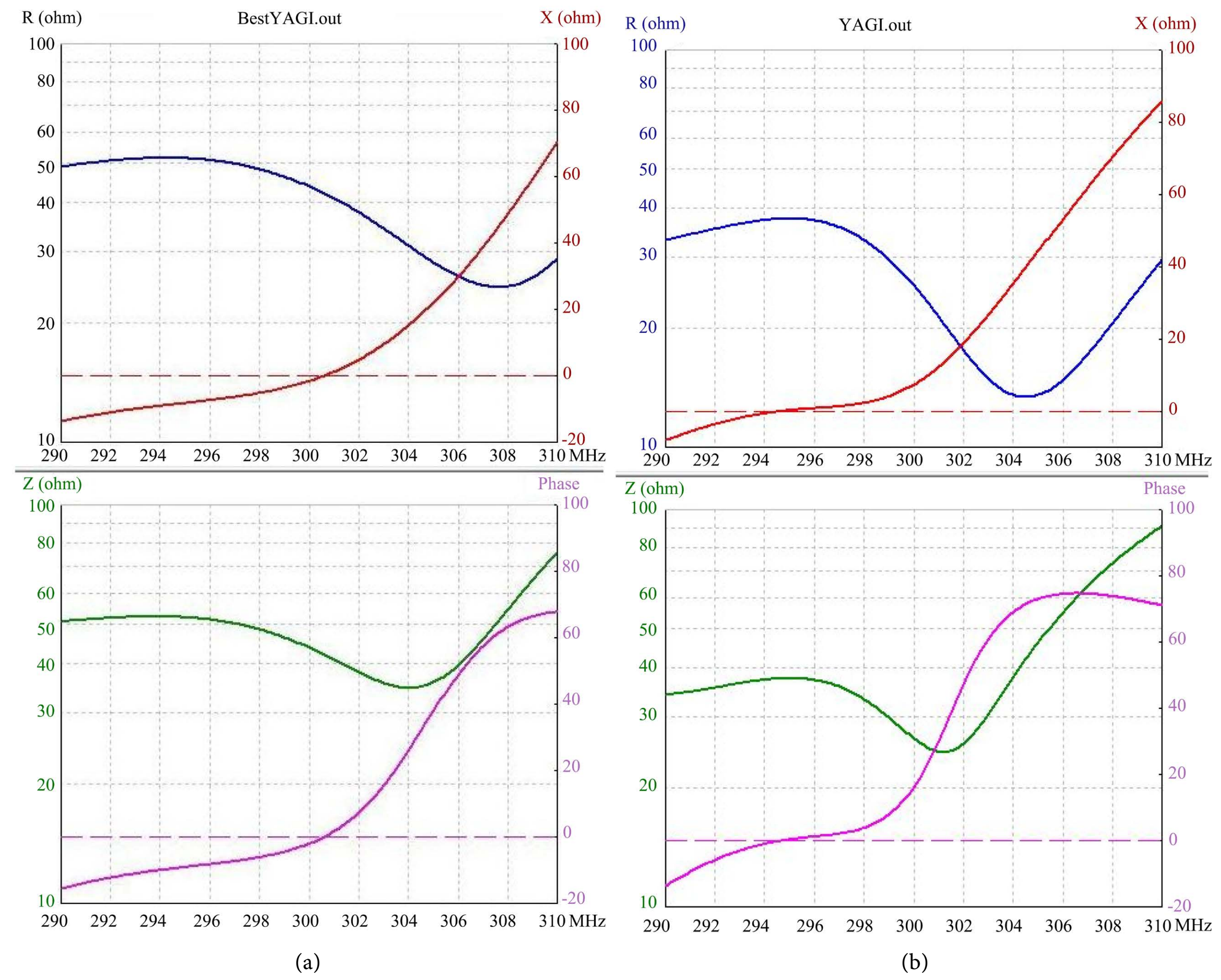


Figure 6. (a). Zin yagi #21(4); (b). Zin design run yagi.

where the variables have the same meanings as before with the new variable $\boldsymbol{g}_t$ being the target midband gain. Using the same run parameters that were used for Yagi #21 this design run achieved midband gain and VSWR, respectively, of 13.02 dBi and 1.48 relative to a feedpoint impedance of 36.65 Ω. The corresponding results for Yagi #21 are a *maximum* gain across the band 294.8 - 304.8 MHz of 13.18 dBi with VSWR//54.34 Ω ranging from 1.18 to 2.24. At midband, however, 299.8 MHz, Yagi #21 has a gain of 12.13 dBi and VSWR//54.34 Ω of 1.23. Thus, it misses the midband gain objective but meets the VSWR objective.

A very important difference between the Yagi #21 and this example design run is the computational effort. The number of NEC runs for Yagi #21 was 6820 whereas only 700 NEC runs were required for the design run. Of course, the Yagi #21 run was made at three frequencies, not one, and an entirely different fitness function was used. Nevertheless it is unlikely that these differences account for requiring nearly ten times as many NEC runs. Rather, it is far more likely that the explanation lies in performing optimization instead of design. As a general proposition, all things being equal, design is likely to be much quicker than optimization, and this design run is an example of that effect.

This example also serves to highlight the importance of a deterministic GSO. Let's say the fitness function Equation (2) was modified by adding two exponents, *m* and *n*, as follows:

$$F = 1 - \left(\boldsymbol{g}_{\mathrm{t}} - \boldsymbol{g}_{\mathrm{M}} / \boldsymbol{S}_{\mathrm{M}}\right)^{m} \Big/ \left(\boldsymbol{g}_{\mathrm{t}} + \boldsymbol{g}_{\mathrm{M}} / \boldsymbol{S}_{\mathrm{M}}\right)^{n} \tag{2a}$$

How should values be assigned to these parameters? There is no obvious theoretical guidance for assigning any specific values, so the inevitable approach is trial and error. The problem with trial and error is that there are dozens of combinations that might be tried, and to evaluate them on a comparative basis, which values work better than others, would take hundreds of runs of a stochastic GSO and very likely tens of thousands of NEC runs. And this is only one change that might be made. It might be desirable or necessary to try many altogether different fitness functions, which raises the same question: Which approach is better, a stochastic GSO or a deterministic one? Which has the same obvious answer, the deterministic one.

In order to compare antenna performances, **Figures 2-6** show the NEC-computed antenna performance data for Yagi #21 and for the Design Run Yagi. Both Yagis are quite good antennas, but depending on the application one may be preferred over the other. The figures are self-explanatory as to their meaning, and they permit a head-to-head comparison of these two arrays.

5. Conclusion

This paper has investigated the effect of adding three extensions to the GSO Central Force Optimization as applied to Yagi-Uda array design and optimization (D/O), those extensions being: 1) A small measure of pseudo randomly injected *Negative Gravity*, ($G < 0$); 2) Two types of *Elitism*, *step-by-step* and *run-to-run*; and 3) *Dynamic Threshold Optimization*. The basic CFO algorithm does not include any of these extensions. This paper extends the work reported in a previous paper that considered only $G < 0$ and which showed a significant performance improvement over a range of optimized arrays. Still better optimization results are obtained by adding to the mix *Elitism* and *DTO*. While this work was limited to the D/O of 6-element Yagis, the reasonable conclusion based on these data is that *any* antenna D/O, indeed *any* GSO problem, antenna or not, utilizing CFO as the GSO engine will benefit by adding all three extensions, probably substantially. Adding *Elitism* to CFO with negative gravity alone improved the best fitness by nearly 11%. Adding *DTO* with enumerated thresholds and no seed probe increased the best fitness by approximately another 4%. Still further improvement is possible by including a *seed probe* (coordinates of the best fitness location from a previous run) and by using calculated instead of enumerated *DTO* thresholds. These modifications increased the best returned fitness from 56.641 to 57.067. For comparison, from [1] the best fitness without *Negative Gravity*, *Elitism* or *DTO* is 47.8932 whereas when all three are added the best fitness increases to 57.0670, an overall increase of 19.16%.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Formato, R.A. (2021) Six-Element Yagi Array Designs Using Central Force Optimization with Pseudo Random Negative Gravity. *Wireless Engineering and Technology*, **12**, 23-51. https://doi.org/10.4236/wet.2021.123003

[2] Luke, S. (2015) Essentials of Metaheuristics. 2nd Edition, Online Ver. 2.2, Oct. 2015, Sect. 3.3.1. https://cs.gmu.edu/~sean/book/metaheuristics/

[3] Formato, R.A. (2012) Dynamic Threshold Optimization—A New Approach? http://arXiv.org/abs/1206.0414

[4] Formato, R.A. (2013) Issues in Antenna Optimization—A Monopole Case Study. *ACES* (*Applied Computational Electromagnetics Society*) *Journal*, **28**, 1122-1133.

