The combined use of a genetic algorithm and the hill-climbing algorithm to find difference triangle sets

Sharon Koubi[†] and Nabil Shalaby[‡]
[†] Department of Computer Science
Memorial University of Newfoundland
Sharon_Koubi@yahoo.com

[‡] Department of Mathematics and Statistics
Memorial University of Newfoundland
nshalaby@math.mun.ca

February 5, 2008

Abstract

In this paper we use a genetic algorithm and direct a hill-climbing algorithm in choosing differences to generate solutions for difference triangle sets. The combined use of the two algorithms optimized the hill-climbing method and produced new improved upper bounds for difference triangle sets.

1 Introduction

An (n, k)-difference triangle set, or $(n, k) - D\Delta S$, is a set $X = \{X_i \mid 1 \leq i \leq n\}$, where $X_i = \{a_{ij} \mid 0 \leq j \leq k\}$, for $1 \leq i \leq n$ are sets of integers called *blocks*, such that all the differences $a_{ij} - a_{ij'}$ for $1 \leq i \leq n$ and $0 \leq j \neq j' \leq k$, are all distinct and non-zero. By ordering the elements of X_i and subtracting the smallest, we get an $(n, k) - D\Delta S$ in normalized form:

$$0 = a_{i0} < a_{i1} < \cdots < a_{ik} \quad \text{for all} \quad 1 \leq i \leq n.$$

All difference triangle sets considered in this correspondence are normalized. The name "difference triangle" arises for writing the differences in a triangular format. The following example shows $(4,3)-D\Delta S$ and its differences in a triangular format:

$\{0,4,15,24\}$	$\{0,6,22,23\}$	$\{0,2,14,21\}$	$\{0,10,13,18\}$
4 11 9	6 16 1	2 12 7	10 3 5
15 20	33 17	14 19	13 8
24	23	21	18

Let $m = m(X)$, the *maximum difference* or the *scope* of an $(n,k)-D\Delta S$, be defined as $m(X) = \max\{a \mid a \in \cup_{i=1}^{n} X_i\}$. The smallest possible scope for an $(n,k)-D\Delta S$ is $m(n,k) = \min\{m(X) \mid X$ is an $(n,k)-D\Delta S\}$. If for some $(n,k)-D\Delta S$ $m(X) = m(n,k)$ then X is called *optimal*. Difference triangle sets have some interesting applications in data communications; it is desirable for these applications to have difference triangle sets with small scopes. Kløve gave tables for upper and lower bounds for scopes of triangle sets. Lorentzen and Nilsen [9] and Shearer [10] were able to find improved lower bounds using a liner programming method. Shearer [11], Ling [8] and Koubi et al [7] improved the upper bounds using computational methods. In this paper we present difference triangle sets, found by computer search using the hill-climbing algorithm, and a genetic algorithm to optimize the hill-climbing search and improved its results.

2 Hill-Climbing Algorithm

We describe in this section the hill-climbing algorithm, a non-exhaustive heuristic. The general idea of the hill-climbing algorithm is to attempt to reach a maximum by generating possible solutions and accepting only those solutions $S \in \sum$, a set T_S of transformations, each of which can be used to change S into another feasible solution S'. The set of solutions that can be reached from S by applying a transformation from T_S is the neighbourhood $N(S)$ of S. S is a local minimum if $c(S) \leq c(S')$ for all $S' \in N(S)$.

In [13] Stinson used the hill-climbing algorithm to find Steiner triple systems (STS) using the hill-climbing heuristic. In [7] Koubi et al described the use of a directed hill-climbing algorithm that improves the upper bounds on some difference triangle sets. The general framework of the algorithm is given as follows:

Hill-climbing for generating triangle sets:
```
        begin
        $s = 0$
        $X = \emptyset$
```

```
while $s < n$ or maximum number of iterations is exceeded
        Generate $B$, a random block of $k+1$ distinct integers
        If $X \cap B$ is an incomplete $(n,k) - D\Delta S$
                $X = X \cap B$
                $s = s + 1$
        else
                Find (if exists) $B_1 \in X$ such that if $(X \cap$
                $\{B_0\}) - \{B_1\}$ is an incomplete $(n,k) - D\Delta S$
                If $B_1$ exists then set $X = (X \cap \{B_0\}) - \{B_1\}$
        end
end
```

The algorithm is directed by the application of two heuristics. The first heuristic method is based on applying the hill-climbing method also in the process of generating a candidate block. In order to minimize conflicts, the algorithm tries to generate a block that does not conflict with any of the other blocks already generated.

