APPENDIX A: FIRST STEPS ON SIMULATOR

The simulator was compiled and tested on Mac OS X Snow Leopard, Mac OS X Lion and Linux Ubuntu. It requires pthreads to work and it is highly recommended to use the Python wrapper and iPython as your command-line interface. Another very useful library is pylab, matplotlib and scipy, since they have a lot of tools for statistics and charting.

Code is licensed under GPL License[32] and can be downloaded from my personal website *http://msbrogli.vialink.com.br/*. Installing SDM simulator is very easy. Go to root directory, run `make` and wait for compile. Go to directory "python" and you should see a file named *libsdm.so*. It's done.

I believe the best and easiest way to learn is by example. So, I did this simple tutorial in lines of codes. Comments explains what is being done, lines starting with ">>>" are commands and other lines are outputs.

Get ready and try some coding!

A.1 HAVING SOME FUN WITH BITSTRINGS

```
>>> # load sdm wrapper
>>> import sdm

>>> # check number of dimension
>>> sdm._dimension
c_int(1000)

>>> # check access radius
>>> sdm._radius
c_int(451)

>>> # check number of hard-locations
>>> sdm._sample
c_int(1000000)

>>> # initialize sdm
>>> # you can change dimension, radius and hard-locations through:
>>> # set_dimension(d), set_radius(r), set_sample(s)
>>> sdm.initialize()

>>> # generating random bitstring
>>> a = sdm.Bitstring()
>>> b = sdm.Bitstring()

>>> # calculating distance between a and b
>>> a.distance_to(b)
491
>>> b.distance_to(a)
491

>>> # generating a zero bitstring (all bits zero)
>>> z = sdm.Bitstring(zero=True)

>>> # how many 1's have a and b?
```

```
>>> a.distance_to(z)
512
>>> b.distance_to(z)
499

>>> # can I see all bits of a?
>>> str(a)
'0001101010101110100011011101110101011110111010100110100001010111110
0001111010111001110110110110101100001000101111110111100100001000001
01011111001100011101010001111011011110110010111011100100110000110011
10100001010110010110111011011101011010010111000001101110110010010010
1101011010110011111000100000010000101100001000101000000110100111101
1111101101000011101011101000011010110000110101101001111111000011100
0011000001011011100110111011010010111000010110010100000010011001000
0101000000111110101010011011110010000110010101100110101101011000110
0000111011000011101001101011111011000010010110001011010010110000101
1001101011111100100111011111111000110000000101011101100010101111111
1111010001110101001001000111101111001100001010101011010010110101010
0000011000110101110000100111100101001111100010101010110001100010011
0110100110011000110101100010000001001101000101101101101110111010000
1011011111011000011111111100000010110101110011111101101000100001001
0101010001100010011111110000100001111101010010101111000111111000'

>>> # how many hard-locations will be activated reading or writing?
>>> sdm.thread_radius_count(a)
1108
>>> sdm.thread_radius_count(b)
1059

>>> # how many hard-locations are common to bitstring a and b?
>>> # this result probably will be different in your case
>>> sdm.thread_radius_count_intersect(a, b)
2

>>> # generate a histogram of number of common hard-locations to
>>> # random bitstrings
>>> # ps: a window will open in your computer.
>>> v = [ sdm.thread_radius_count_intersect(sdm.Bitstring(), sdm.
    Bitstring()) for i in xrange(1000) ]
>>> import pylab
>>> pylab.hist(v, bins=max(v)-min(v), normed=True)

>>> # destroy sdm
>>> sdm.free()
```

A.2 SIMPLE WRITING AND READING

There are two functions for reading and writing: "write", "read", "thread_read" and "thread_write". The first two functions don't parallels the search for active hard-locations, the later functions uses threads to distribute the search. I strongly recommend always using the "thread_" functions.

```
>>> # load sdm wrapper
>>> import sdm

>>> # check number of threads that will be used
>>> sdm._thread_count
4

>>> # initialize sdm
>>> sdm.initialize()
```

```
>>> # our memory is empty.
>>> # let's write first bitstring.
>>> # first argument is address and second is data.
>>> # remember that we use data as address, so both are equal.
>>> # write method return the number of hard-locations activated
    during execution.
>>> a = sdm.Bitstring()
>>> sdm.thread_write(a, a)
1108

>>> # let's read from a
>>> b = sdm.thread_read(b)
>>> a.distance_to(b)
0
>>> str(a) == str(b)
True

>>> # as our memory has only one bitstring, we except
>>> # to read this bitstring even 300 bits away
>>> c = a.copy()          # copy bitstring
>>> c.bitrandomswap(300)  # c is 300 bits away from a
>>> a.distance_to(c)
300
>>> a.distance_to(sdm.thread_read(c))
0

>>> # let's write some random bitstring.
>>> for i in xrange(1000):
>>>     x = sdm.Bitstring()
>>>     sdm.thread_write(x, x)
1131
1032
1099
999
1037
980
:
:

>>> # let's try again reading at c
>>> a.distance_to(sdm.thread_read(c))
272

>>> # the distance is less than original c distance to a
>>> # so, if we do an iterative-reading we will converge
>>> b1 = sdm.thread_read(c)
>>> a.distance_to(b1)
272
>>> b2 = sdm.thread_read(b1)
>>> a.distance_to(b2)
229
>>> b3 = sdm.thread_read(b2)
>>> a.distance_to(b3)
105
>>> b4 = sdm.thread_read(b3)
0
>>> str(a) == str(b4)
True

>>> # destroy sdm
>>> sdm.free()
```

B

APPENDIX B: ADDITIONAL ITERATIVE READING RESULTS FOR 1000-DIMENSIONAL MEMORY

For the sake of completeness, in this appendix we include the entire set of figures generated with 1000 dimensions, a single write of target bitstring, ranging from 1 to 40 iterative-readings.

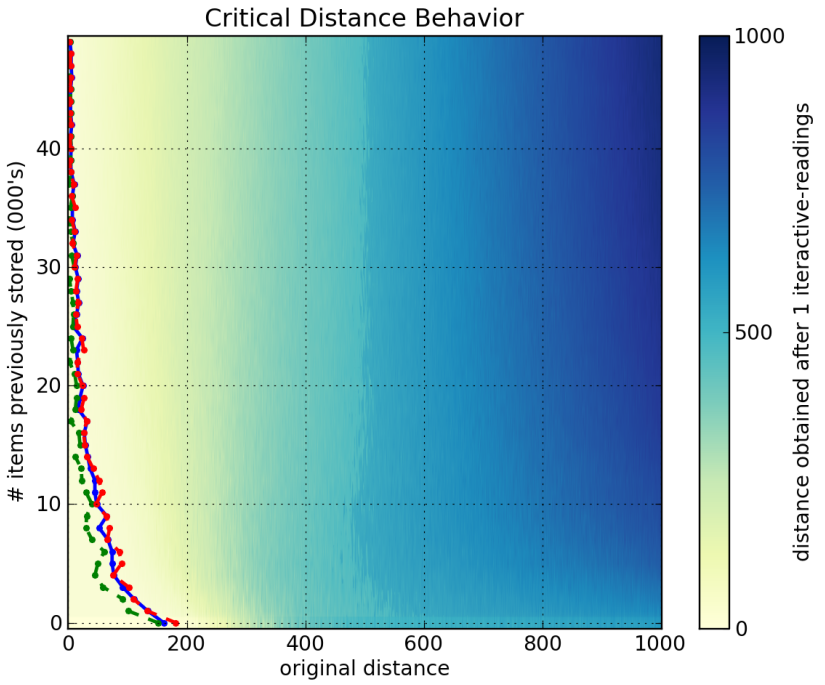

Figure 13: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 1 iterative-reading

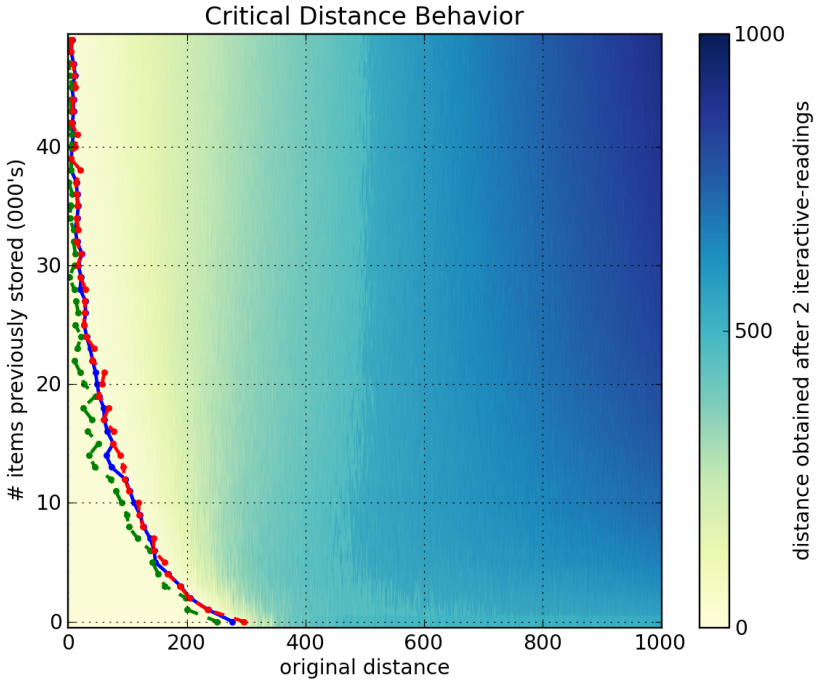

Figure 14: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 2 iterative-reading

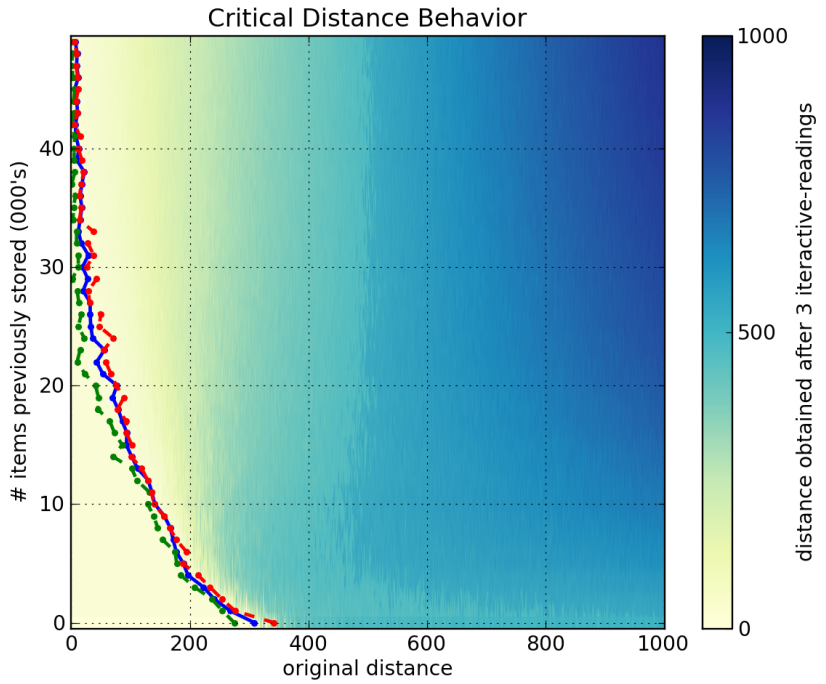

Figure 15: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 3 iterative-reading

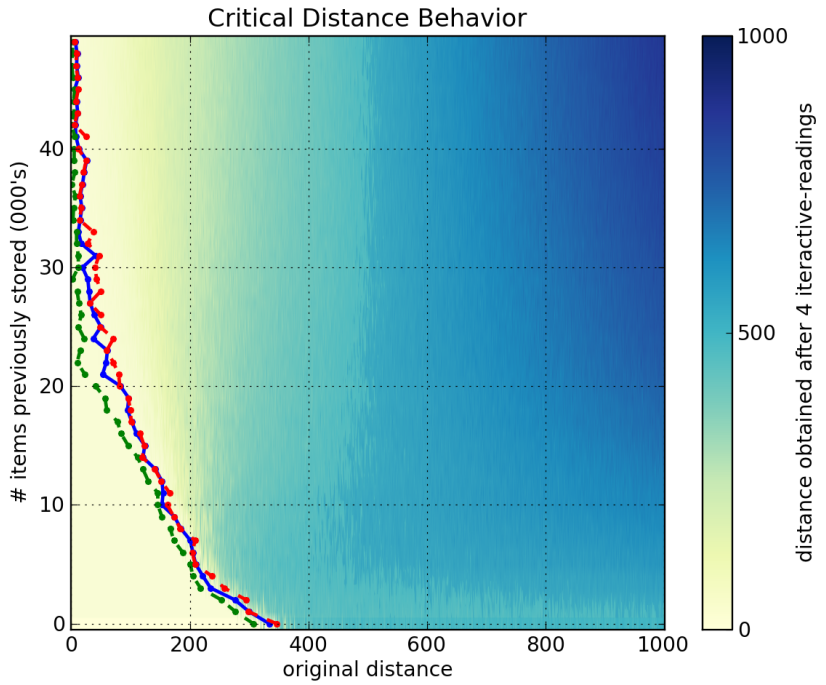

Figure 16: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 4 iterative-reading

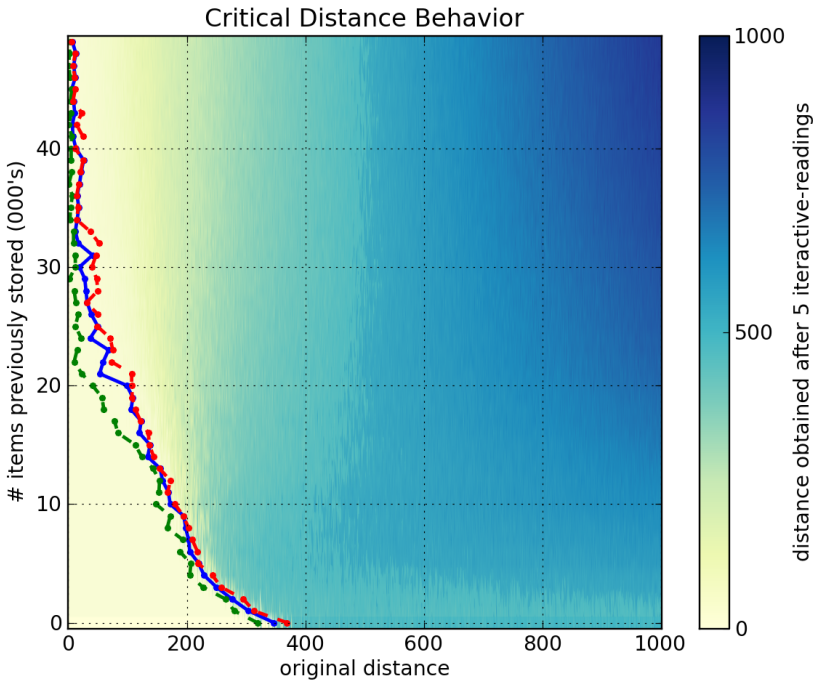

Figure 17: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 5 iterative-reading