[5] Formato, R.A. (2017) Determinism in Electromagnetic Design & Optimization—Part II: BBP-Derived π Fractions for Generating Uniformly Distributed Sampling Points in Global Search and Optimization Algorithms. Forum for Electromagnetic Research Methods and Application Technologies. Vol. 19, Article No. 10, https://www.e-fermat.org

[6] Formato, R.A. (2017) Determinism in Electromagnetic Design & Optimization—Part I: Central Force Optimization. FERMAT, Forum for Electromagnetic Research Methods and Application Technologies. Vol. 19, Article No. 9. https://www.e-fermat.org

[7] Dib, N., Sharaqa, A. and Formato, R.A. (2013) Variable Z_0 Applied to the Optimal Design of Multi-Stub Matching Network and a Meander Monopole. *International Journal of Microwave and Wireless Technologies*, **6**, 55-514. https://doi.org/10.1017/S1759078713001049

[8] U.S. Patent No. 8776002. https://portal.uspto.gov/pair/PublicPair

[9] Burke, G.J. (2011) Numerical Electromagnetics Code—NEC-4.2 Method of Moments, Part I: User's Manual. LLNL-SM-490875, Lawrence Livermore National Laboratory (USA), Livermore, CA, 2011.

[10] Wang, J. (2011) Particle Swarm Optimization with Adaptive Parameter Control and Opposition. *Journal of Computational Information Systems*, **7**, 4463-4470

[11] Formato, R.A. (2010) Parameter-Free Deterministic Global Search with Simplified Central Force Optimization. In: Huang, D.-S., Zhao, Z., Bevilacqua, V. and Figueroa, J.C., Eds., *Advanced Intelligent Computing Theories and Applications* (ICIC2010), Lecture Notes in Computer Science, 309-318, Springer-Verlag, Berlin. https://doi.org/10.1007/978-3-642-14922-1_39

[12] Hayes, B. (2011) Quasirandom Ramblings. *American Scientist*, **99**, 282-287. https://doi.org/10.1511/2011.91.282 https://www.americanscientist.org/ http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.365.4074&rep=rep1&type=pdf

Appendix: Dynamic Threshold Optimization

The following material is adapted from the author's arXiv post: http://arXiv.org/abs/1206.0414.

A.1. Problem Statement

In a bounded hyperspace $\Omega := \left\{ \vec{x} \mid x_i^{\min} \le x_i \le x_i^{\max}, i = 1, \cdots, N_d \right\}$ (decision space, DS), the x_i being decision variables and $\vec{x}$ a decision vector, determine the locations and values of the global maxima of the objective function $f\left(x_1, x_2, \cdots, x_{N_d}\right)$, that is, $\max\left\{ f\left(\vec{x}\right) : \vec{x} \in \Omega \subset \Re^{N_d}, f : \Omega \subset \Re^{N_d} \to \Re \right\}$. The value of objective function $f\left(\vec{x}\right)$ at each point $\vec{x}$ is its "fitness," and the problem's "landscape" (topology over DS) is $L = \Omega \cup f\left(\vec{x}\right)$, $\vec{x} \in \Omega \subset \Re^{N_d}$

A.2. Dynamic Threshold Optimization: Theory

DTO conceptually is quite simple. Figure A1 is a schematic illustration of how it works in a one-dimensional (1-D) DS. Objective function $f\left(\vec{x}\right)$ is multimodal with many local maxima and a single global maximum, and the problem is to locate that maximum (coordinates and value). DTO bounds $f\left(\vec{x}\right)$ from below using a series of successively increasing "thresholds," in effect compressing DS in the direction of the dependent variable (from "below") instead of, as is sometimes done, shrinking DS by reducing an independent variable's domain (from the "sides"). Locating the global maximum is easier in the compressed DS because unwanted local maxima are progressively filtered out as the "floor" (threshold) rises. Because DTO is a general geometric technique, it is algorithm-independent so that it can be used with any global search and optimization algorithm. Although DTO is described in the context of maximization, it can be applied to minimization as well with obvious modifications.

Procedure $OPT\left[q\left(\vec{x}\right), \vec{x}^*, q^*, q_{\min}\right]$ is a global search and optimization (GSO) routine that returns 1) the N_d coordinates $\vec{x}^*$ of a maximum of function $q\left(\vec{x}\right)$, 2) its value q^*, and 3) a minimum value $q_{\min}$ (no coordinates). $OPT[\cdot]$ may comprise any search and optimization algorithm (singly or in combination with others) regardless of its type, deterministic, stochastic, or hybrid; and different algorithms may be used on successive calls to $OPT[\cdot]$.

Selecting DTO's thresholds can be tricky because its returned values are highly dependent on how well the modified objective function landscape can be searched. One approach is to initialize DTO by applying $OPT[\cdot]$ to $f\left(\vec{x}\right)$ without any threshold. Its return values then are used to define a starting threshold that subsequently is updated by applying $OPT[\cdot]$ to the auxiliary function $g\left(\vec{x}\right) = \left[f\left(\vec{x}\right) - T_k\right] \cdot U\left[f\left(\vec{x}\right) - T_k\right] + T_k$, where T_k is the threshold value on the k^{th} DTO pass, and $U[\cdot]$ is the Unit Step function, $U(z) = \begin{cases} 1, & z \ge 0 \\ 0, & z < 0 \end{cases}$. Thus, for $f\left(\vec{x}\right) \ge T_k$, $g\left(\vec{x}\right) = f\left(\vec{x}\right)$, whereas for $f\left(\vec{x}\right) < T_k$, $g\left(\vec{x}\right) = T_k$. On each successive pass, DTO *changes the topology* of the objective function by *redefining* it using auxiliary function $g\left(\vec{x}\right)$. The DTO run continues until a user-specified termination criterion is met (often maximum number of passes or fitness saturation). Its pseudocode appears in Figure A2.

A.3. Setting DTO Thresholds

How to set DTO's starting threshold and how it is updated are determined by the algorithm designer. It can be done by enumeration, calculation, or some combination of both. One obvious starting value is the minimum fitness returned by $OPT[\cdot]$, that is, $T_0 = f_{\min}$ (threshold by calculation). This appears to be a good default choice when there is no other information about the objective function that permits specifying specific threshold values (enumeration). But updating the thresholds T_k as DTO progresses is more problematic because of the floor's profound impact on $f(\vec{x})$'s landscape. More and more local maxima are removed from the landscape as the threshold rises, so that effectively sampling DS becomes progressively more difficult (the topology becomes flatter and flatter). In the limit of the floor rising to a global maximum, DS collapses to a plane, and there is no information available for performing a search. How well DS can be explored thus becomes more and more of an issue as the threshold rises, and the search algorithm's exploration characteristics become very important. One approach to setting T_k is shown in Figure A1 in which successive thresholds are set to the best returned fitness, $T_k = g_k^*$, but this approach has not worked well in numerical tests because a good GSO often sets the threshold too high too early in the run. The 2-D example that follows employs a different approach, and it clearly illustrates the effect of flattening the landscape too much.