The second heuristic optimizes the selection of the random number that is generated in every iteration when constructing a block. The heuristic is based on experimental observations regarding the distribution of the numbers in each position in a block. According to the measurements, the standard deviations for the elements in the middle of the block tend to be higher that those for elements near the edges. This could be explained by observing that a number close to the middle of the range can be used to obtain only about half the number of unique differences that can be obtained from a number that is near the extremes. The heuristic controls the random generation of new elements according to these observations.

As it is described in the next section, the application of the directed hill-climbing algorithm directed by a genetic algorithm produced new improved upper bounds for difference triangle sets.

3 Genetic Algorithm

Genetic algorithms were introduced by Holland [4]. They work by considering a set of candidates (a population) rather than one candidate; the population may sometimes contain duplicates. The aim is to maximize a function called *fitness*. The notations and ideas are borrowed from genetics and evolution. The candidates are called *chromosomes*, and represented as sequences $(x_1, x_2, \ldots, x_k)$, where each variable x_i has a small range of possible values. At each stage a new population (next generation) is constructed from the previous one using three operations. The first operation is selection by which candidates are selected randomly according to a fitness function. The second is crossover in which the selected chromosomes are randomly assigned to pairs and crossed over. The third operation is mutation. A probability q is specified, and any gene in any chromosome has a probability of q to be changed in value. In [1] Ashlock described the use of a genetic algorithm to generate combinatorial designs. His approach applied to genetic algorithm directly to the combinatorial problem.

From our experiments we did not find genetic algorithms to be efficient for the generation of difference triangle sets. The approach taken in this paper uses a genetic algorithm to direct the hill-climbing algorithm. The hill-climbing algorithm described in [7] is based on two heuristics. The second heuristic of the algorithm uses a large number of parameters. The genetic algorithm was used in order to help determine these parameters and direct the hill-climbing algorithm. It was also used as a mean to automate the determination of the parameters to be used. Let S be a collection of triangle sets of scope M. Let $B = \{X_i \mid X_i \in X \text{ and } X \in S\}$, the collection of all the blocks of the triangle sets of S. Let $D_j = \{a_{ij} \mid a_{ij} \in X_i$ and $X_i \in B\}$, the collection of all the elements from all the blocks that are in position j where $1 \leq j \leq k$. As explained in [7] our hypothesis was that the distribution of the elements in D_j is approximately described by $N(\sigma_j, \mu_j)$ with mean μ_j and standard deviation σ_j, where $\mu_j = \left(\frac{M}{k}\right) j$. Therefore, the parameters for the second heuristic are pairs (σ_j, μ_j) where $1 \leq j \leq k$. In our previous work the values to parameters were determined by analyzing a large number of difference triangle sets with a higher scope. In this work we used a genetic algorithm to find if a better assignment is possible. The general description of the genetic search algorithm is as shown:

Genetic Search

1. Generate a random population of sets of parameters (a parameter set is k sets of pairs (σ_j, μ_j).
2. Repeat steps 2 to 7 for a fixed number of generations.
3. Evaluate the fitness of each set by using them as input to the hill-climbing algorithm and finding the number of blocks that are generated.
4. Select pairs of sets (parents) with a fitness bias.
5. Pairs of parents use crossover to produce pairs of children.
6. The children are mutated.

K	5	6	7	8	9	10	11	12	13	14	15
5						165†		198†			
6					225†			305†			
7			251‡	287‡	323‡	363‡	400‡	435‡	474‡		
8	244‡	294‡									
9	319‡										

Figure 1: Improved upper bounds for $(n, k) - D\Delta S$. † Previous bounds found in [7]. ‡ Previous bounds found in [10].