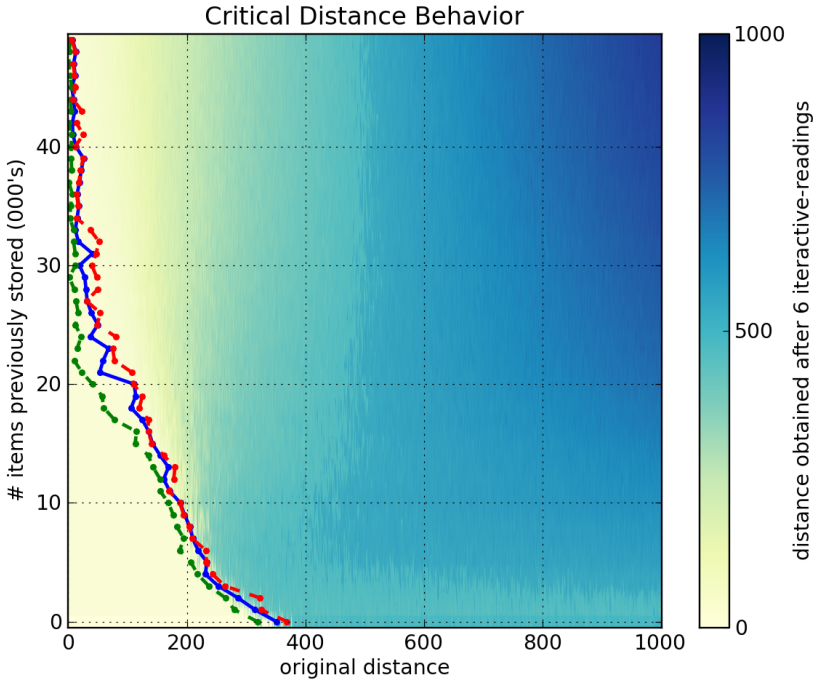

Figure 18: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 6 iterative-reading

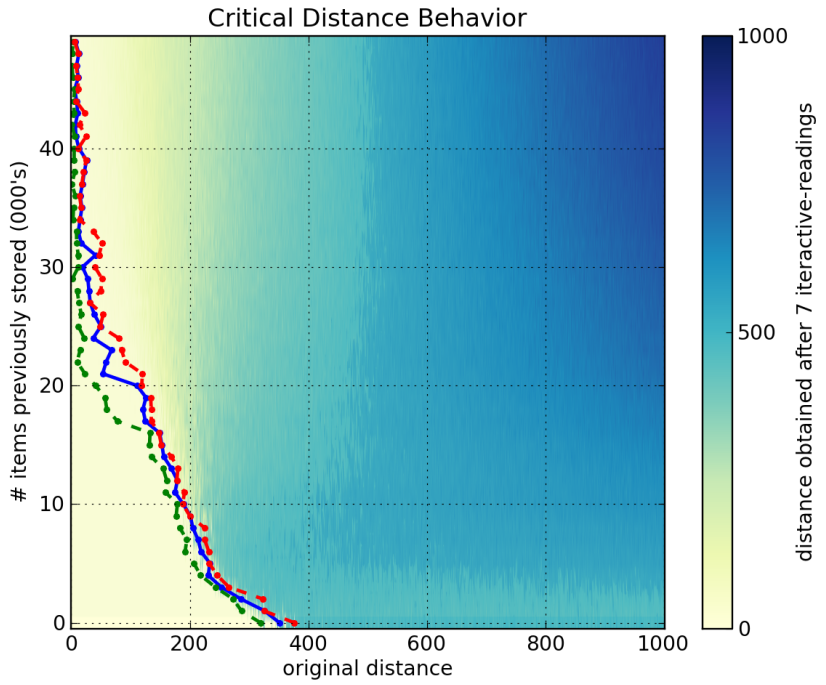

Figure 19: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 7 iterative-reading

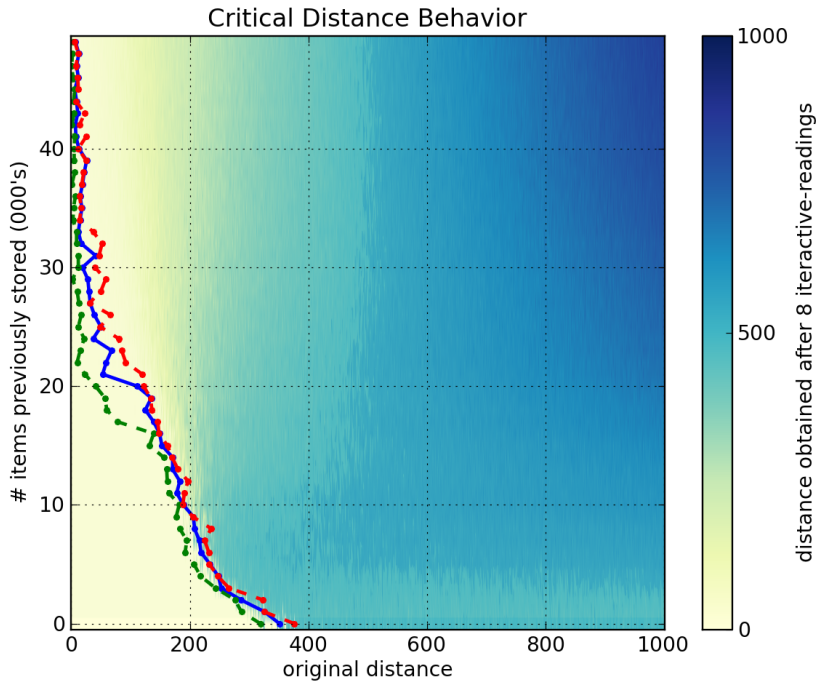

Figure 20: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 8 iterative-reading

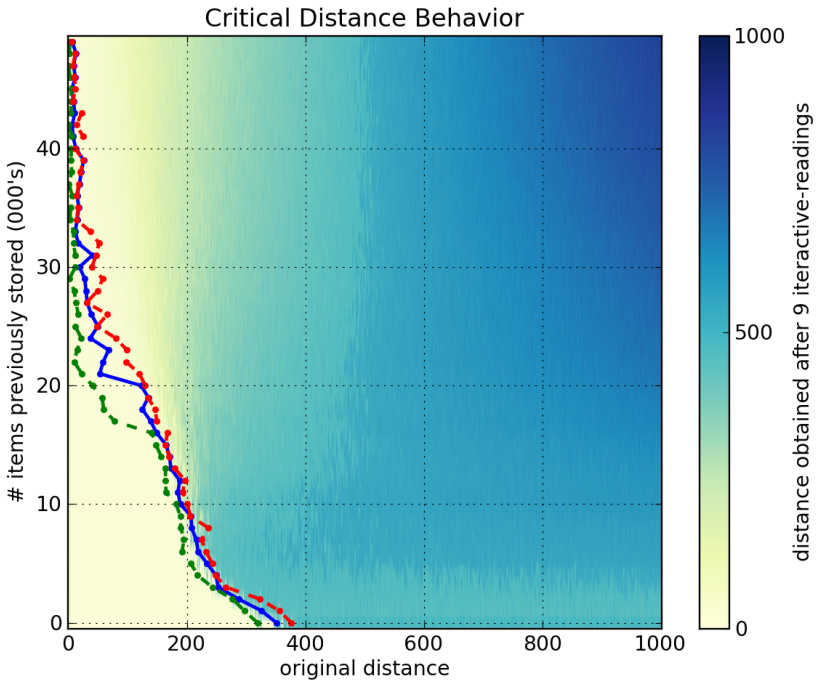

Figure 21: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 9 iterative-reading

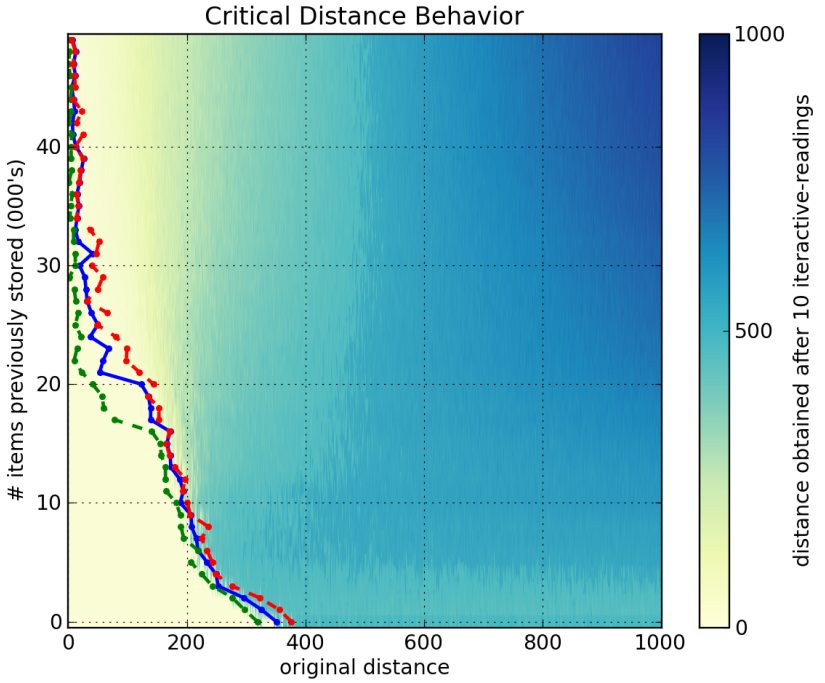

Figure 22: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 10 iterative-reading

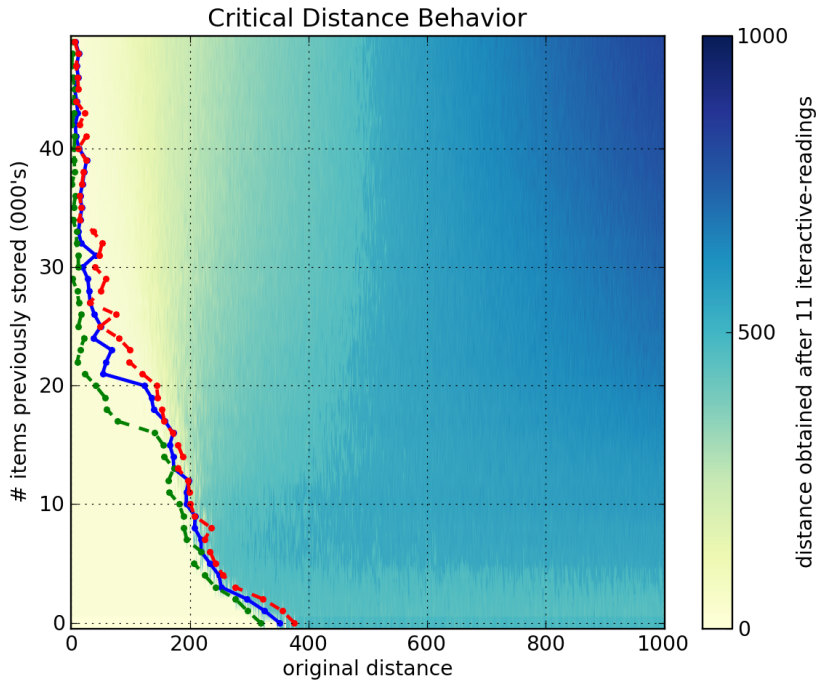

Figure 23: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 11 iterative-reading

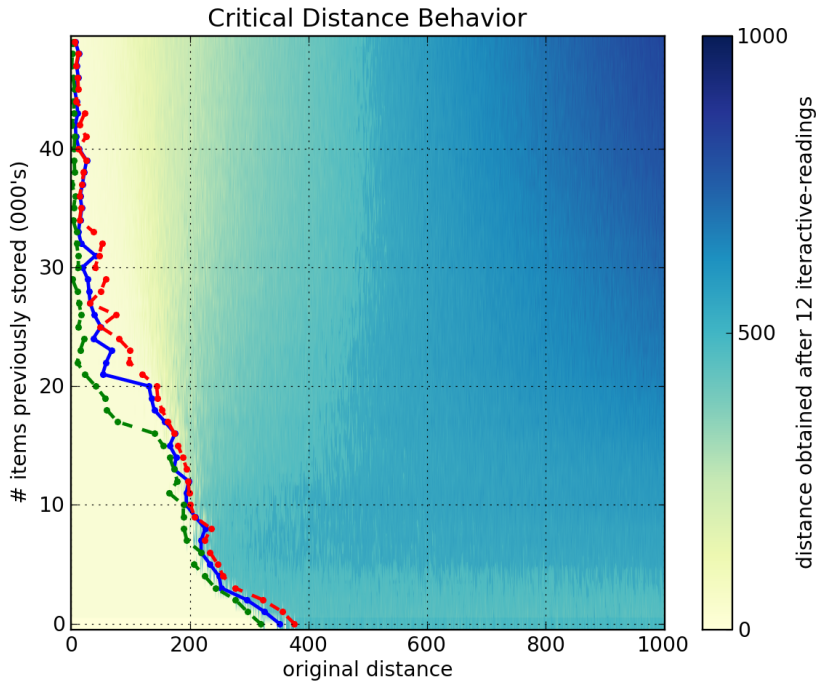

Figure 24: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 12 iterative-reading

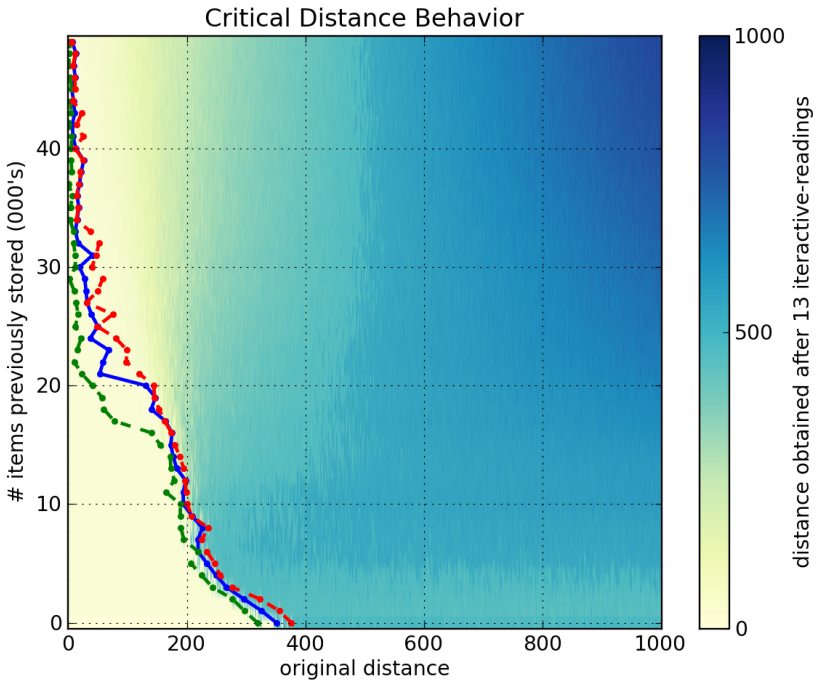

Figure 25: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 13 iterative-reading