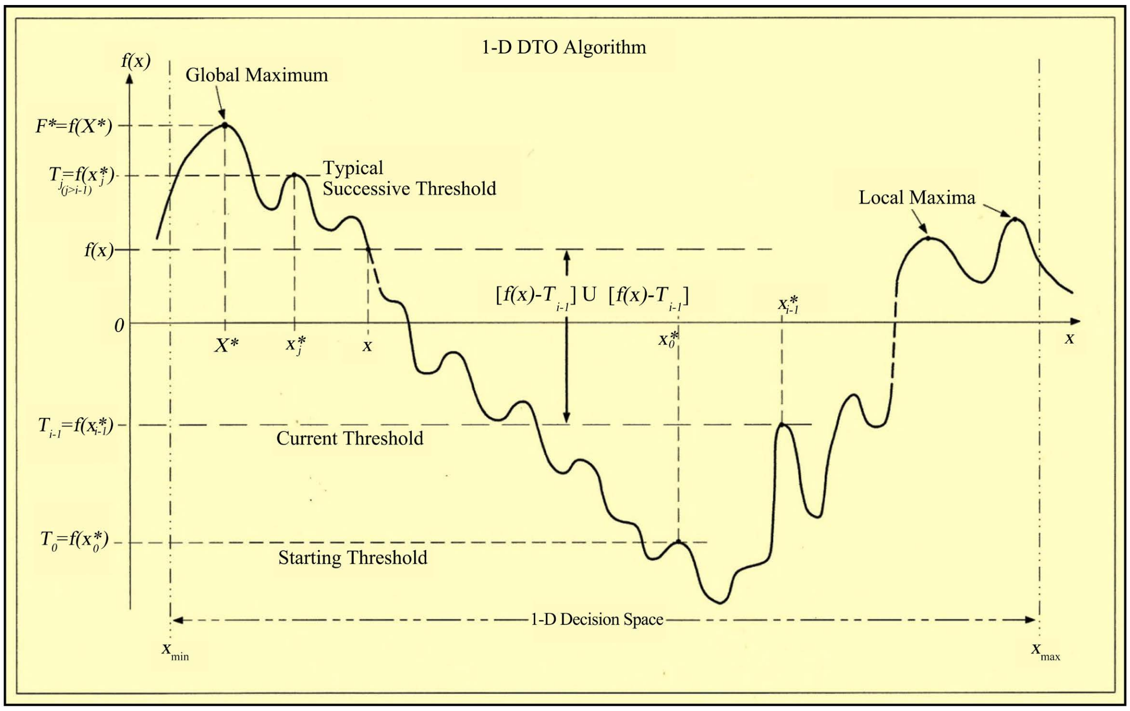


Figure A1. DTO concept with thresholds at successive local maxima.

```
Algorithm DTO

(i)  Initialization
     CALL  $OPT[f(\vec{x}), \vec{x}_0{}^*, f_0{}^*, f_{\min}]$
     SET  $T_0$  (Starting Threshold – see text; typically  $T_0 = f_{\min}$ )
(ii) Loop over successive thresholds
        $k \leftarrow 0$  (following standard notation  $\leftarrow$  means "is set to")
        $F^* = -N$  (initialize best overall fitness, very large number < 0)
        DO UNTIL [Termination Criterion]  (see text)
            (a)  $k \leftarrow k+1$  (increment pass #)
            (b)  CALL  $OPT[g(\vec{x}), \vec{x}_k{}^*, g_k{}^*, g_{\min}]$  where
                 $g(\vec{x}) = [f(\vec{x}) - T_{k-1}] \cdot U[f(\vec{x}) - T_{k-1}] + T_{k-1}$
            (c)  IF  $g_k{}^* \ge F^*$  $\therefore$  $F^* = g_k{}^*, \vec{X}^* = \vec{x}_k{}^*$  where
                     $\vec{X}^*$  is the location of the best overall fitness
            (d)  UPDATE THRESHOLD:  $T_k$  (see text)
     LOOP
(iii) Return:  $\vec{X}^*$,  $F^* = f(\vec{X}^*)$  (best overall fitness: coordinates & value)
```

Figure A2. DTO pseudocode.

A.4. 2D Example: Schwefel's Problem 2.26

As an example of how it works, DTO was applied to Schwefel's Problem 2.26 in 2D using basic CFO as the GSO algorithm (not CFO-GED). In an N_d-dimensional decision space, this objective function is defined as

$f(x) = \sum_{i=1}^{N_d} \left[x_i \sin\left(\sqrt{|x_i|}\right) \right]$, $-500 \le x_i \le 500$. It has a single global maximum of $418.9829 \times N_d$ at $[420.9687]^{N_d}$ ([10], p.4467). The 2D global maximum is 837.9658@(420.9687, 420.9687). DTO was implemented with a number of passes $P = 10$ with a progressively increasing threshold computed as

$T_k = F_{\min} + \frac{C_{th} k}{P}\left(F^* - F_{\min}\right)$, $k = 1, \cdots, P$ (no threshold for $k = 0$), where F^* and $F_{\min}$, respectively, are the best and worst overall fitnesses returned through pass k. The coefficient $C_{th} = 0.98$ in this case is included to keep the threshold far enough below the global maximum that the landscape is not compressed into a plane. This formula for setting the threshold was chosen as much for its ability to illustrate the DTO concept (see plots below) as for its ability to produce good results, and there no doubt are countless other approaches to setting the threshold that will work as well or better.

The number of CFO probes was initialized to $N_p = 4$, and it was doubled on each successive pass in order to enhance CFO's exploration. Each run comprised $N_t = 25$ time steps. While CFO is an inherently deterministic search and optimization metaheuristic, in this case it was implemented with a random initial probe distribution (IPD) instead of the usual "Probe Line" IPD [11]. The reason for this change, again, was to enhance CFO's exploration in the progressively flatter landscape.

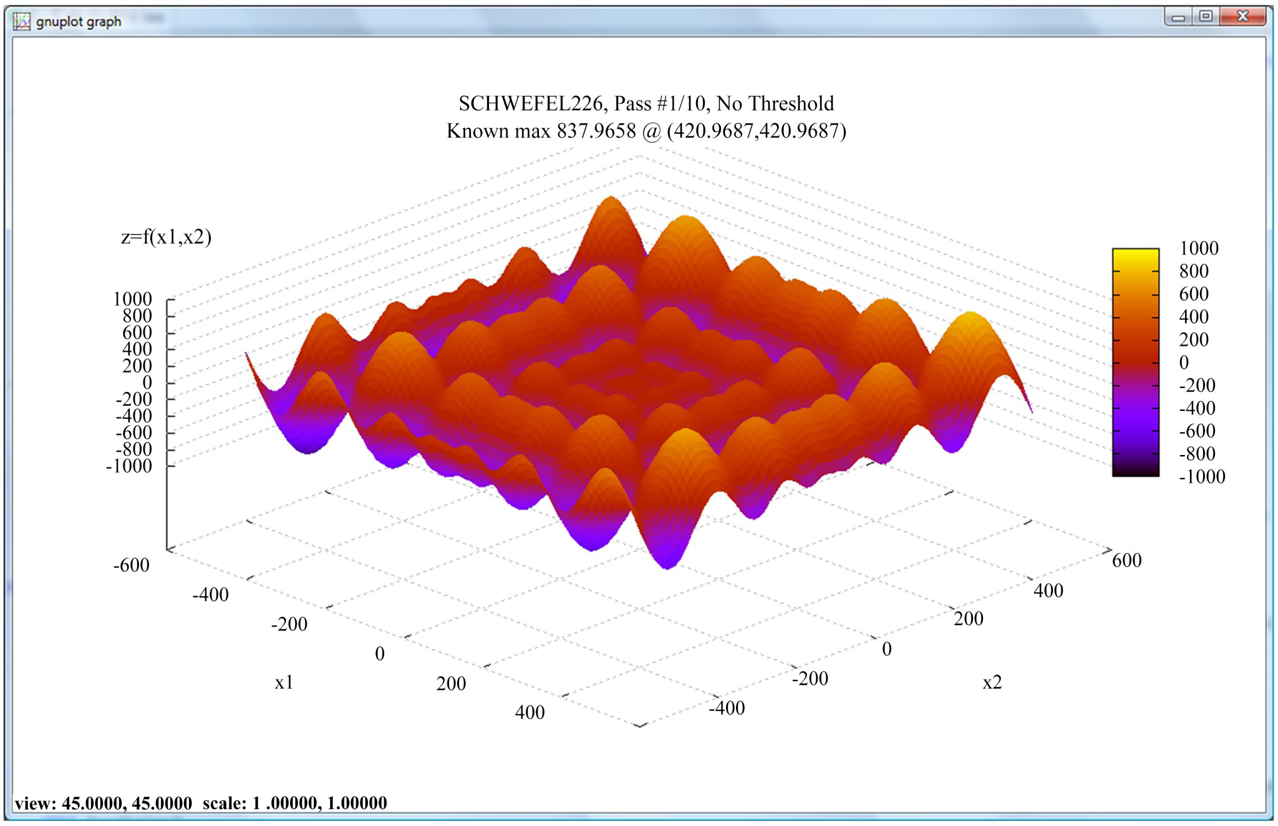
The DTO/CFO algorithm returned a best overall fitness of $F^* = 837.965574726692$ at the point $(x_1, x_2) = (421.007498176246, 420.959700549993)$ using a total of 106,392 function calls. The errors in the fitness and in the coordinates, respectively, are 0.0002253 and (−0.0387982, 0.0089995) [computed as Known minus DTO], which are quite small. Table A1 summarizes the DTO threshold evolution pass-by-pass and CFO's best fitness.