7. Children replace the parent sets in the current population.

The genetic algorithm is used to optimize the results of the hill-climbing algorithm by determining the best set of parameters to be used. It also benefits the process by automating the selection of appropriate parameters thus allowing to scan a larger number of possibilities in less time. By using the genetic algorithm we were able to find many new results for difference triangle sets. In Figure 1 we give a summary of the improvements to upper bounds achieved by the hill-climbing algorithm directed by the genetic algorithm. The new triangle sets found are given in the appendix. The results in Figure 1 were obtained by employing an implementation of the combined hill-climbing/genetic algorithms. Each case was executed in parallel on a network of 20 computers running Linux with a 1.8Mhz Pentium processor. A maximum running time of 30 hours was given for each case.

4 Conclusion

In [7] we showed that the hill-climbing algorithm is useful for finding difference triangle sets. In this work we described how we further improved the results attained by the hill-climbing algorithm by directing it using a genetic algorithm. This combination yielded improved results.

References

[1] D. Ashlock, "Finding Designs with Genetic Algorithms", Chapter in "Computational and Constructive Design Theory", edited by W.D Wallis, Kluwer Press, 1996.

[2] Y.M. Chee and C.J. Colbourn, "Constructions for Difference Triangle Sets", IEEE Trans. Inform. Theory, vol. 43, pp. 1346-1349, 1997.

[3] C.J. Colbourn, "Difference Triangle Sets", Chapter in The CRC Handbook of Combinatorial Designs by C.J. Colbourn and J. Dintz (ISBN 0-8493-8948-8), pp. 312-317, 1996.

[4] J.H. Holland, "Adaptation in Natural and Artificial Scientific Systems", The MIT Press, Cambridge Massachusetts, 1994.

[5] T. Kløve, "Bounds on the size of optimal difference sets," IEEE Trans. Inform. Theory, vol. 34, pp. 355-361, 1988.

[6] T. Kløve, "Bounds and constructions for difference triangle sets," IEEE Trans. Inform. Theory, vol. 35, pp. 879-886, 1989.

[7] S. Koubi, M. Mata-Montero and N. Shalaby, "Using Directed Hill-Climbing for the Construction of Difference Triangle Sets", IEEE Trans. Inform. Theory, vol. 51, pp. 331-335, 2005.

[8] A. Ling, "Difference Triangle Sets From Affine Planes", IEEE Trans. Inform. Theory, vol. 48, pp. 2399-2401, 2002.

[9] R. Lorentzen and R. Nilsen, "Application of Linear Programming to the Optimal Difference Triangle Set Problem", IEEE Trans. Inform. Theory, vol. 37, pp. 1486-1488, 1991.

[10] J.B. Shearer, "Some New Difference Triangle Sets", The Journal of Combinatorial Mathematics and Combinatorial Computing, vol. 27, pp. 65-76, 1998.

[11] J.B. Shearer, "Improved LP lower bounds for difference triangle sets", Electronic Journal of Combinatorics, vol. 6(1), #R31, 1999.

[12] J.B. Shearer, "Difference triangle sets - upper bounds", http://www.research.ibm.com/people/s/shearer/dtsub.html#js3.

[13] D.R. Stinson, "Hill-Climbing Algorithms for the Construction of Combinatorial Designs", Annals of Discrete Mathematics, vol. 26, pp. 321-324, 1985.