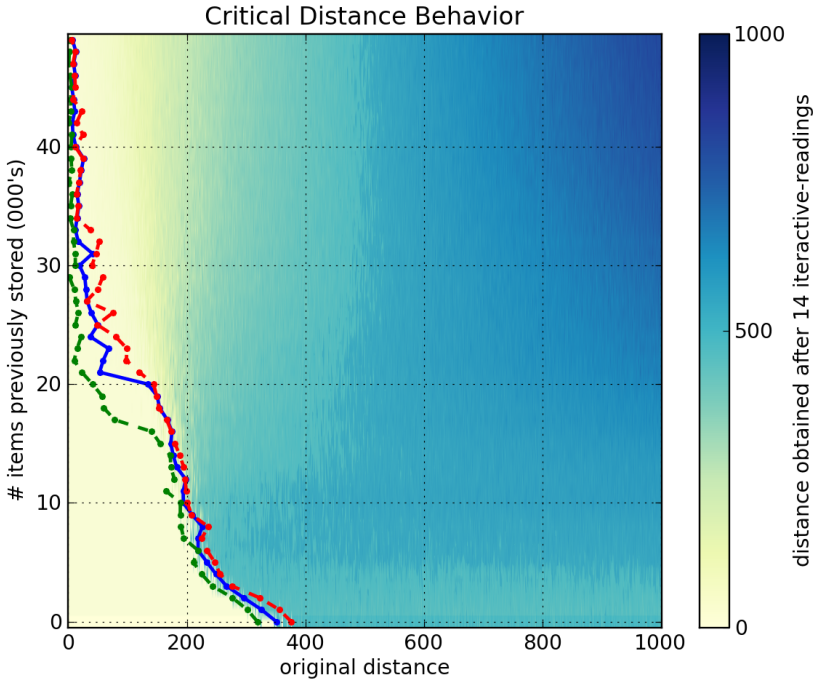

Figure 26: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 14 iterative-reading

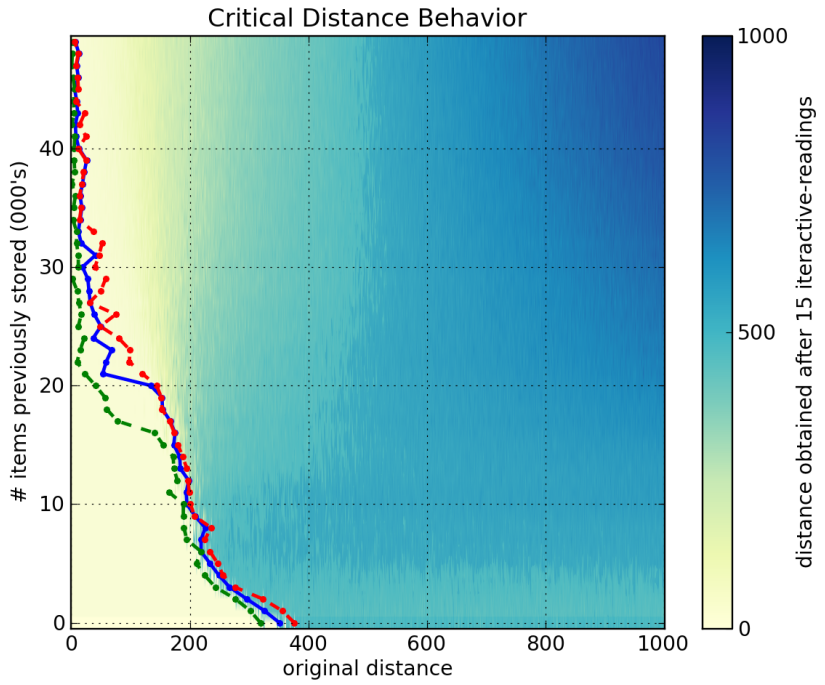

Figure 27: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 15 iterative-reading

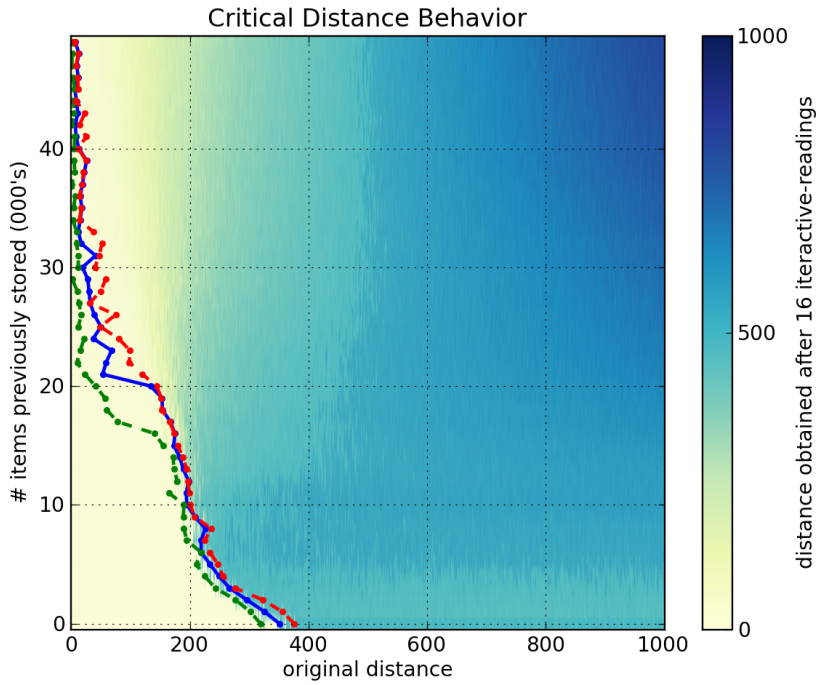

Figure 28: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 16 iterative-reading

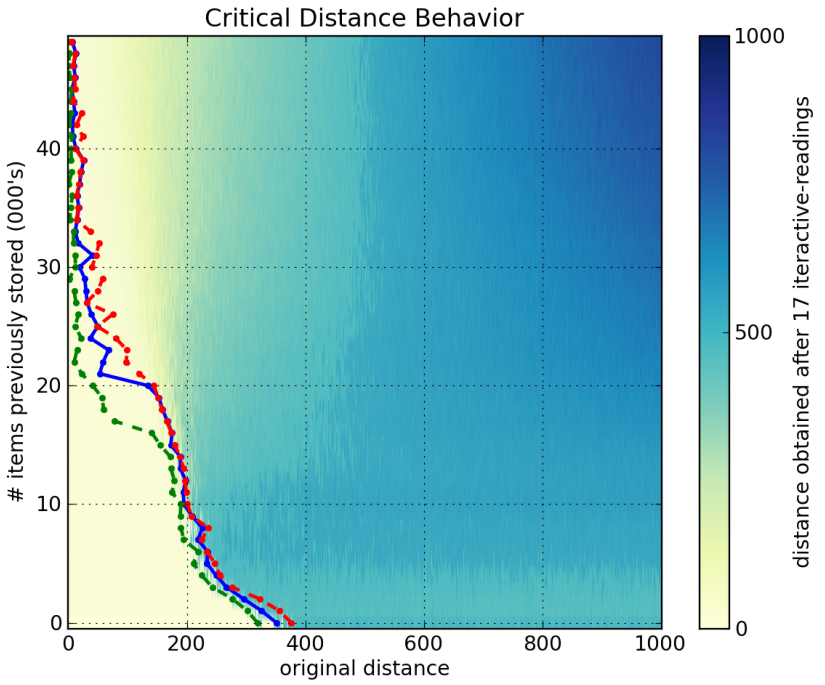

Figure 29: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 17 iterative-reading

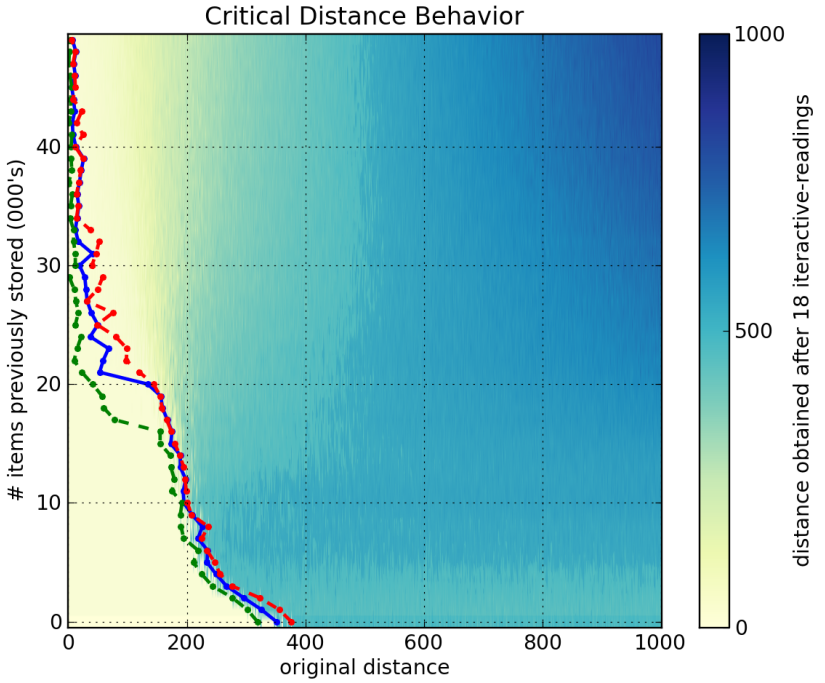

Figure 30: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 18 iterative-reading

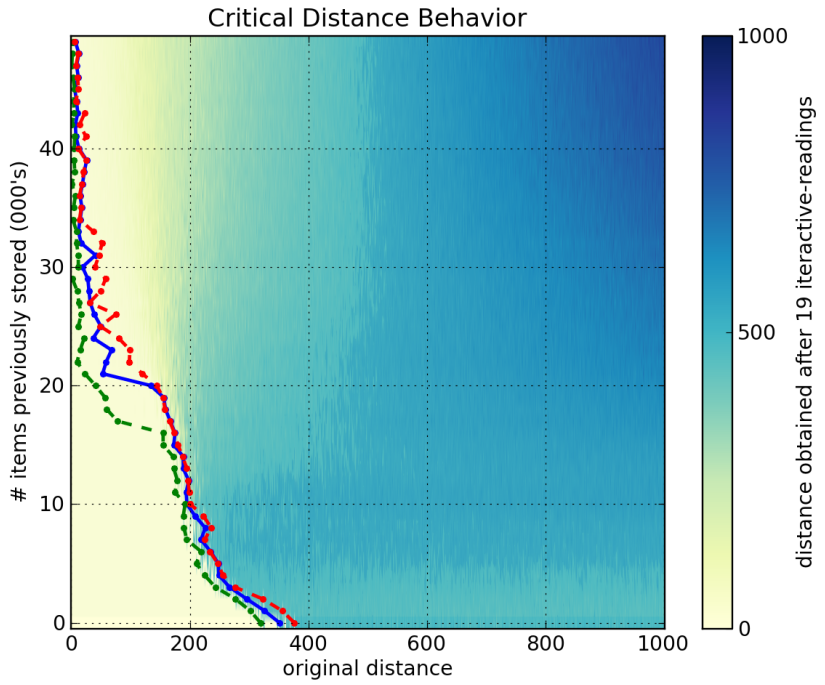

Figure 31: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 19 iterative-reading

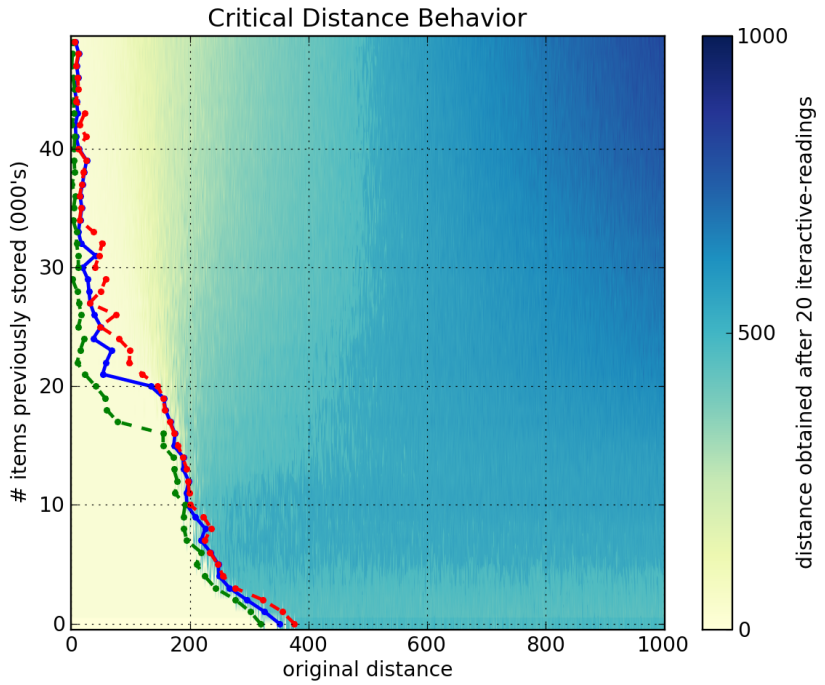

Figure 32: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 20 iterative-reading

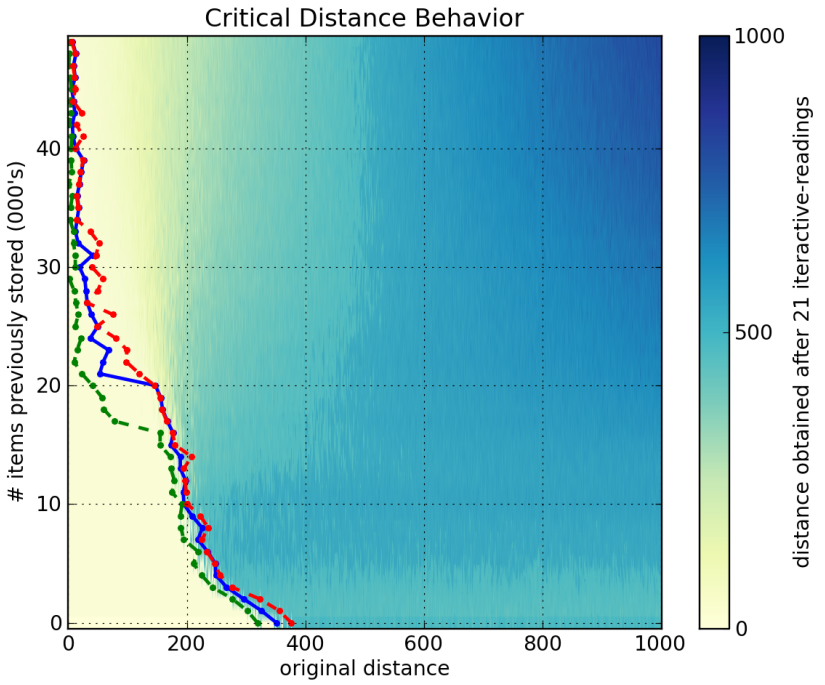

Figure 33: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 21 iterative-reading