Figure A3 shows how DTO compresses the landscape as its threshold increases. The objective function is plotted at each of the 10 passes. The first pass (no threshold) shows the Schwefel Problem 2.26's complex landscape. It is highly multimodal with many similar amplitude local maxima. As DTO progresses more and more of these maxima are filtered out because the floor is higher and higher relative to the single global maximum. At pass #8, for example, 16 local maxima are visible, whereas at thresholds #9 and 10, respectively, the number of maxima falls to 8 and to 3. On the last pass the global maximum is clearly visible on the right side of the plot.

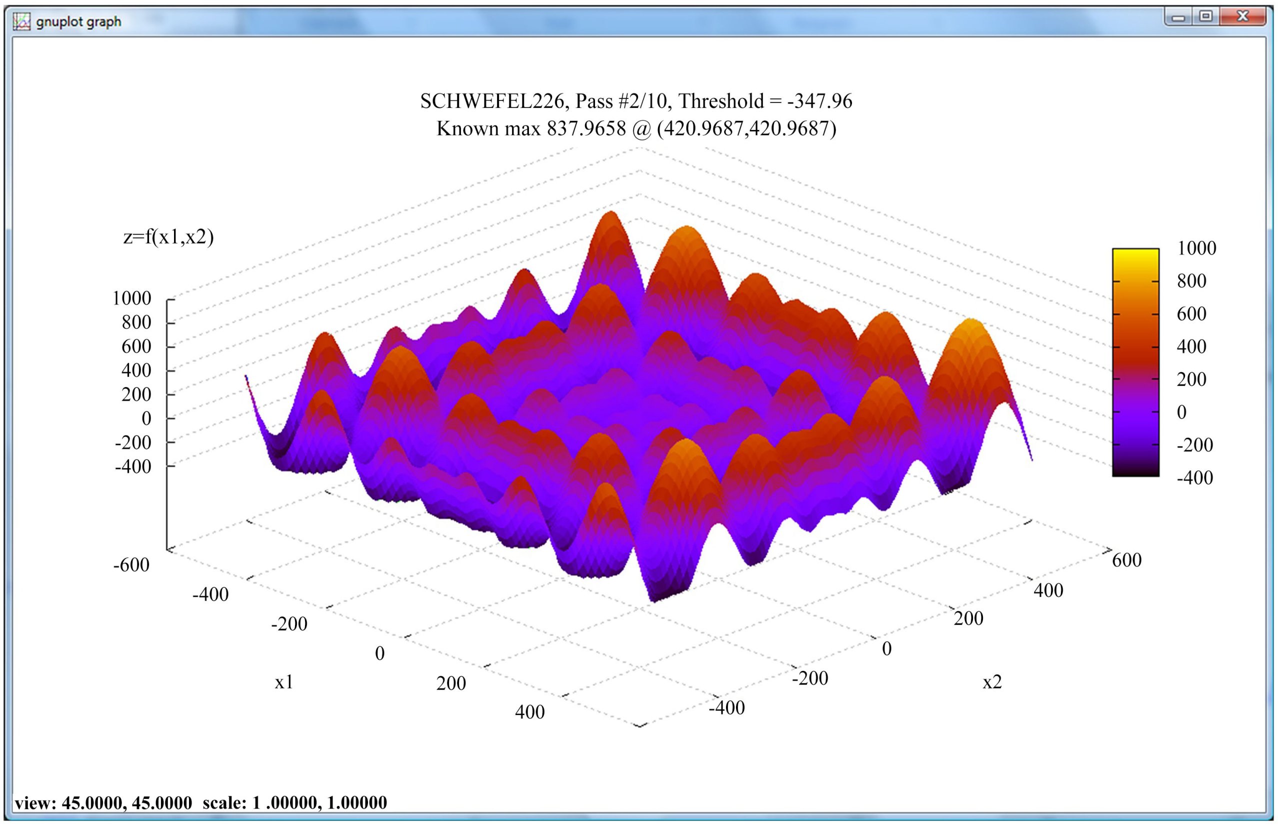
DTO also was tested against Schwefel 2.26 in 30D. Six passes were made using the linear threshold scheme described above, but with $C_{th} = 0.6$. Unlike the 2D case, a deterministic CFO implementation was used with a Probe Line IPD and $\gamma = [0,1]$, $\Delta\gamma = 0.1$ (see [12] for details). Other CFO parameters were the same as above except for $N_t = 15$. Passes 2 through 6, in order, had calculated thresholds of −10,176.09, −7828.153, −5480.216, −3132.279, and −212.722. The best fitness returned by DTO was 12,569.28 at the point $x_i = 420.7353, i = 1, \cdots, 29$, $x_{30} = 420.7662$. This result is quite good compared to the known maximum of 12,569.487 (fractional error of 1.647×10^{-5}) requiring a total of 44,352 function evaluations.

Table A1. DTO/CFO results for 2-D Schwefel Problem 2.26.

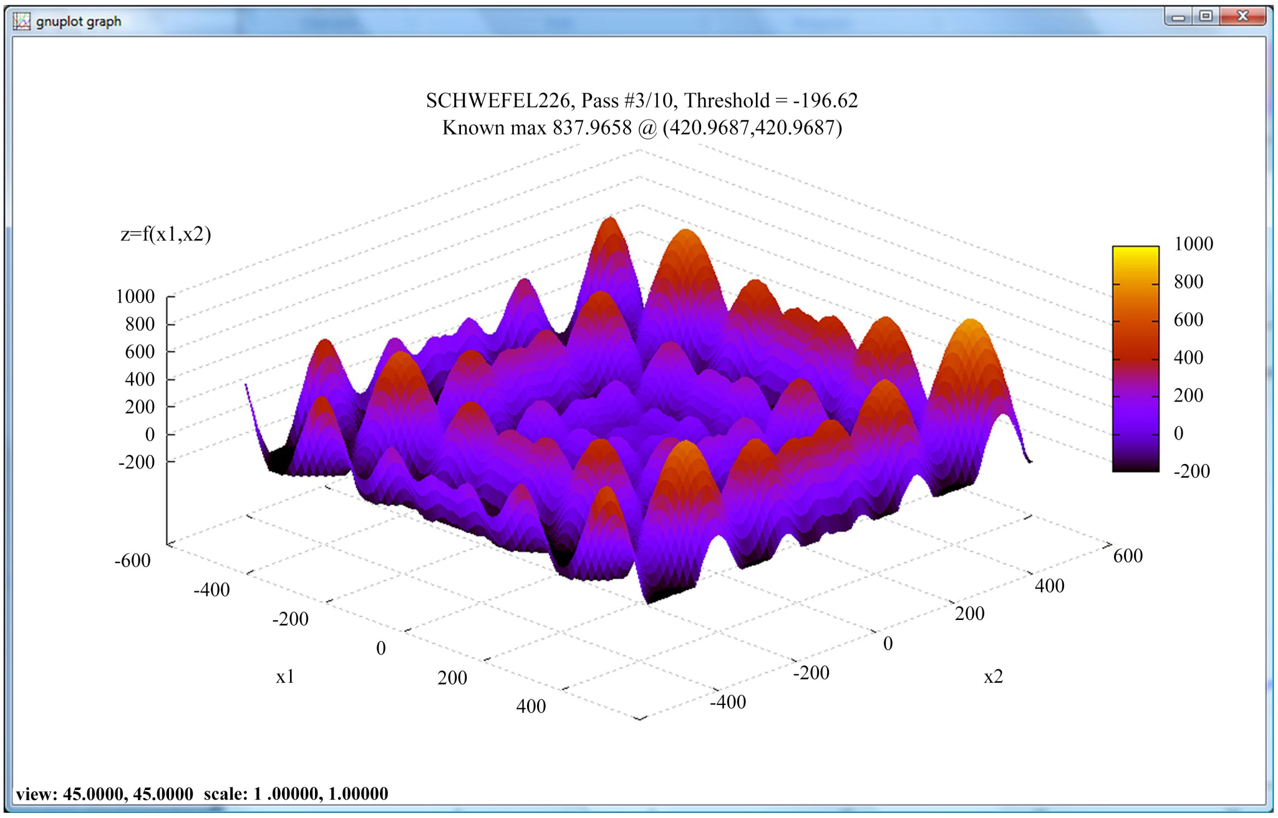
Pass #	*Threshold*	*Best Fitness*
1	*none*	580.2973878
2	−347.955	837.8781823
3	−196.617	719.2845548
4	−70.522	837.8956233
5	55.580	837.9282076
6	181.693	837.9654027
7	307.815	837.9504301
8	433.919	837.9647313
9	560.022	837.9650286
10	686.126	837.9655747


gnuplot graph
SCHWEFEL226, Pass #1/10, No Threshold
Known max 837.9658 @ (420.9687,420.9687)
z=f(x1,x2)
1000
800
600
400
200
0
-200
-400
-600
-800
-1000
x1
x2
-600
-400
-200
0
200
400
600
view: 45.0000, 45.0000 scale: 1 .00000, 1.00000