K	*N*	Difference Sets
5	10	{0,11,34,81,120,165}, {0,18,43,107,114,156}, {0,15,48,76,145,158}, {0,4,31,87,139,159}, {0,8,59,85,126,150}, {0,22,32,112,133,147}, {0,6,36,99,104,157}, {0,9,38,88,161,162}, {0,16,19,94,148,160}, {0,17,57,117,119,163}
5	12	{0,6,48,120,145,198}, {0,3,64,144,146,193}, {0,17,45,121,178,194}, {0,7,90,124,165,189}, {0,9,79,100,171,185}, {0,1,69,102,112,196}, {0,22,62,74,125,181}, {0,4,81,113,168,179}, {0,20,39,128,174,187}, {0,26,31,136,163,186}, {0,8,38,96,131,191}, {0,18,54,140,169,184}
6	9	{0,5,46,132,188,190,225}, {0,17,61,100,130,201,214}, {0,6,24,118,145,222,223}, {0,8,33,97,131,203,213}, {0,7,45,133,148,196,218}, {0,16,67,90,182,210,224}, {0,21,50,102,159,212,221}, {0,11,47,107,172,175,215}, {0,12,32,87,91,167,186}
6	12	{0,4,48,142,214,225,305}, {0,23,70,102,175,281,303}, {0,30,43,147,232,283,292}, {0,2,40,115,218,224,300}, {0,5,69,126,153,261,282}, {0,7,78,119,174,242,273}, {0,18,95,161,215,268,302}, {0,1,98,101,159,270,289}, {0,15,50,74,205,254,291}, {0,29,46,162,229,255,294}, {0,16,28,165,198,279,287}, {0,10,62,150,186,206,296}
7	7	{0,9,40,126,153,186,221,251}, {0,2,61,76,124,163,173,243}, {0,12,38,67,145,188,229,245}, {0,7,18,89,147,155,232,238}, {0,19,44,81,134,185,213,237}, {0,4,36,92,109,210,223,244}, {0,3,23,45,139,203,249,250}
7	8	{0,15,60,95,144,186,249,287}, {0,17,23,88,163,221,261,273}, {0,13,32,136,145,147,228,275}, {0,3,59,85,90,197,240,267}, {0,28,44,78,152,203,269,279}, {0,7,29,93,148,187,259,283}, {0,14,67,68,167,188,224,285}, {0,4,37,106,168,209,257,282}
7	9	{0,10,86,107,189,305,309,323}, {0,40,42,127,175,285,301,320}, {0,38,94,109,182,199,303,312}, {0,6,63,163,210,232,294,302}, {0,13,52,72,153,228,277,321}, {0,1,68,151,196,292,315,318}, {0,12,65,111,190,248,298,322}, {0,37,64,155,166,244,272,304}, {0,25,80,123,266,271,300,307}
7	10	{0,12,46,178,242,265,296,363}, {0,21,41,154,186,258,314,342}, {0,1,43,140,202,252,287,348}, {0,3,52,120,194,239,297,344}, {0,7,115,141,210,276,336,346}, {0,5,18,111,162,225,300,325}, {0,33,71,88,170,271,349,360}, {0,14,29,123,182,247,304,357}, {0,24,40,131,167,204,283,359}, {0,4,81,90,129,173,303,322}

7	11	{0, 10, 15, 217, 244, 277, 328, 400}, {0, 3, 73, 79, 249, 261, 367, 375}, {0, 13, 62, 149, 254, 270, 347, 382}, {0, 21, 64, 150, 259, 290, 337, 378}, {0, 39, 61, 97, 230, 264, 332, 388}, {0, 1, 19, 144, 174, 305, 355, 395}, {0, 4, 63, 193, 222, 231, 344, 370}, {0, 7, 98, 153, 178, 310, 338, 358}, {0, 17, 69, 179, 232, 306, 317, 398}, {0, 24, 66, 120, 220, 319, 321, 365}, {0, 14, 37, 108, 223, 280, 312, 387}
7	12	{0, 3, 16, 128, 340, 416, 420, 435}, {0, 52, 53, 150, 237, 284, 408, 414}, {0, 23, 94, 210, 316, 371, 415, 433}, {0, 2, 74, 194, 230, 343, 352, 409}, {0, 5, 107, 216, 249, 331, 396, 406}, {0, 26, 63, 114, 259, 305, 423, 431}, {0, 41, 58, 261, 296, 303, 364, 434}, {0, 14, 25, 165, 195, 327, 358, 412}, {0, 21, 49, 69, 235, 312, 336, 403}, {0, 34, 56, 189, 253, 282, 332, 428}, {0, 43, 83, 121, 202, 329, 402, 429}, {0, 32, 92, 137, 176, 266, 377, 389}
7	13	{0, 30, 59, 197, 212, 220, 356, 474}, {0, 16, 38, 49, 162, 227, 434, 458}, {0, 34, 62, 115, 260, 399, 401, 470}, {0, 14, 93, 201, 242, 375, 394, 457}, {0, 17, 117, 192, 288, 362, 422, 465}, {0, 25, 126, 132, 261, 334, 391, 446}, {0, 7, 61, 183, 275, 302, 433, 459}, {0, 1, 128, 223, 322, 332, 374, 472}, {0, 12, 80, 217, 293, 328, 359, 449}, {0, 5, 88, 160, 257, 346, 430, 466}, {0, 18, 50, 166, 253, 330, 432, 453}, {0, 39, 85, 125, 354, 363, 410, 468}, {0, 37, 143, 147, 191, 387, 390, 454}
8	5	{0, 18, 54, 68, 98, 170, 193, 197, 244}, {0, 6, 11, 49, 82, 194, 213, 223, 235}, {0, 2, 28, 113, 120, 182, 206, 227, 243}, {0, 9, 48, 90, 103, 156, 181, 239, 240}, {0, 15, 35, 67, 75, 132, 163, 233, 236}
8	6	{0, 4, 12, 118, 155, 157, 266, 279, 294}, {0, 6, 16, 35, 82, 103, 223, 253, 280}, {0, 9, 43, 99, 158, 222, 239, 285, 292}, {0, 1, 72, 105, 166, 184, 235, 259, 273}, {0, 11, 55, 78, 128, 211, 214, 263, 288}, {0, 26, 58, 80, 142, 190, 250, 255, 286}
9	5	{0, 11, 53, 88, 90, 207, 214, 275, 300, 318}, {0, 8, 36, 80, 121, 202, 233, 256, 285, 304}, {0, 4, 14, 69, 103, 142, 169, 243, 292, 301}, {0, 5, 50, 51, 157, 195, 221, 251, 291, 313}, {0, 20, 33, 115, 130, 147, 206, 290, 293, 314}