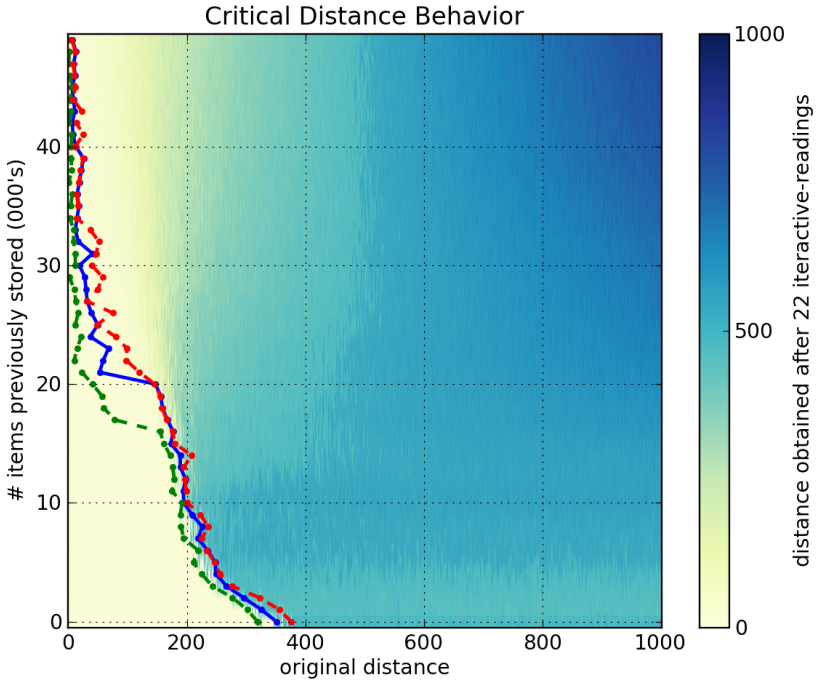

Figure 34: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 22 iterative-reading

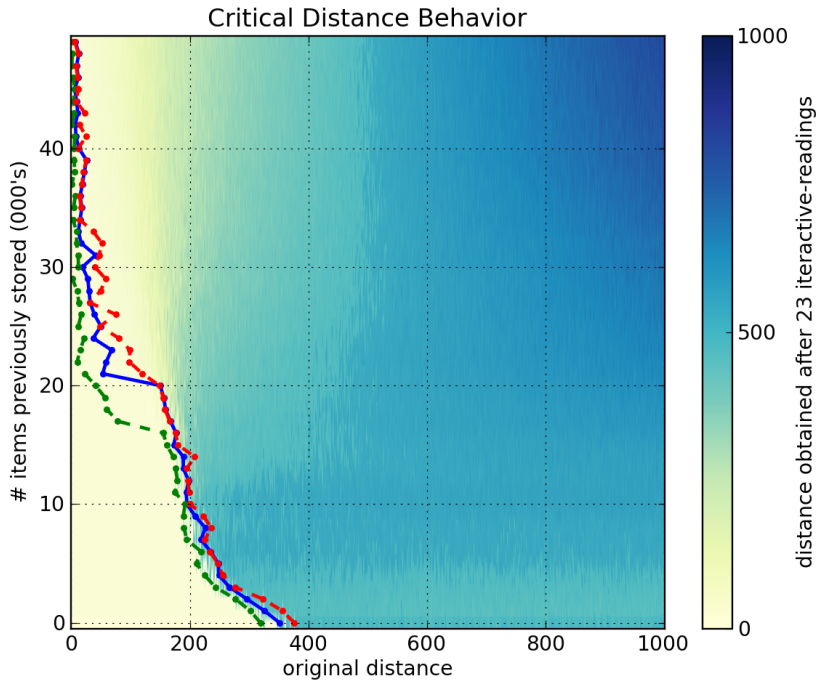

Figure 35: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 23 iterative-reading

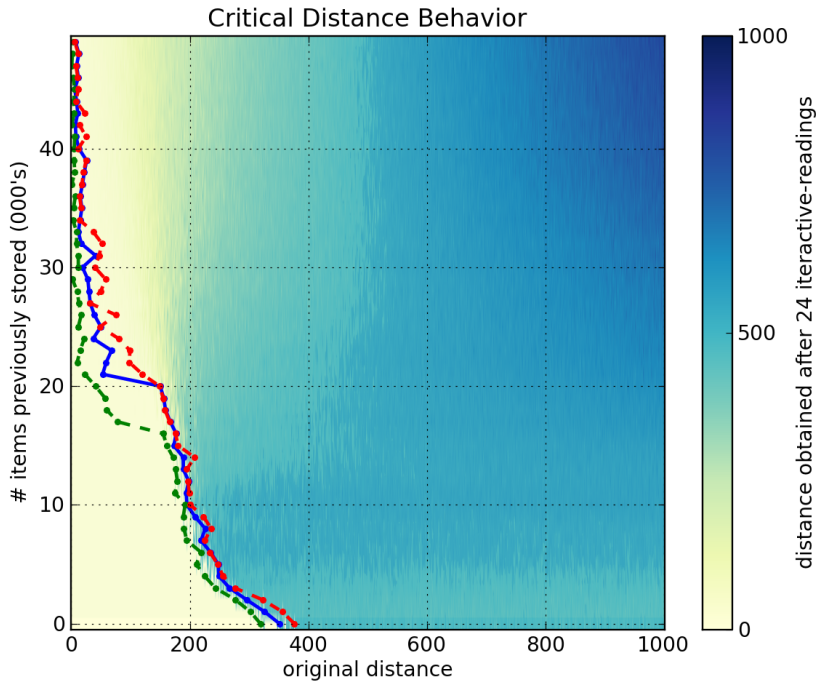

Figure 36: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 24 iterative-reading

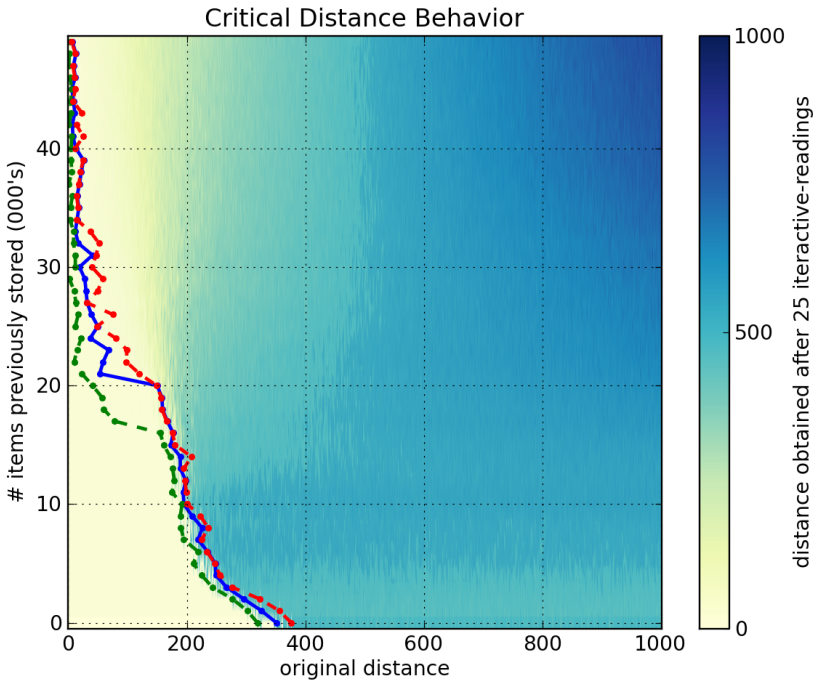

Figure 37: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 25 iterative-reading

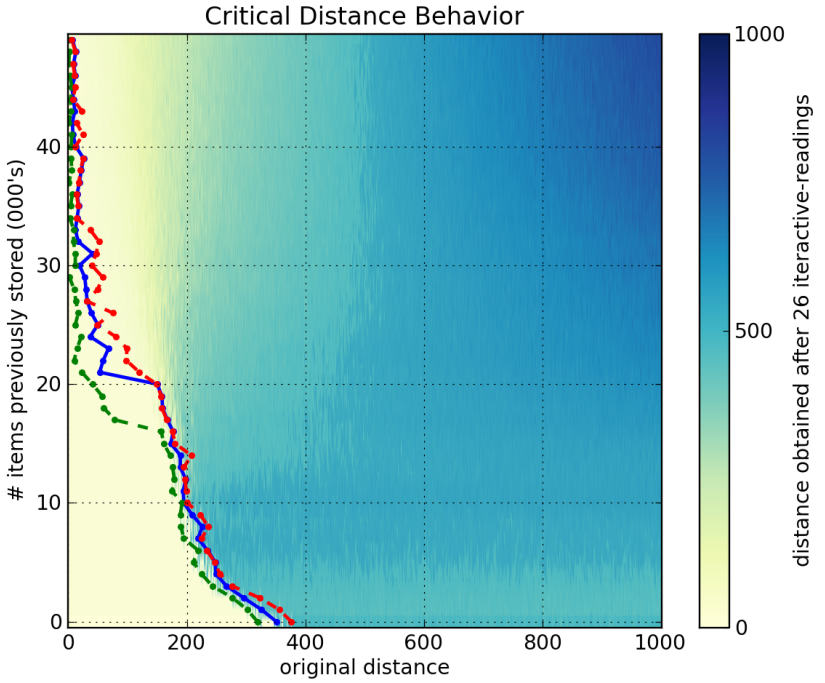

Figure 38: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 26 iterative-reading

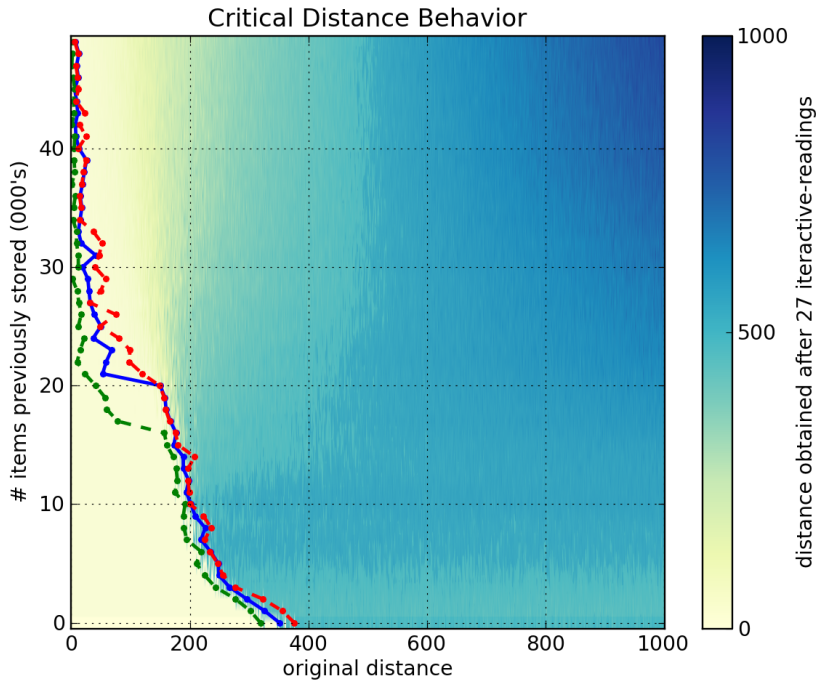

Figure 39: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 27 iterative-reading

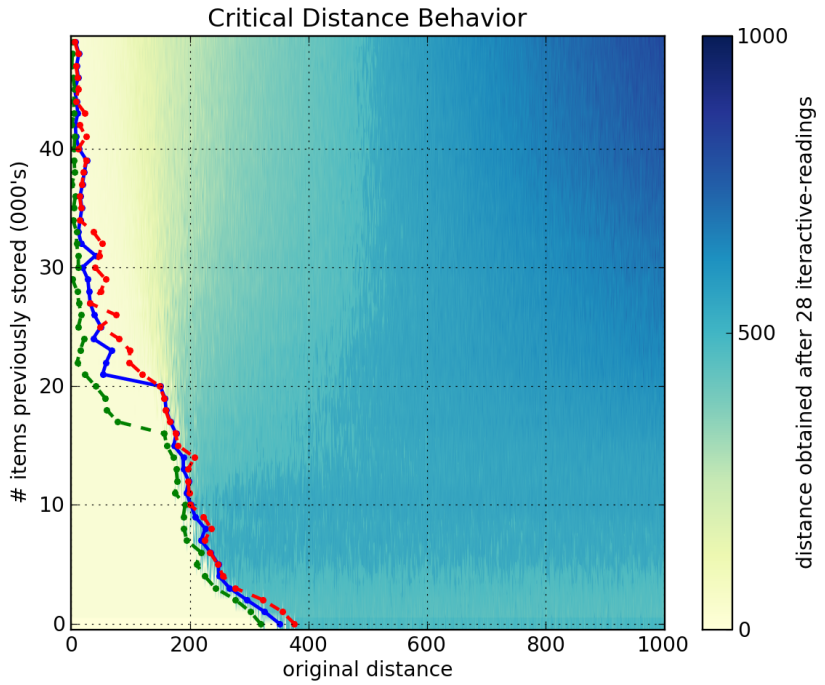

Figure 40: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 28 iterative-reading

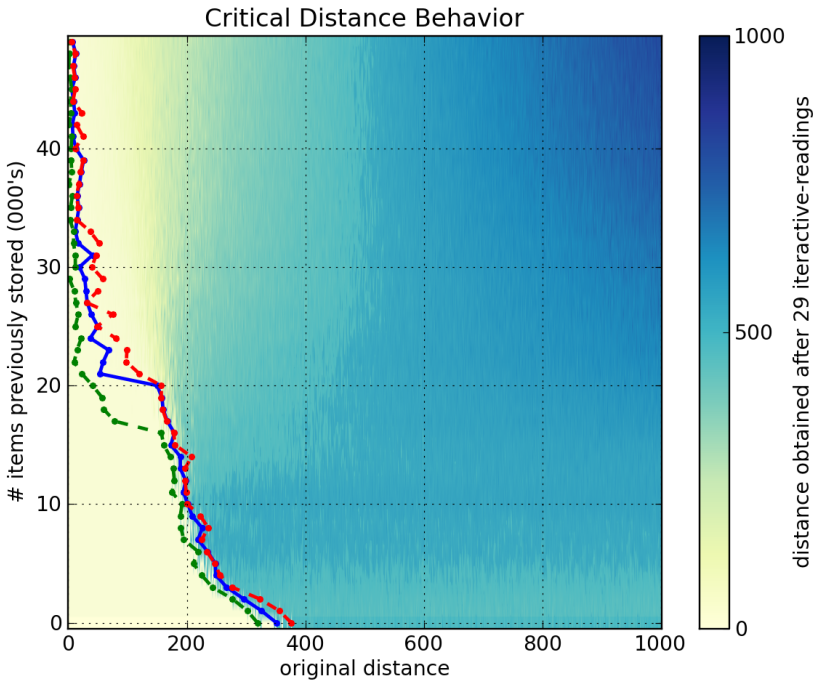

Figure 41: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 29 iterative-reading

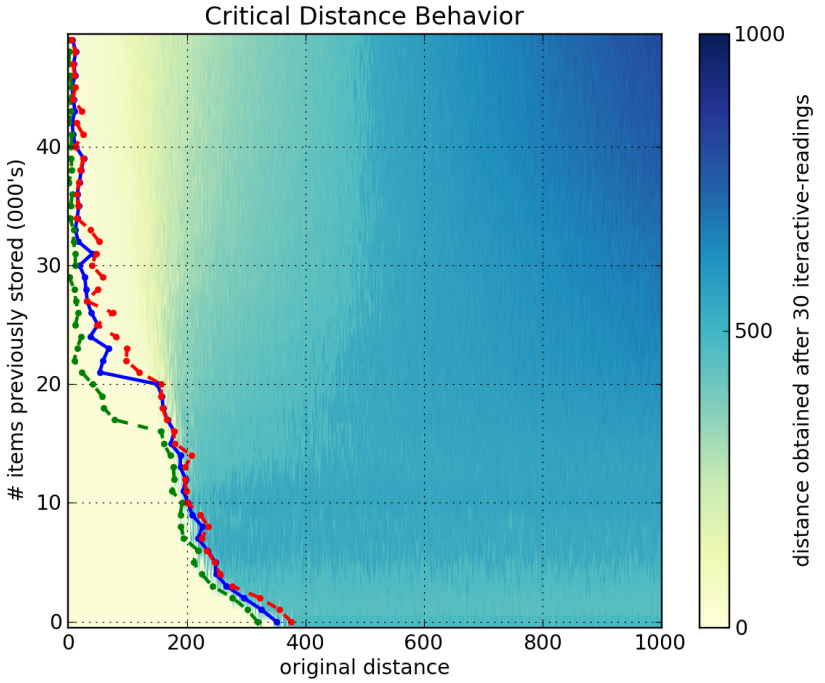

Figure 42: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 30 iterative-reading