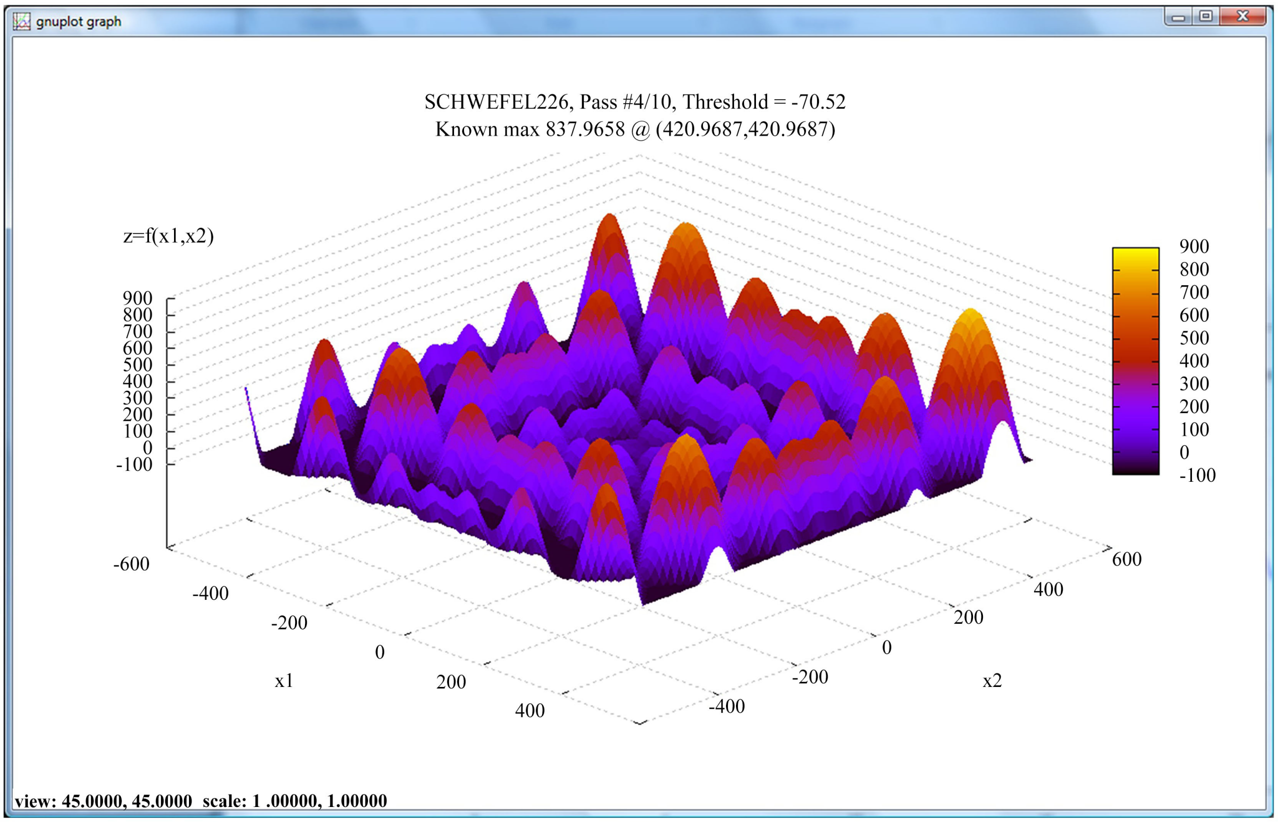
Pass #1


gnuplot graph
SCHWEFEL226, Pass #2/10, Threshold = -347.96
Known max 837.9658 @ (420.9687,420.9687)
z=f(x1,x2)
1000
800
600
400
200
0
-200
-400
x1
x2
-600
-400
-200
0
200
400
600
view: 45.0000, 45.0000 scale: 1 .00000, 1.00000

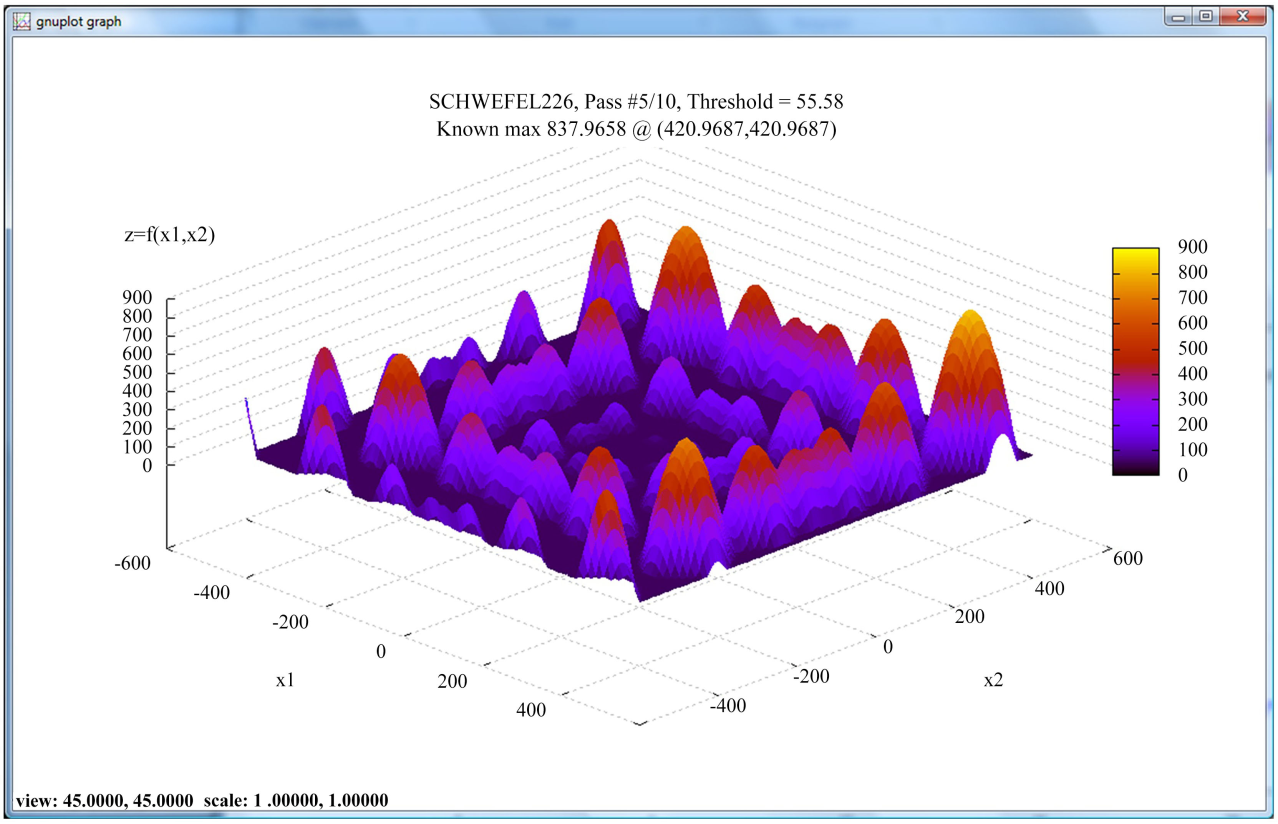
Pass #2


gnuplot graph
SCHWEFEL226, Pass #3/10, Threshold = -196.62
Known max 837.9658 @ (420.9687,420.9687)
z=f(x1,x2)
1000
800
600
400
200
0
-200
-600
-400
-200
0
x1
200
400
-400
-200
0
200
400
600
x2
1000
800
600
400
200
0
-200
view: 45.0000, 45.0000 scale: 1 .00000, 1.00000