Corrigendum/Addendum to: Almost resolvable 4-cycle systems

I. J. Dejter, C. C. Lindner, M. Meszka and C. A. Rodger

In [1], some typographical errors appear in Example 2.1. Below we correct these errors, and using this almost resolvable 4-cycle system of order 17, an almost resolvable 4-cycle system of order 33 is constructed[1]. Order 33 is one of the missing cases in [1]; the cases 41 and 57 remain open.

In the correction below, symbol 15 has been replaced by ∞, and 16 by 15, so the vertex set is now $\{0, 1, \ldots, 14, 15\} \cup \{\infty\}$.

Corrected Example 2.1 from [1]:

$\{(2,5,12,9)$,	$(3,6,13,10)$,	$(4,7,14,11)$,	$(\infty,1,8,15)\}$;
$\{(4,5,13,14)$,	$(6,9,15,7)$,	$(2,10,0,3)$,	$(\infty,8,11,12)\}$;
$\{(10,11,2,1)$,	$(13,4,12,3)$,	$(15,6,14,0)$,	$(\infty,5,8,7)\}$;
$\{(12,13,15,2)$,	$(3,11,1,4)$,	$(9,0,7,10)$,	$(\infty,6,5,14)\}$;
$\{(7,2,13,1)$,	$(11,6,0,5)$,	$(4,15,10,8)$,	$(\infty,3,14,9)\}$;
$\{(10,5,9,4)$,	$(2,14,8,6)$,	$(3,1,12,7)$,	$(\infty,11,0,13)\}$;
$\{(1,5,3,15)$,	$(13,11,9,7)$,	$(8,2,0,12)$,	$(\infty,4,6,10)\}$;
$\{(9,13,8,3)$,	$(7,5,15,11)$,	$(14,12,6,1)$,	$(\infty,0,4,2)\}$;
$\{(0,1,9,8)$,	$(12,15,14,10)\}$.		

Addendum: an almost resolvable 4-cycle system of order 33
The vertex set is $\{0, 1, \ldots, 14, 15\} \cup \{0', 1', \ldots, 14', 15'\} \cup \{\infty\}$.

Sixteen almost parallel classes, and one short class, are:

[1]Found by Elizabeth J. Billington, the University of Queensland, Australia.