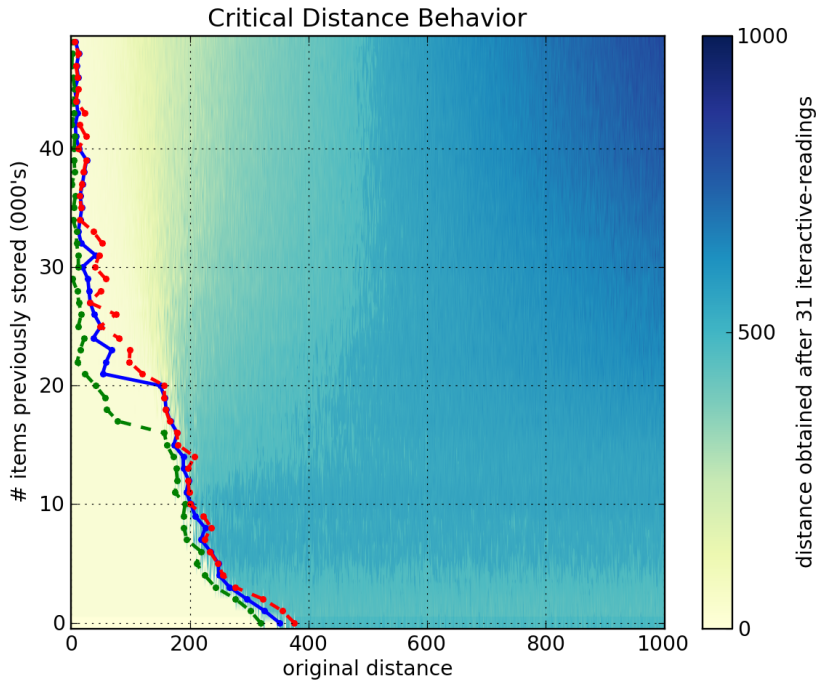

Figure 43: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 31 iterative-reading

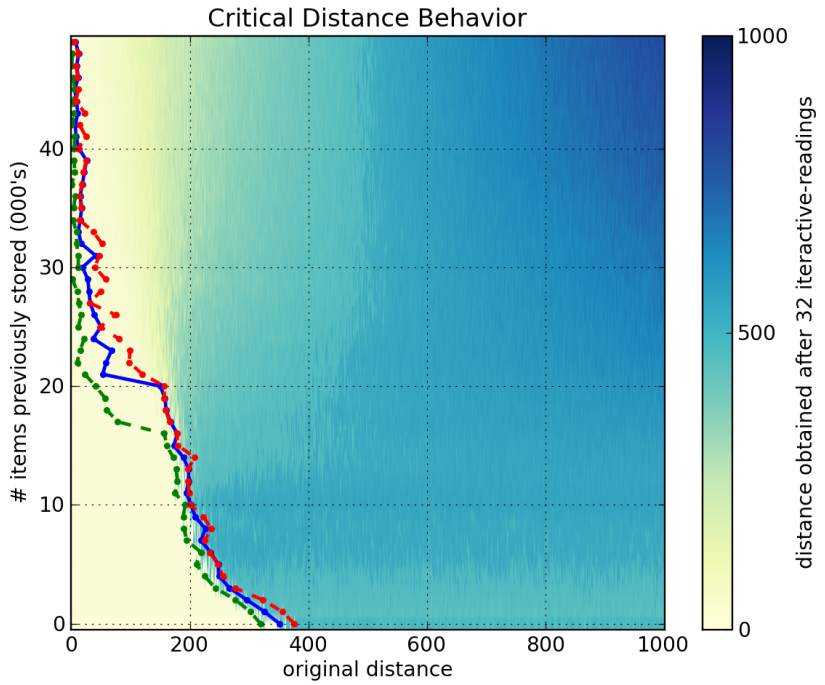

Figure 44: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 32 iterative-reading

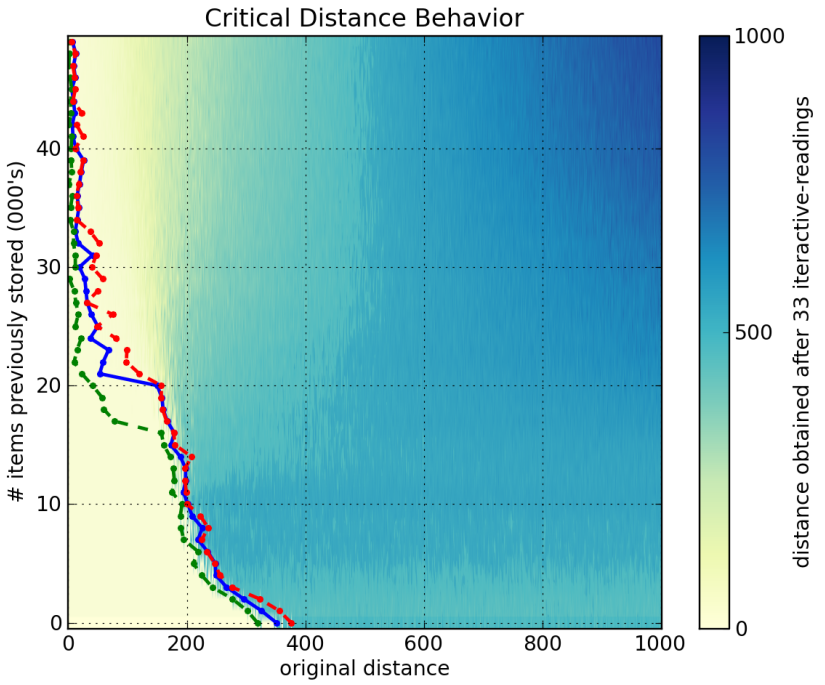

Figure 45: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 33 iterative-reading

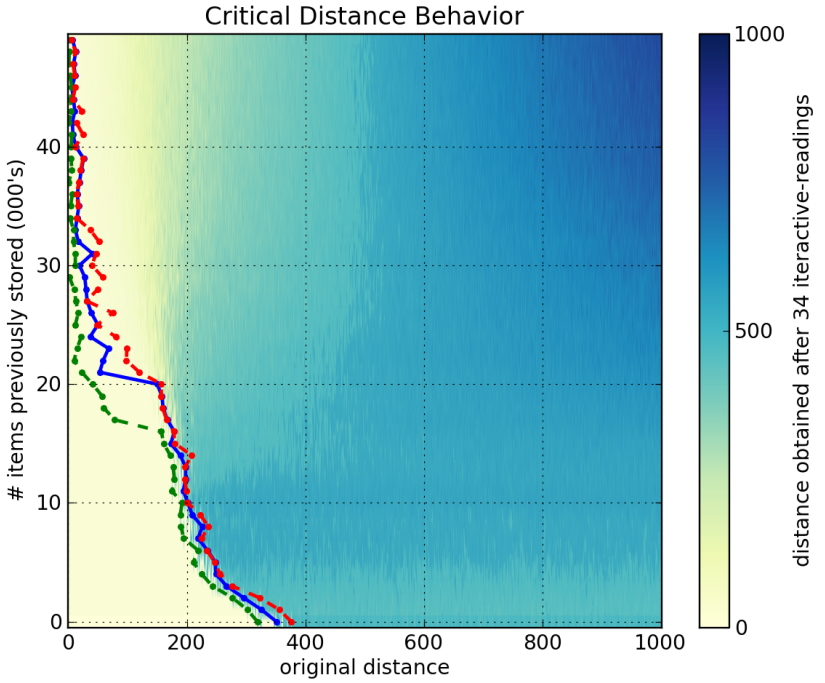

Figure 46: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 34 iterative-reading

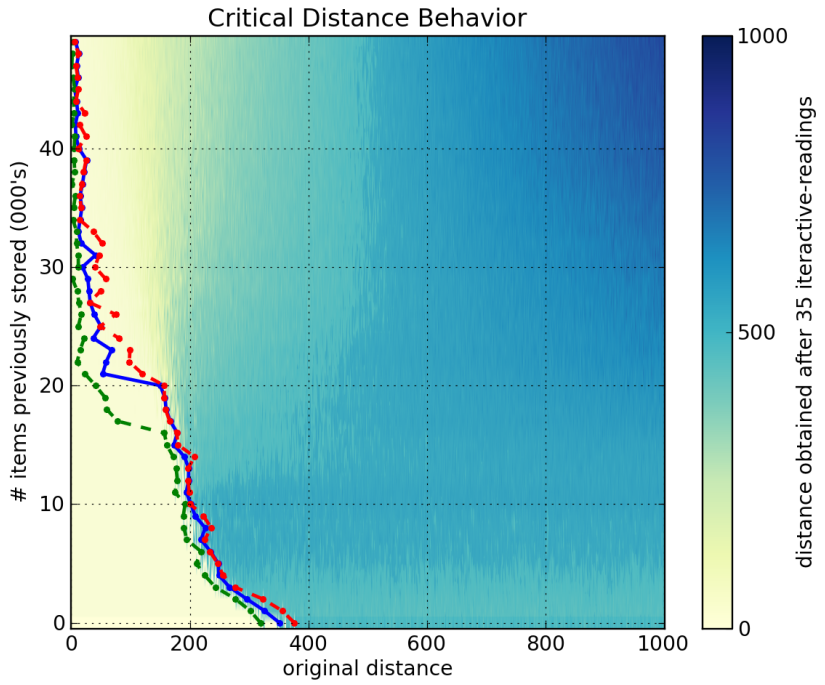

Figure 47: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 35 iterative-reading

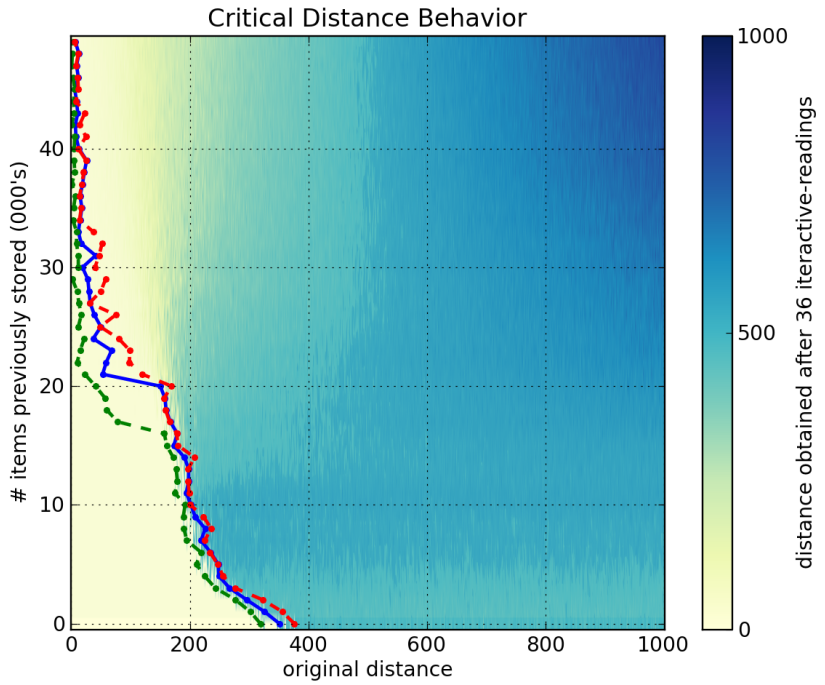

Figure 48: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 36 iterative-reading

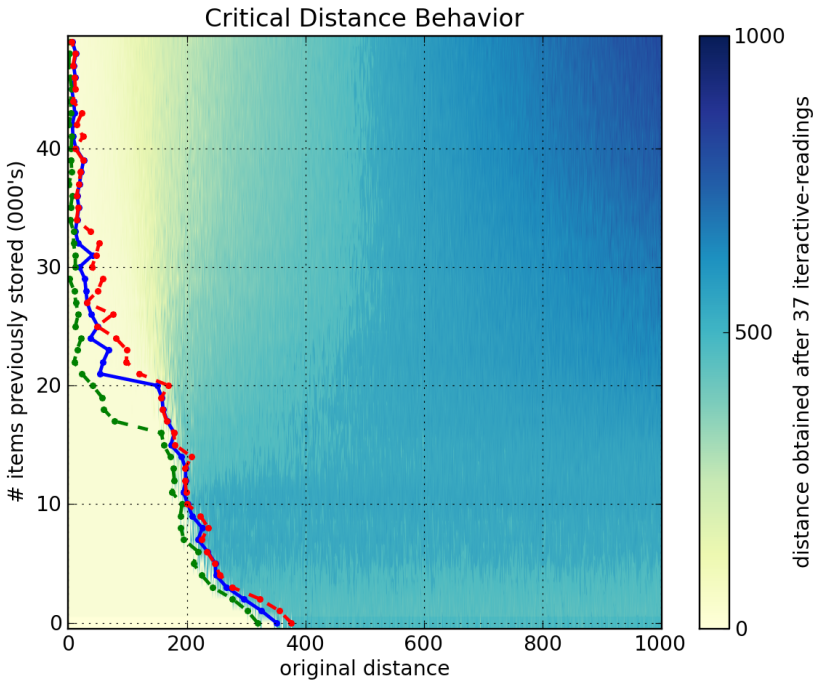

Figure 49: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 37 iterative-reading

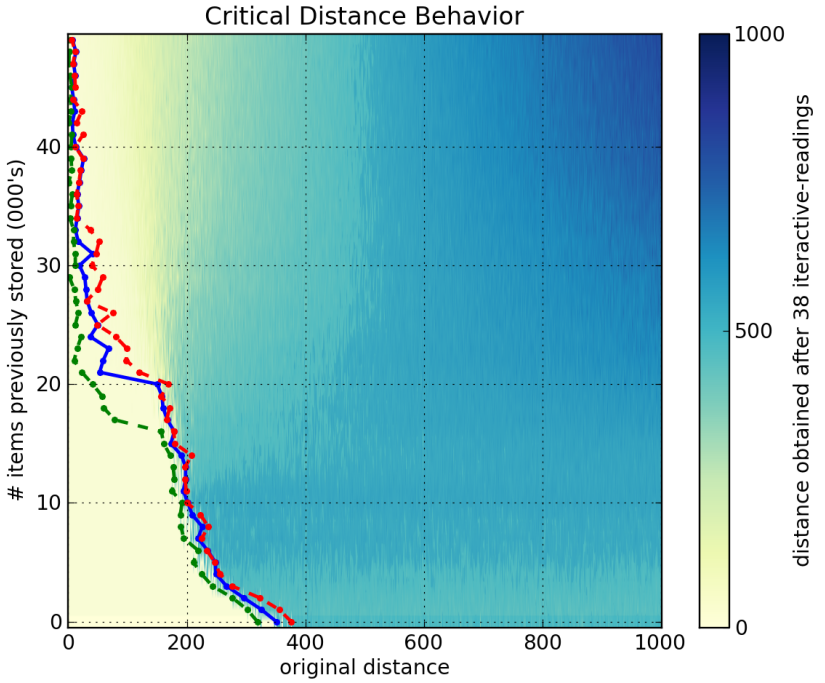

Figure 50: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 38 iterative-reading