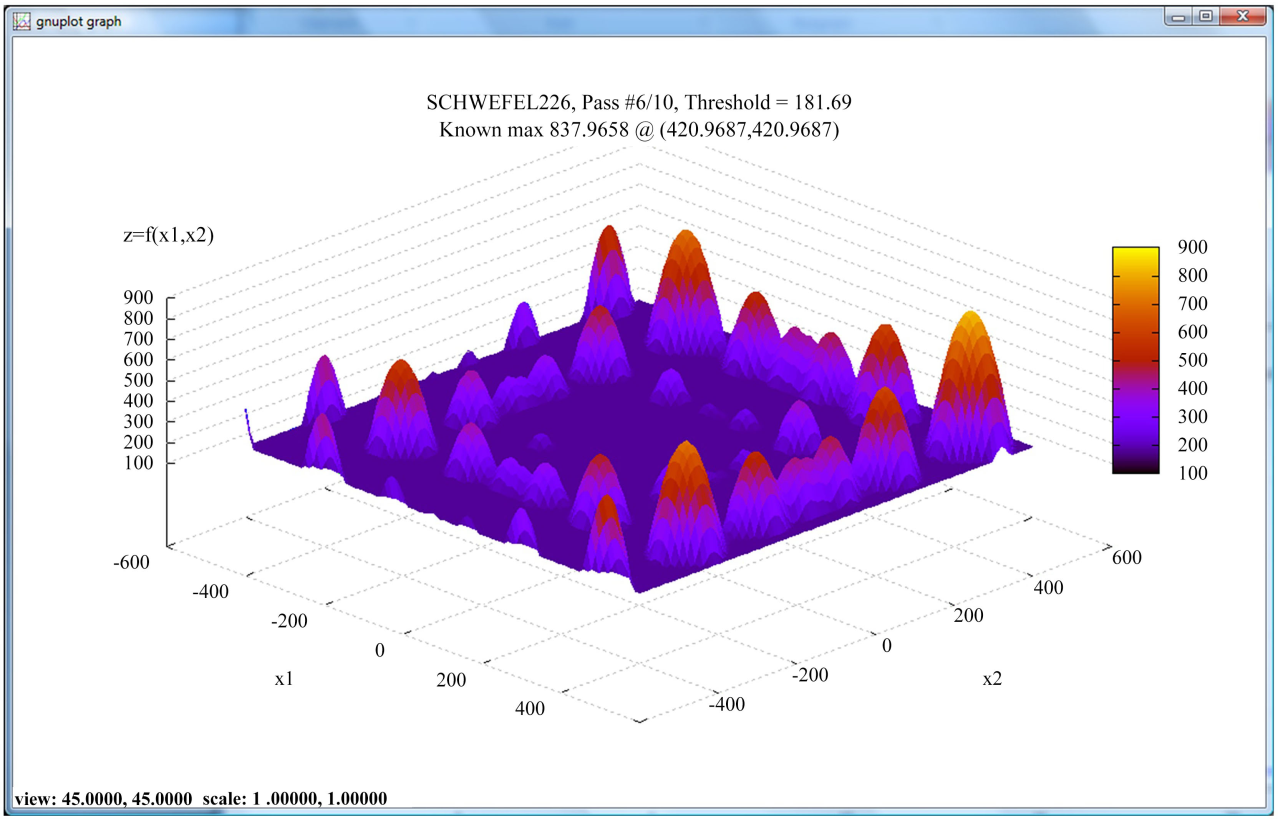
Pass #3


gnuplot graph
SCHWEFEL226, Pass #4/10, Threshold = -70.52
Known max 837.9658 @ (420.9687,420.9687)
z=f(x1,x2)
900
800
700
600
500
400
300
200
100
0
-100
-600
-400
-200
0
x1
200
400
-400
-200
0
200
400
600
x2
900
800
700
600
500
400
300
200
100
0
-100
view: 45.0000, 45.0000 scale: 1 .00000, 1.00000

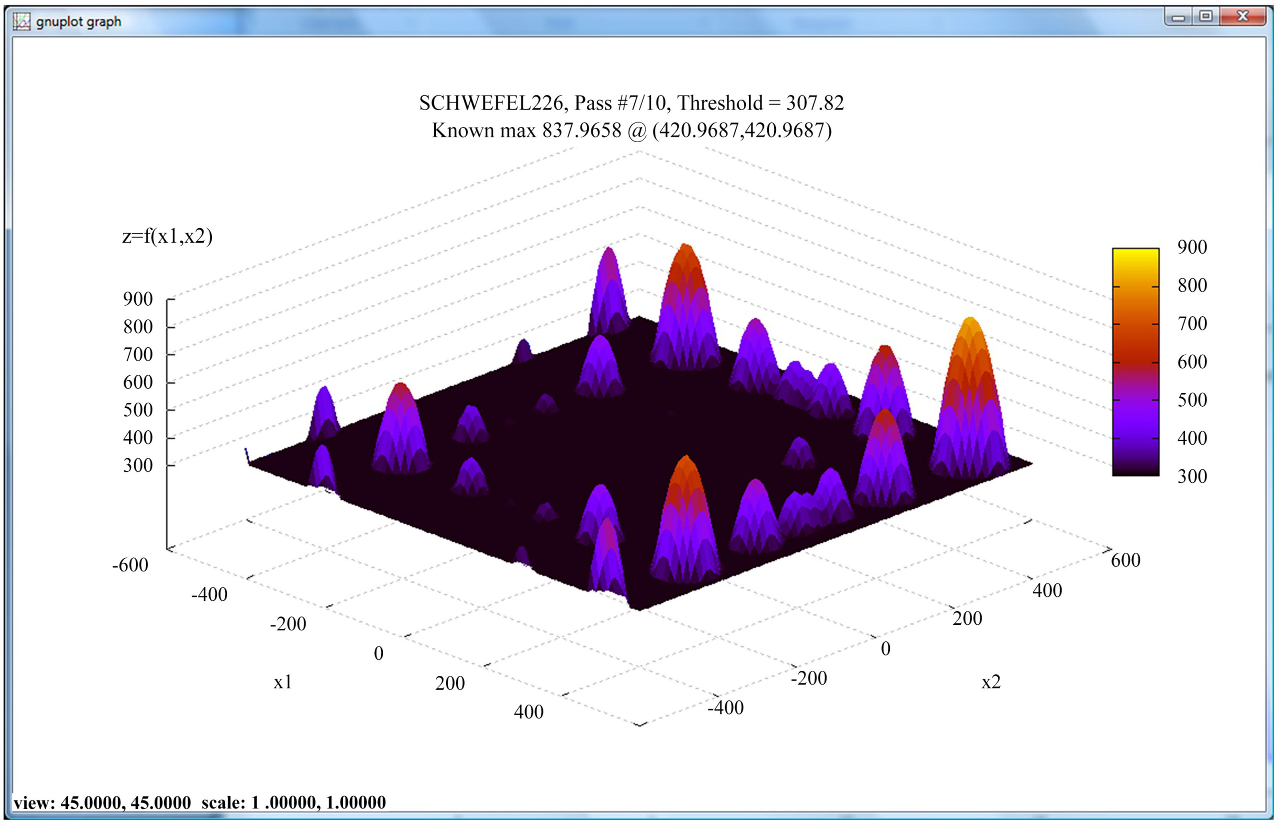
Pass #4


gnuplot graph
SCHWEFEL226, Pass #5/10, Threshold = 55.58
Known max 837.9658 @ (420.9687,420.9687)
z=f(x1,x2)
900
800
700
600
500
400
300
200
100
0
-600
-400
-200
0
200
400
x1
-400
-200
0
200
400
600
x2
view: 45.0000, 45.0000 scale: 1 .00000, 1.00000