$\{(2,5,12,9)$,	$(2',5',12',9')$,	$(3,6,13,10)$,	$(3',6',13',10')$,
$(4,7,14,11)$,	$(4',7',14',11')$,	$(0',1,8,15)$,	$(\infty,1',8',15')\}$;
$\{(1,\infty,15,2')$,	$(8,8',12,12')$,	$(5,4',7,10')$,	$(6,5',14,7')$,
$(3,11',9,13')$,	$(11,6',13,14')$,	$(4,1',10,15')$,	$(0,3',2,9')\}$;
$\{(4,5,13,14)$,	$(4',5',13',14')$,	$(6,9,15,7)$,	$(6',9',15',7')$,
$(2,10,0,3)$,	$(2',10',0',3')$,	$(1',8,11,12)$,	$(\infty,8',11',12')\}$;
$\{(1,8',15,12')$,	$(8,\infty,12,15')$,	$(5,5',7,7')$,	$(6,6',14,14')$,
$(3,4',9,10')$,	$(11,3',13,9')$,	$(4,11',10,13')$,	$(0,0',2,2')\}$;
$\{(10,11,2,1)$,	$(10',11',2',1')$,	$(13,4,12,3)$,	$(13',4',12',3')$,
$(15,6,14,0)$,	$(15',6',14',0')$,	$(9',5,8,7)$,	$(\infty,5',8',7')\}$;
$\{(1,1',15,15')$,	$(8,5',12,7')$,	$(5,3',7,\infty)$,	$(6,0',14,2')$,
$(3,6',9,14')$,	$(11,11',13,13')$,	$(4,4',10,10')$,	$(0,8',2,12')\}$;
$\{(12,13,15,2)$,	$(12',13',15',2')$,	$(3,11,1,4)$,	$(3',11',1',4')$,
$(9,0,7,10)$,	$(9',0',7',10')$,	$(8',6,5,14)$,	$(\infty,6',5',14')\}$;
$\{(1,5'.15,7')$,	$(8,6',12,14')$,	$(5,1',7,15')$,	$(6,\infty,14,12')$,
$(3,3',9,9')$,	$(11,4',13,10')$,	$(4,0',10,2')$,	$(0,11',2,13')\}$;
$\{(7,2,13,1)$,	$(7',2',13',1')$,	$(11,6,0,5)$,	$(11',6',0',5')$,
$(4,15,10,8)$,	$(4',15',10',8')$,	$(12',3,14,9)$,	$(\infty,3',14',9')\}$;
$\{(1,6',15,14')$,	$(8,4',12,10')$,	$(5,11',7,13')$,	$(6,3',14,9')$,
$(3,\infty,9,8')$,	$(11,0',13,2')$,	$(4,5',10,7')$,	$(0,1',2,15')\}$;
$\{(10,5,9,4)$,	$(10',5',9',4')$,	$(2,14,8,6)$,	$(2',14',8',6')$,
$(3,1,12,7)$,	$(3',1',12',7')$,	$(15',11,0,13)$,	$(\infty,11',0',13')\}$;
$\{(1,4',15,10')$,	$(8,0',12,2')$,	$(5,8',7,12')$,	$(6,11',14,13')$,
$(3,5',9,7')$,	$(11,\infty,13,1')$,	$(4,3',10,9')$,	$(0,6',2,14')\}$;
$\{(1,5,3,15)$,	$(1',5',3',15')$,	$(13,11,9,7)$,	$(13',11',9',7')$,
$(8,2,0,12)$,	$(8',2',0',12')$,	$(14',4,6,10)$,	$(\infty,4',6',10')\}$;
$\{(1,11',15,13')$,	$(8,3',12,9')$,	$(5,0',7,2')$,	$(6,4',14,10')$,
$(3,1',9,15')$,	$(11,8',13,12')$,	$(4,\infty,10,6')$,	$(0,5',2,7')\}$;
$\{(9,13,8,3)$,	$(9',13',8',3')$,	$(7,5,15,11)$,	$(7',5',15',11')$,
$(14,12,6,1)$,	$(14',12',6',1')$,	$(10',0,4,2)$,	$(\infty,0',4',2')\}$;
$\{(1,3',15,9')$,	$(8,11',12,13')$,	$(5,6',7,14')$,	$(6,1',14,15')$,
$(3,0',9,2')$,	$(11,5',13,7')$,	$(4,8',10,12')$,	$(0,\infty,2,4')\}$.
$\{(0,1,9,8)$,	$(12,15,14,10)$,	$(0',1',9',8')$,	$(12',15',14',10')\}$.

REFERENCE

[1] I. J. Dejter, C. C. Lindner, M. Meszka and C. A. Rodger, Almost resolvable 4-cycle systems, *J. Combin. Math. Combin. Computing* 63 (2007), 173–182.