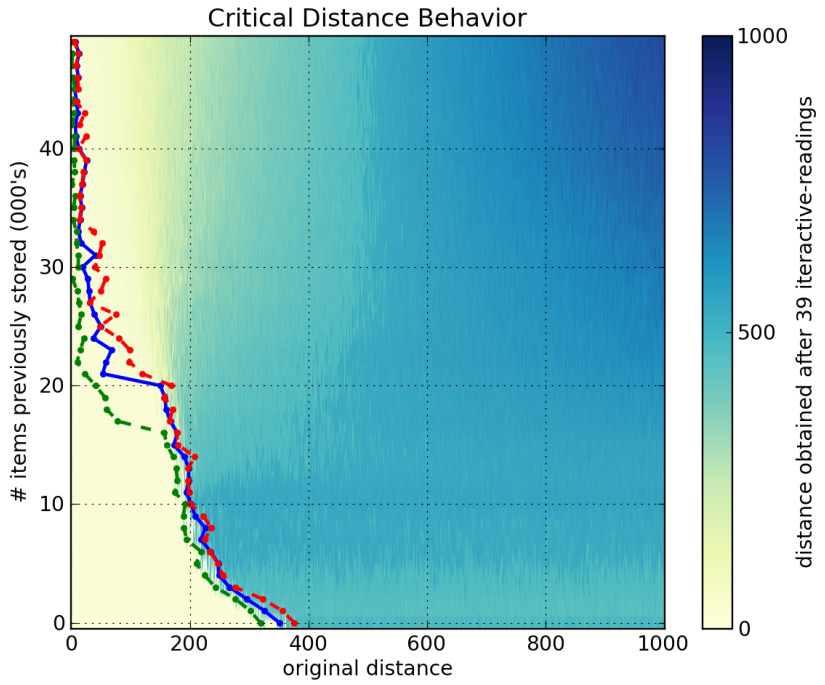

Figure 51: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 39 iterative-reading

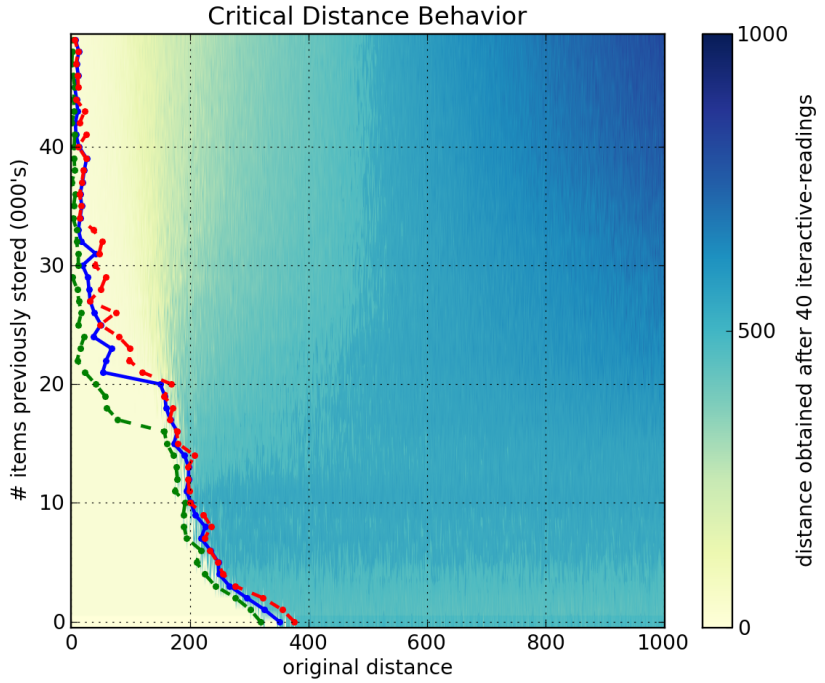

Figure 52: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 40 iterative-reading

C

APPENDIX C: ADDITIONAL ITERATIVE READING RESULTS FOR 256-DIMENSIONAL MEMORY

For the sake of completeness, in this appendix we include the entire set of figures generated with 256 dimensions, a single write of target bitstring, ranging from 1 to 40 iterative-readings.

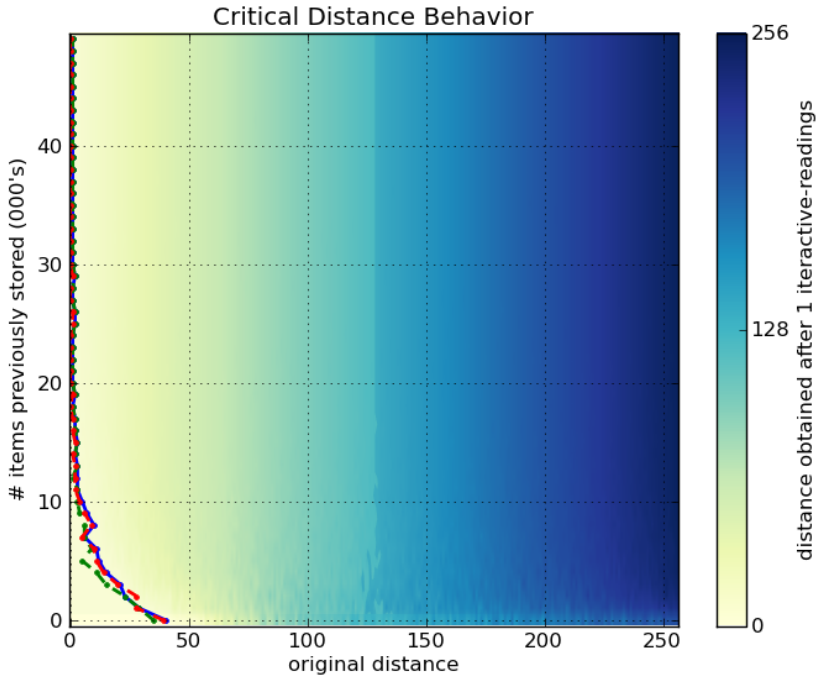

Figure 53: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 1 iterative-reading

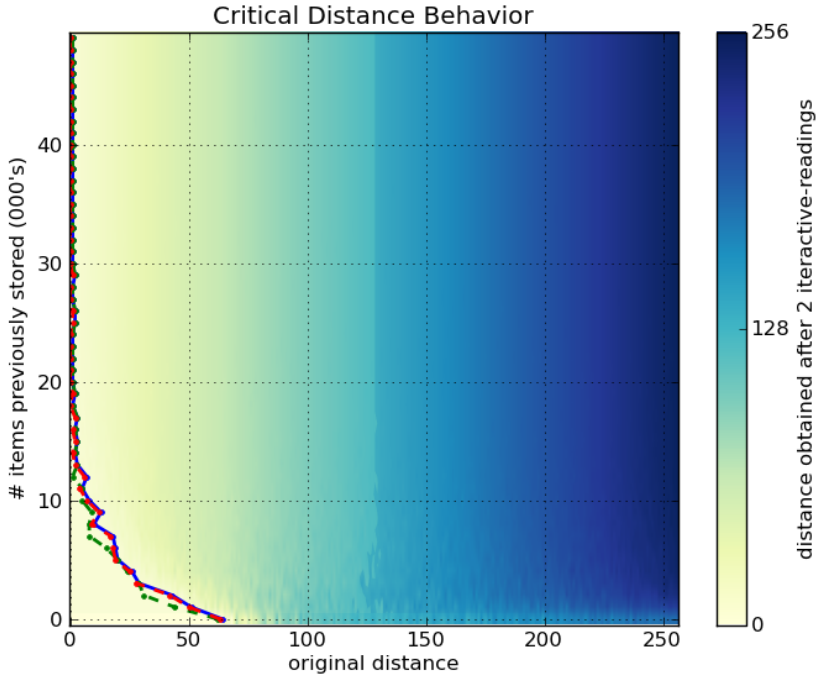

Figure 54: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 2 iterative-reading

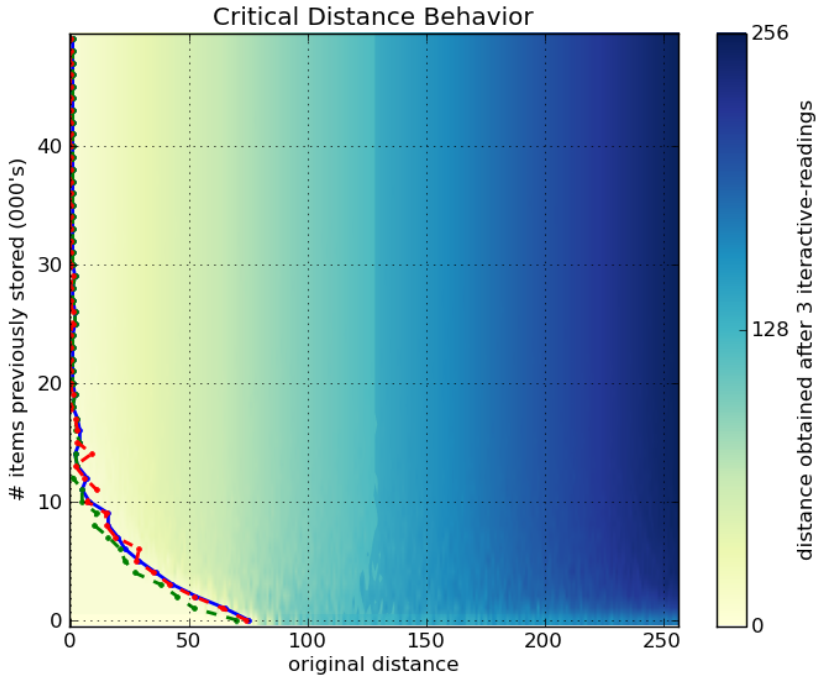

Figure 55: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 3 iterative-reading

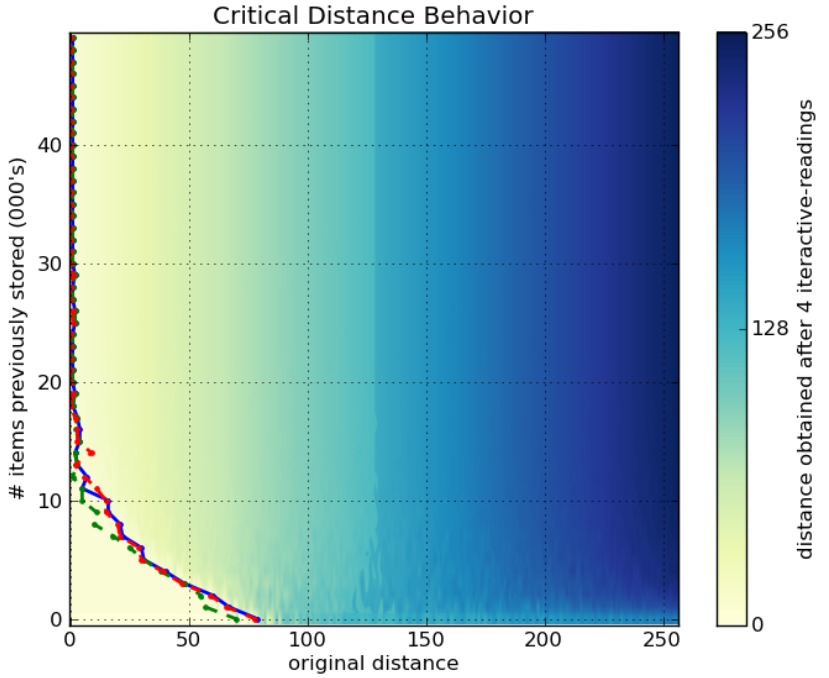

Figure 56: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 4 iterative-reading

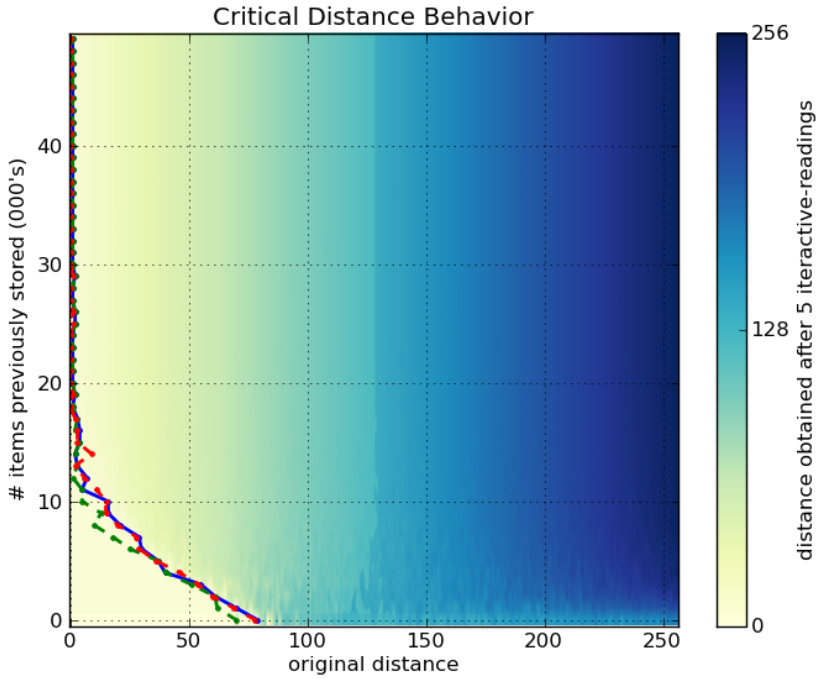

Figure 57: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 5 iterative-reading

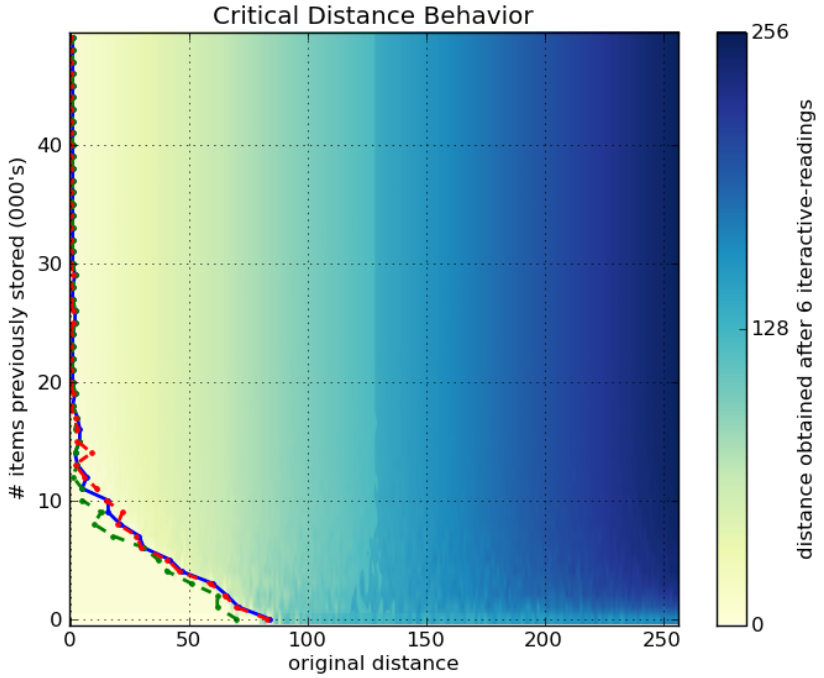

Figure 58: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 6 iterative-reading