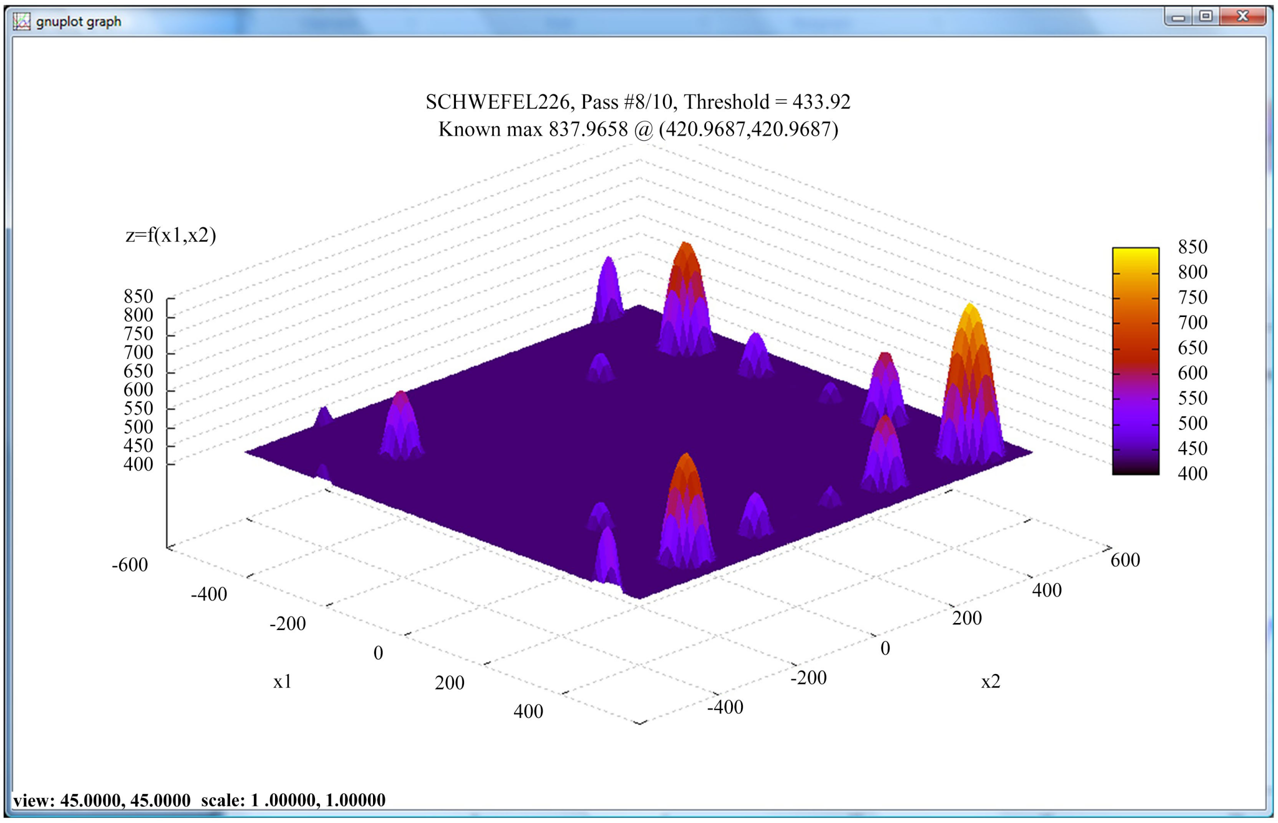
Pass #5


gnuplot graph
SCHWEFEL226, Pass #6/10, Threshold = 181.69
Known max 837.9658 @ (420.9687,420.9687)
z=f(x1,x2)
900
800
700
600
500
400
300
200
100
-600
-400
-200
0
200
400
x1
-400
-200
0
200
400
600
x2
view: 45.0000, 45.0000 scale: 1 .00000, 1.00000

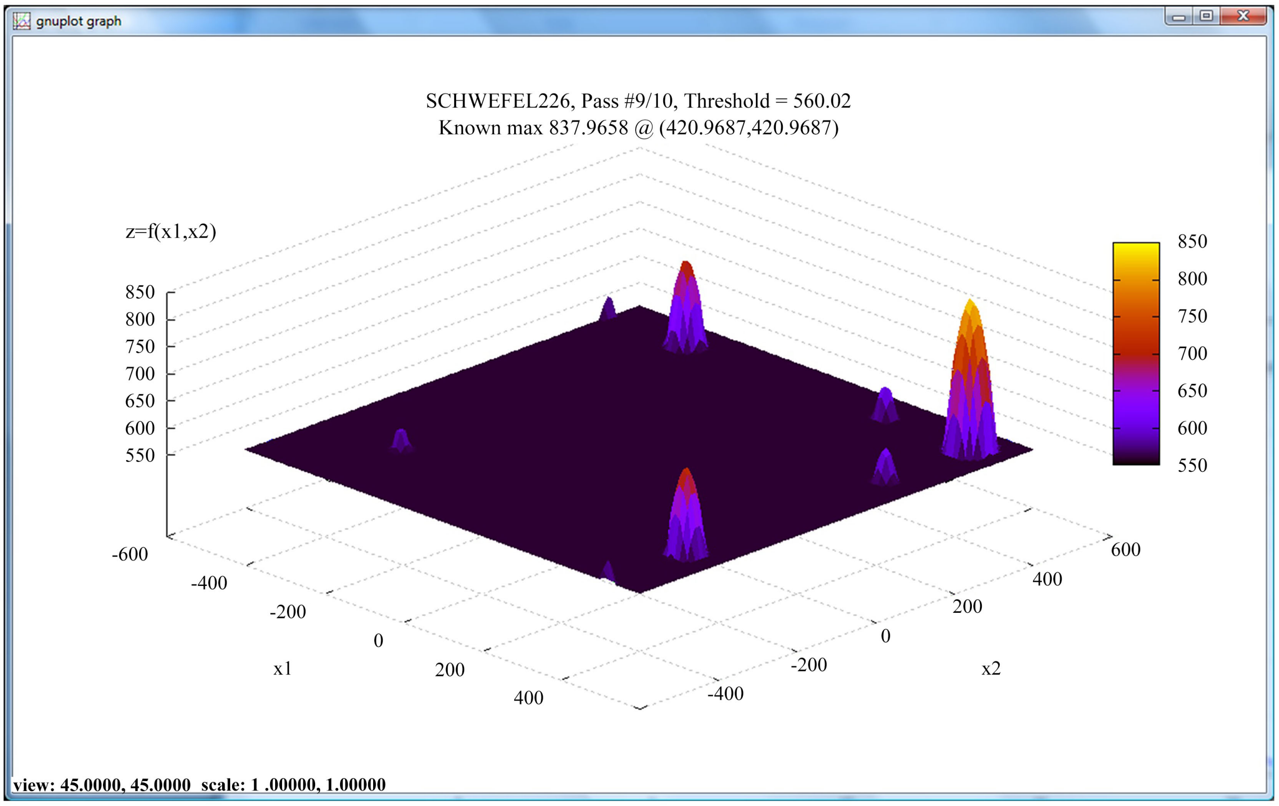
Pass #6


gnuplot graph
SCHWEFEL226, Pass #7/10, Threshold = 307.82
Known max 837.9658 @ (420.9687,420.9687)
z=f(x1,x2)
900
800
700
600
500
400
300
-600
-400
-200
0
200
400
x1
600
400
200
0
-200
-400
x2
view: 45.0000, 45.0000 scale: 1 .00000, 1.00000