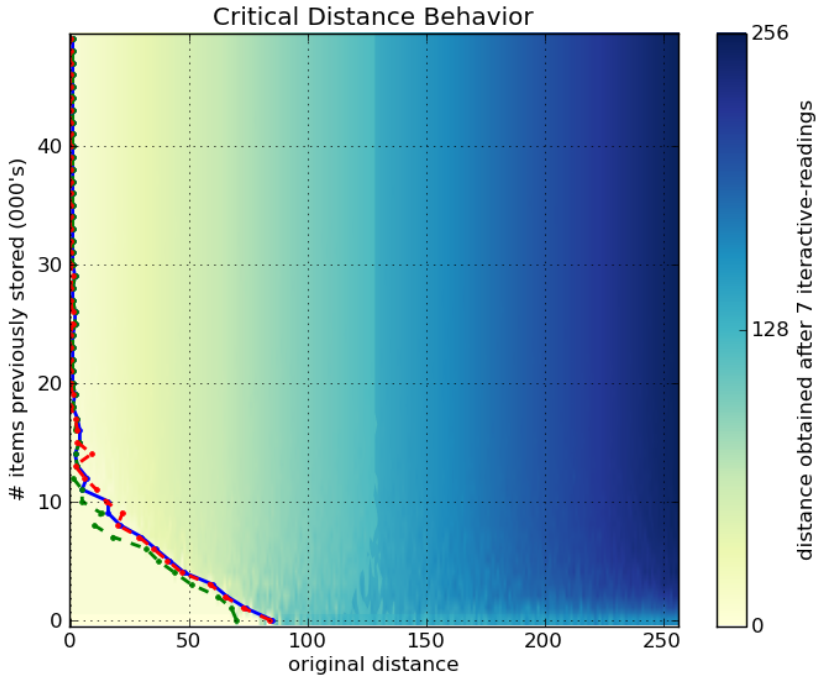

Figure 59: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 7 iterative-reading

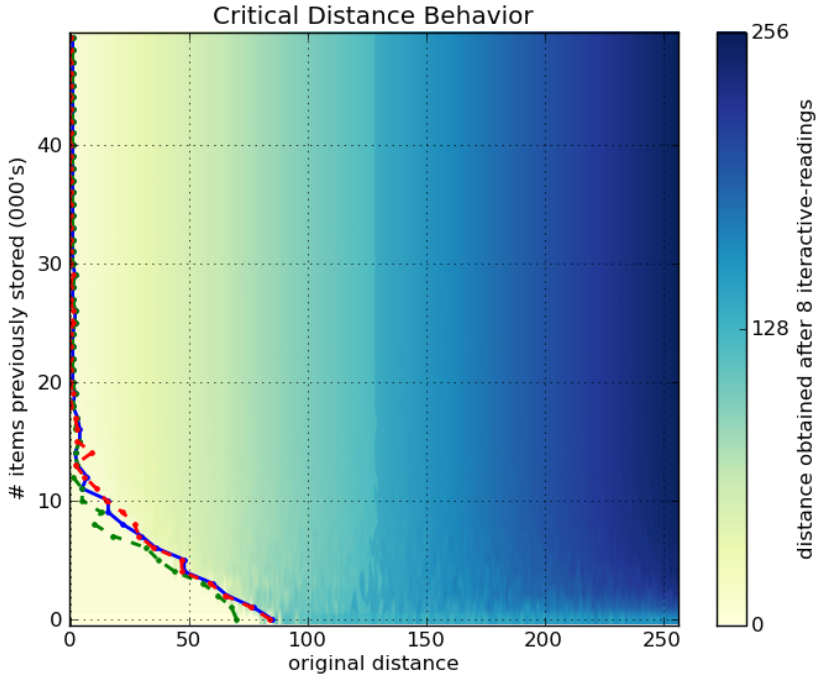

Figure 60: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 8 iterative-reading

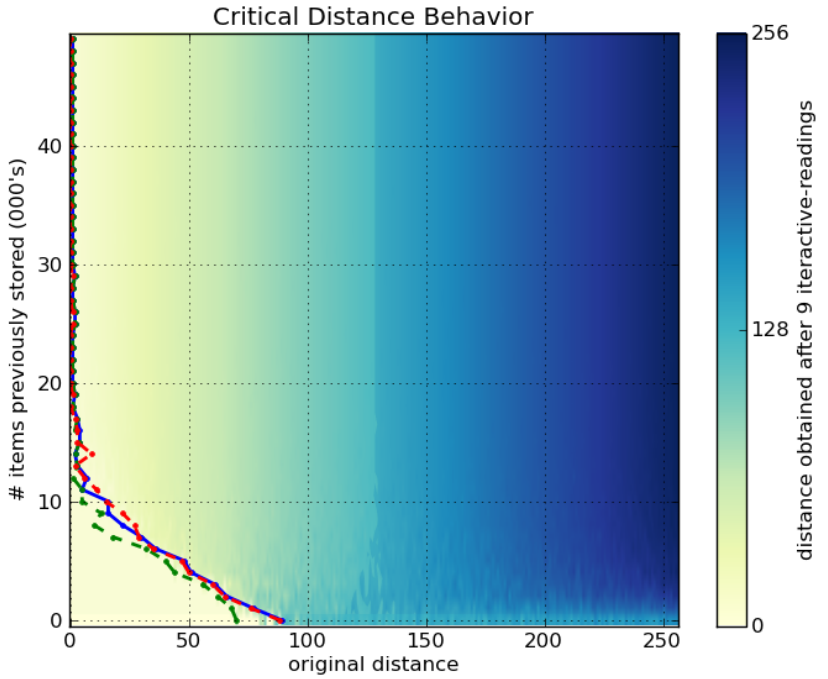

Figure 61: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 9 iterative-reading

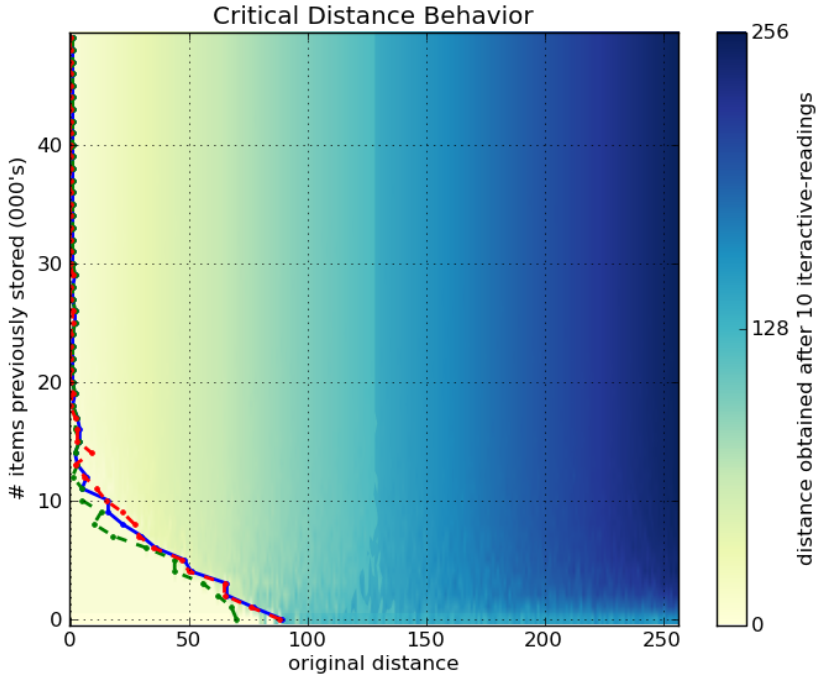

Figure 62: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 10 iterative-reading

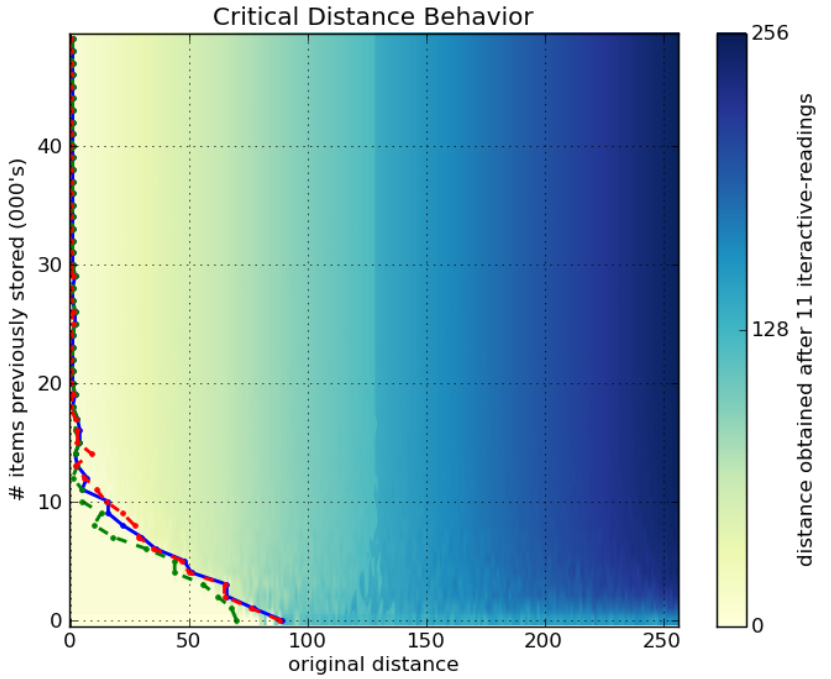

Figure 63: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 11 iterative-reading

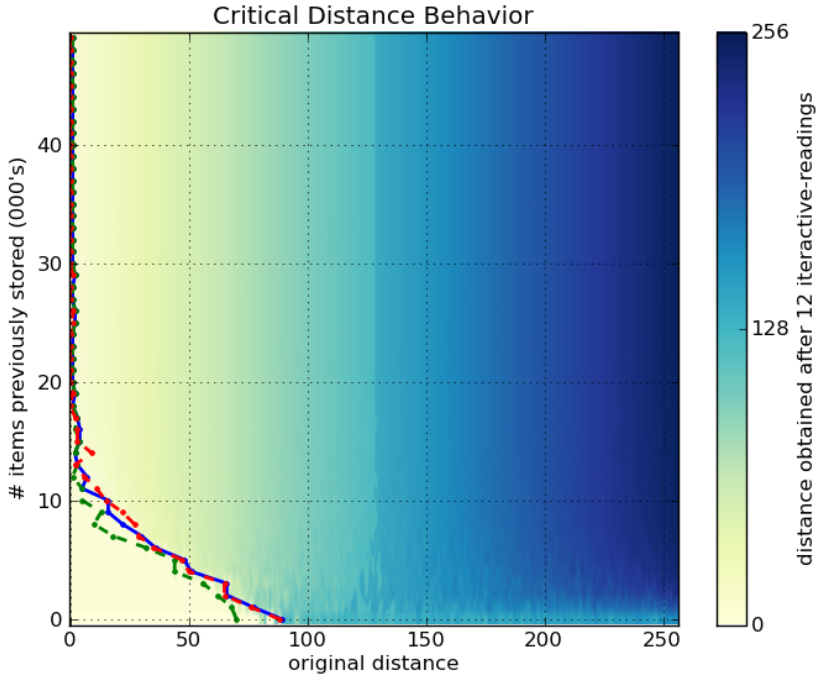

Figure 64: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 12 iterative-reading

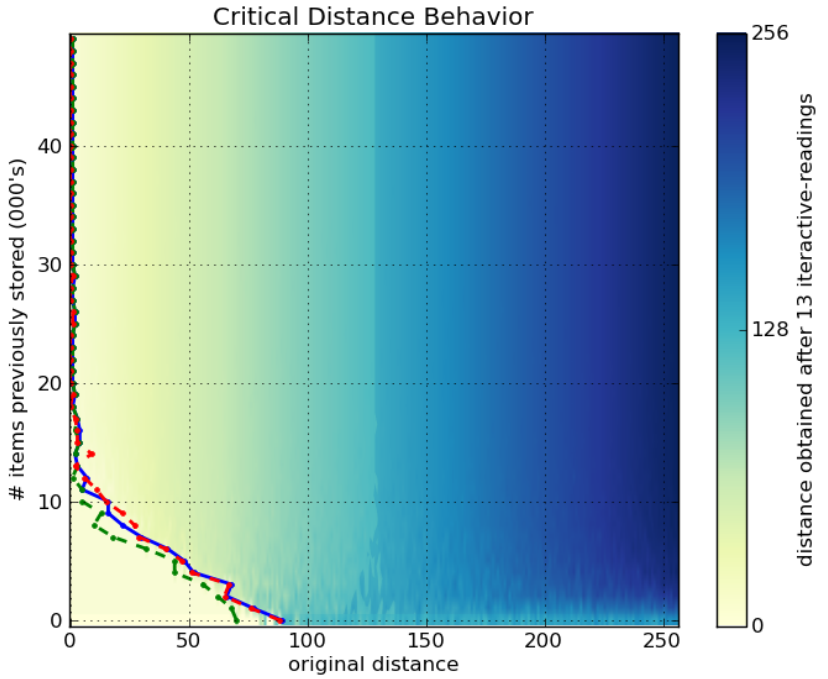

Figure 65: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 13 iterative-reading

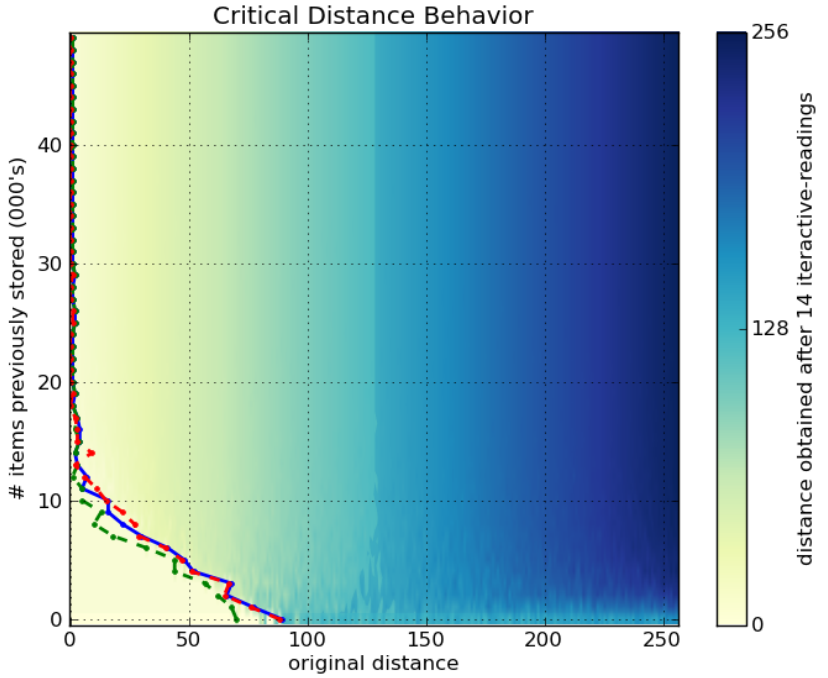

Figure 66: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 14 iterative-reading