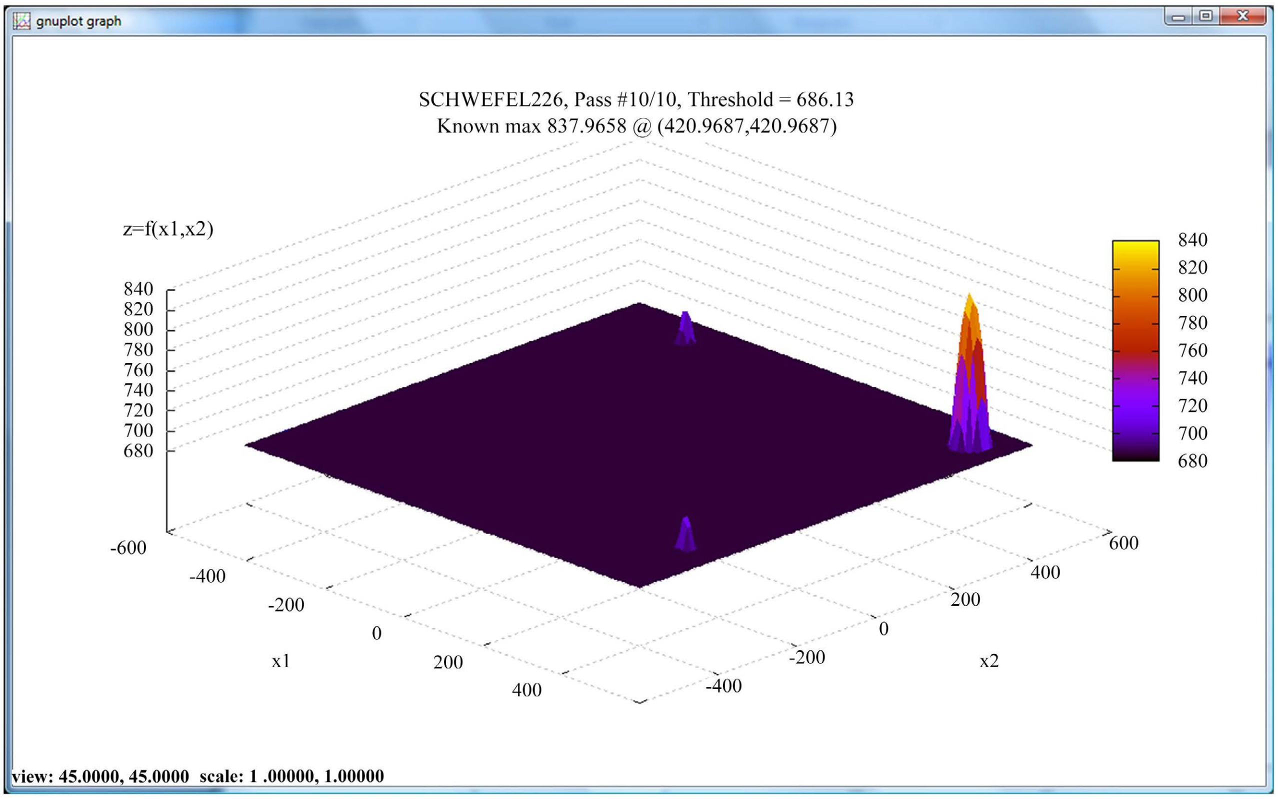
Pass #7


gnuplot graph
SCHWEFEL226, Pass #8/10, Threshold = 433.92
Known max 837.9658 @ (420.9687,420.9687)
z=f(x1,x2)
850
800
750
700
650
600
550
500
450
400
-600
-400
-200
0
200
400
x1
600
400
200
0
-200
-400
x2
view: 45.0000, 45.0000 scale: 1 .00000, 1.00000

Pass #8



Pass #9



Pass #10

Figure A3. DTO compression of 2D Schwefel Problem 2.26 with successive thresholds.

Running DTO against the 2D Schwefel 2.26 with this same deterministic CFO setup returns a best overall fitness of 837.965567554534@(420.997385258614, 420.997385258614) compared to the known maximum of 837.9658@(420.9687, 420.9687), again using 44,352 function calls because CFO this time was deterministic. The DTO-computed thresholds appear in Table A2.

A.5. Speculation

DTO is optimization algorithm-independent because it is fundamentally geometrical in nature. Procedure $OPT\left[q(\vec{x}),\vec{x}^*,q^*,q_{\min}\right]$ can be *any* global search and optimization routine, some combination of routines, different ones on successive DTO passes, or perhaps *none at all*. This observation raises the possibility of a new optimization approach that does not rely, as typically is the case, on a metaheuristic based on a metaphor drawn from Nature, or, for that matter, on any existing optimization methodology, heuristic or otherwise. Instead, it may be possible to develop a new optimization algorithm using only DTO's geometrical approach.

One possible approach might be to implement $OPT\left[q(\vec{x}),\vec{x}^*,q^*,q_{\min}\right]$ as a group of quasirandom (QR) samplings of DS at each DTO threshold (any sampling scheme can be used, but QR is attractive because these sequences are deterministic). This approach is especially attractive because of its simplicity. The data in each group could be used to develop statistics characterizing DS's topology at that threshold. Those statistics, in turn, can provide a measure of the likelihood of locating maxima. As DTO's threshold moves up, any peak at or below the floor cannot be a global maximum (unless the landscape is compressed into a plane). As the problem's topology is progressively compressed, QR sampling will return more and more sample points on the floor, that is, points at which there is no maximum of any kind. Repeatedly sampling a given threshold develops a picture of where the current maxima (local and global) might be located. In the limit, every point on the floor would be visited, and the global maxima located precisely. Of course, only a finite number of runs can be made, but it seems likely that very good statistics could be developed fairly quickly as DTO's threshold increases. At a minimum, this approach should be able to provide a reliable estimate of the likelihood of locating global maxima.

Table A2. Deterministic DTO/CFO 2-D Schwefel 2.26.

Pass #	*Threshold*
1	*none*
2	−682.382
3	−498.169
4	−331.149
5	−163.129
6	2.887

Hayes' excellent article on QR sequences [12] may provide a blueprint for a DTO-QR optimization algorithm. For example, the problem of determining the area of a leaf is analogous to computing the area under $f(\vec{x})$'s maxima (peaks) projected onto a particular DTO threshold. If the compressed DS is sampled (QR or otherwise), then $1-\frac{N_{th}}{N_s}$ estimates the probability of being within the peaks' projections (N_{th} and N_s being the number of sampling points falling *on* the threshold and the *total* number of points, respectively). Repeating this procedure *on each threshold* a sufficient number of times builds confidence in the estimate. Contrary to what might be intuitive, the objective of this approach is to reduce this probability to zero as the threshold increases. Zero probability of being within the maxima's projections corresponds to the threshold being at a global maximum because the landscape has been compressed onto a plane. If this happens, as pointed out above, all information on the maximum's location is lost, so as DTO progresses information on where maxima are found must be preserved in order to determine the global maximum's coordinates.

Besides changing DS's topology from below, statistics gathered as described above may be useful in shrinking DS from the "sides," that is, truncating $f(\vec{x})$'s domain of definition to create a smaller search space that is more easily explored. This might be accomplished by grouping proximate sample points above the threshold, that is, points within the "footprints" (projections) of local maxima, and then breaking the domain into smaller regions containing each footprint. The likelihood of locating all footprints on a given threshold increases with the number of $OPT[\cdot]$ runs made at that threshold.

Of course, all of these remarks are pure speculation at this point. Whether or not implementing some of these ideas may lead to a new and effective optimization methodology is an open question. But DTO appears to hold enough promise to be investigated further. One approach might be to initialize DTO with a deterministic algorithm such as CFO with a Probe Line IPD, because it tends to converge quickly to the vicinity of global maxima, followed by QR-based exploration as described above (or possibly a stochastic algorithm) because of potentially improved exploration.

A.6. Final Remarks

DTO appears to be an effective technique for adaptively changing the topology of the decision space in a multidimensional search and optimization problem. DTO should be useful with any search and optimization algorithm. Bounding the objective function from below removes local maxima, and as the threshold or "floor" is increased, more and more local maxima are eliminated. In the limit, the problem's landscape collapses to a plane whose value ("height") corresponds to the value of the global maximum. In that case, DS contains no information as to the global maximum's location, but the maximum's value is known precisely. In order to preserve location information, the DTO threshold should not be set too high, thereby retaining enough structure for efficient DS exploration.

There are many unanswered questions concerning how DTO should be implemented. For example, there almost certainly are better ways to set the threshold than the simple linear scheme used here. Thresholds that are progressively closer together probably will work better. Another question arises in connection with what optimization algorithm should be used. Even though DTO is algorithm-independent, it may work best when different algorithms are combined to take advantage of their different strengths and weaknesses. For example, CFO, which is inherently deterministic, often converges very quickly to the vicinity of a global maximum (good exploitation). But its very determinism inhibits exploration in decision spaces with "sparse" structure (mostly planar, few local maxima). By contrast, stochastic algorithms (for example, Particle Swarm, Ant Colony, or Differential Evolution) exhibit better exploration, but they completely lack repeatability when implemented using the true random variables in their underlying equations (computed from probability distributions). Combining a deterministic algorithm used first with a stochastic one used later may provide better results by emphasizing exploitation early in the run and exploration later in the run. Or, in the case of CFO, it might be started deterministically and then switched to stochastic mode (recall that the CFO used here was stochastic for the first 2D Schwefel 2.26 run and deterministic for the subsequent 30D/2D cases). Another improvement might utilize "lateral" DS compression on one of DTO's thresholds. It may be possible in the DTO-compressed landscape to reliably determine the global maximum's approximate location and based on that information shrink DS "from the sides" or "laterally" (reduce the domain of definition), making it easier to search the smaller DS. If DTO is a novel approach to optimization, as the author believes it is, then all of these possibilities merit consideration as fruitful areas of research, and the author hopes that these remarks will encourage such work.