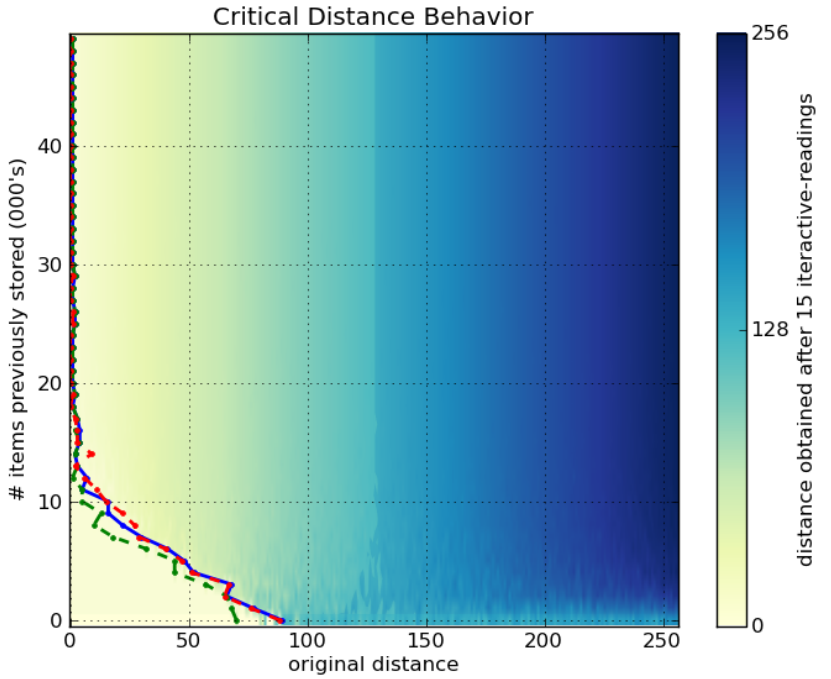

Figure 67: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 15 iterative-reading

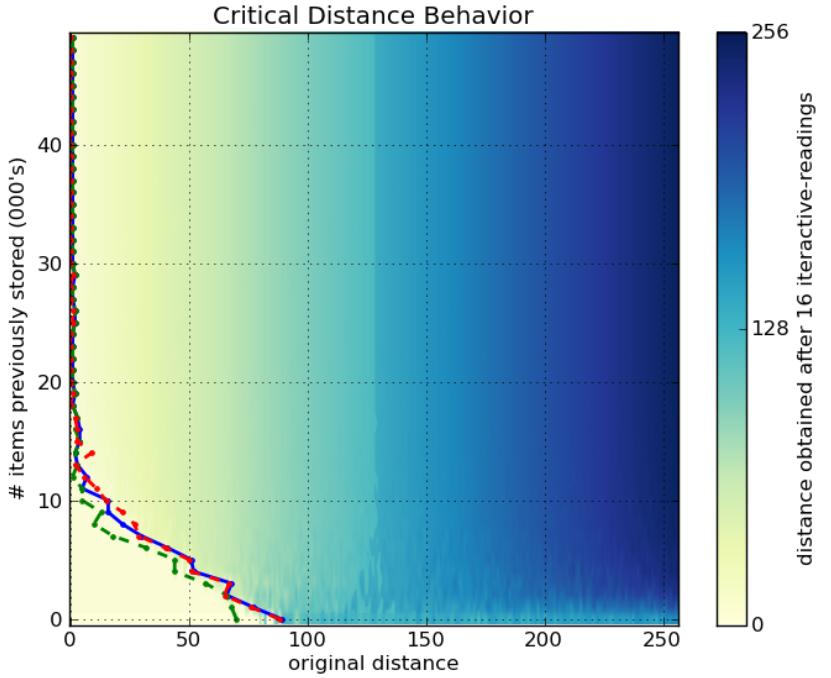

Figure 68: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 16 iterative-reading

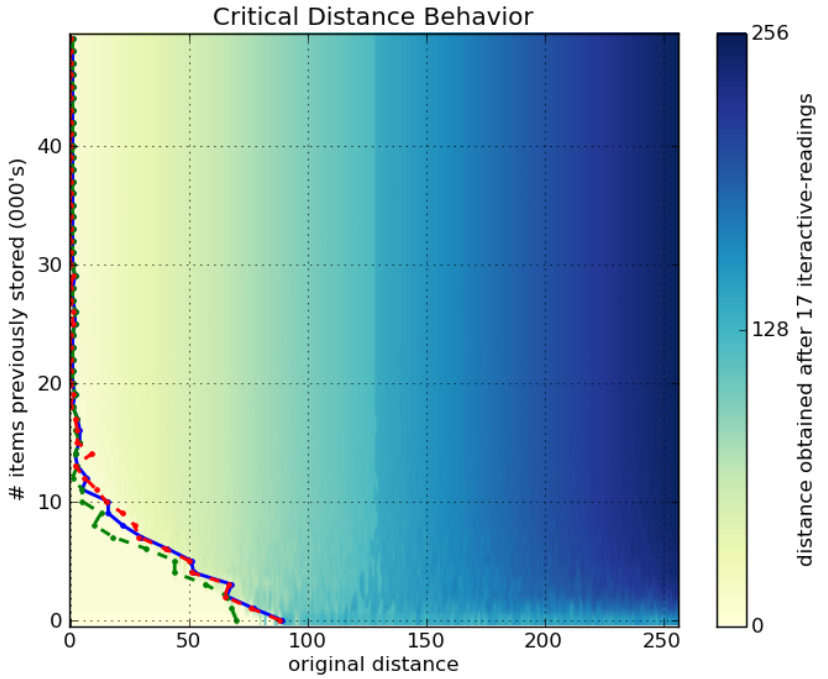

Figure 69: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 17 iterative-reading

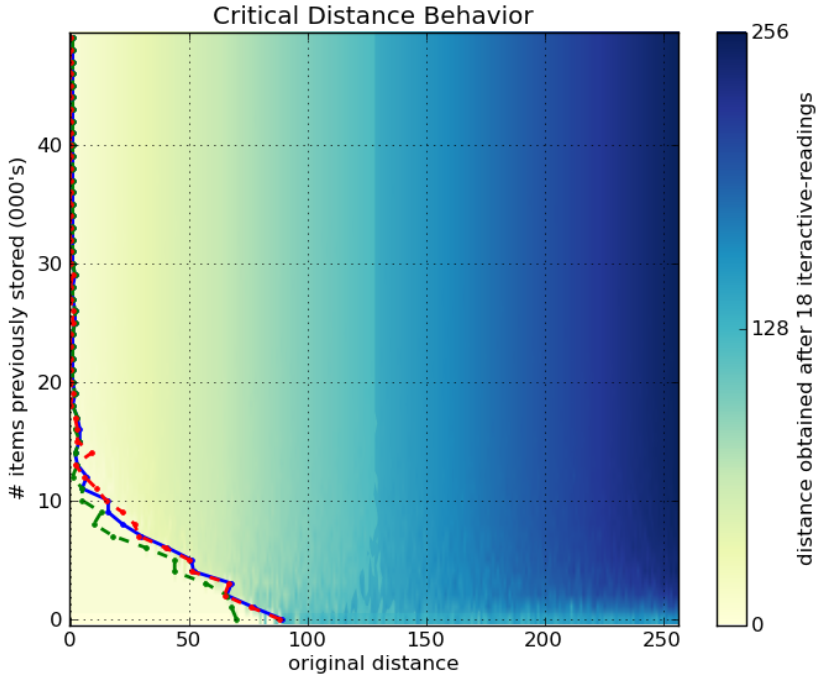

Figure 70: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 18 iterative-reading

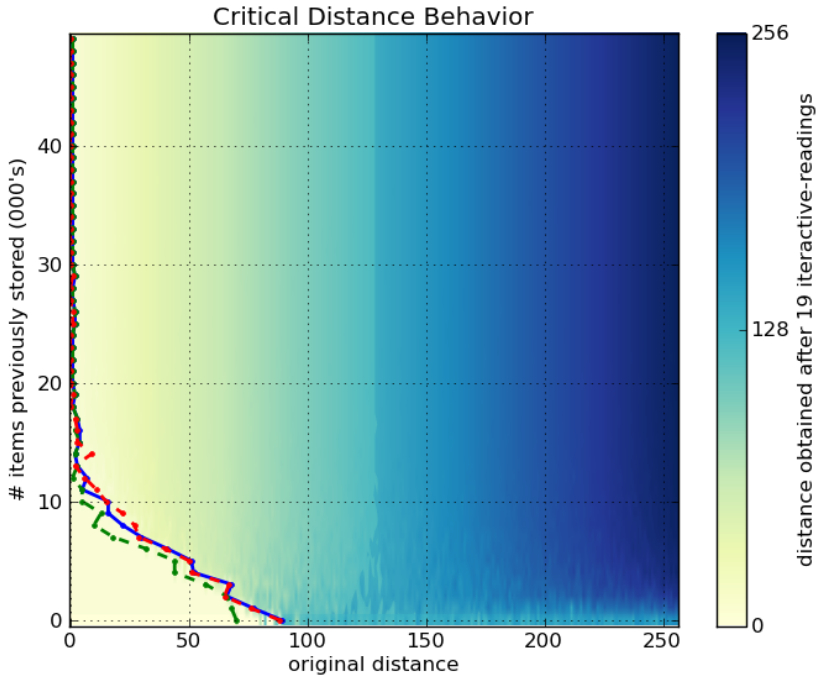

Figure 71: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 19 iterative-reading

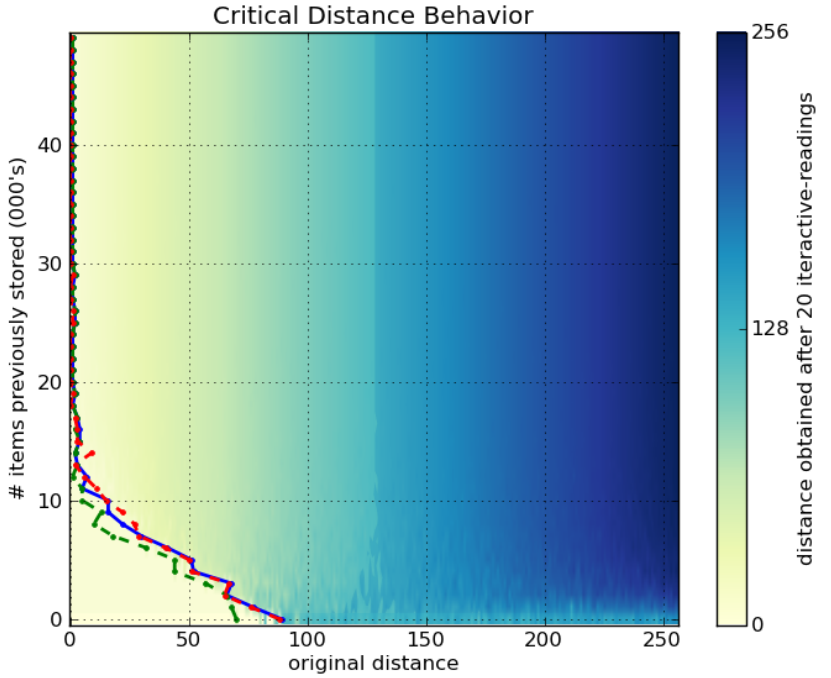

Figure 72: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 20 iterative-reading

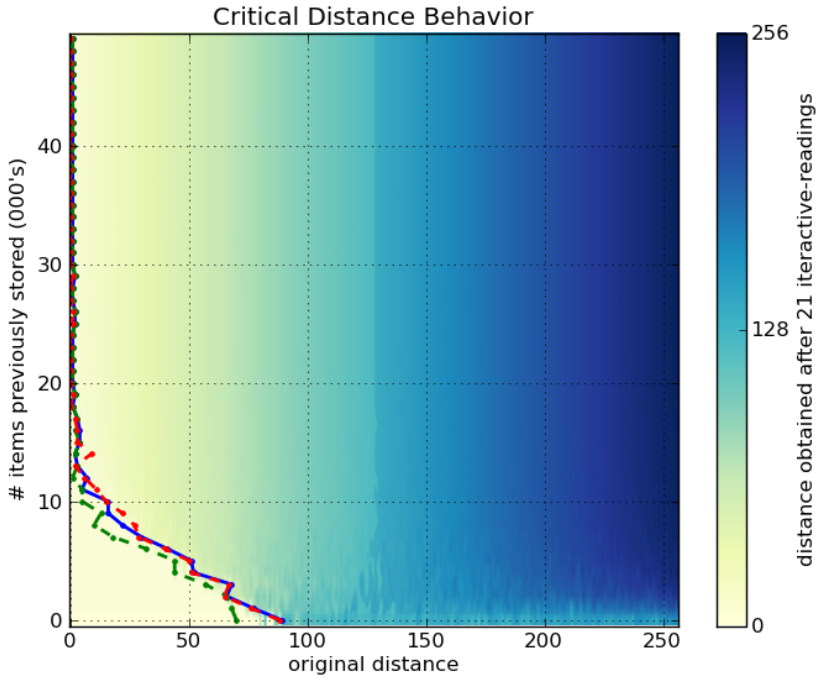

Figure 73: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 21 iterative-reading

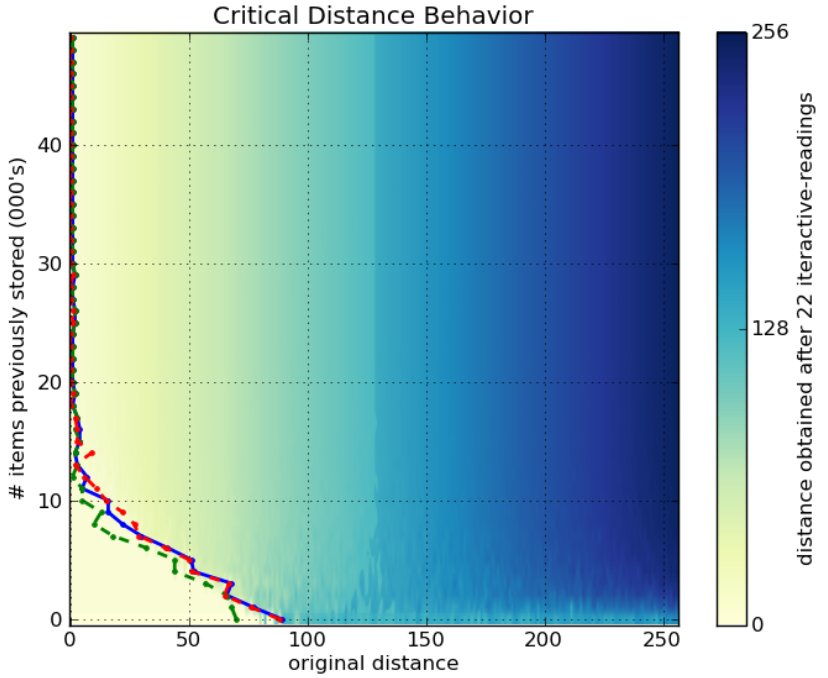

Figure 74: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 22 iterative-reading

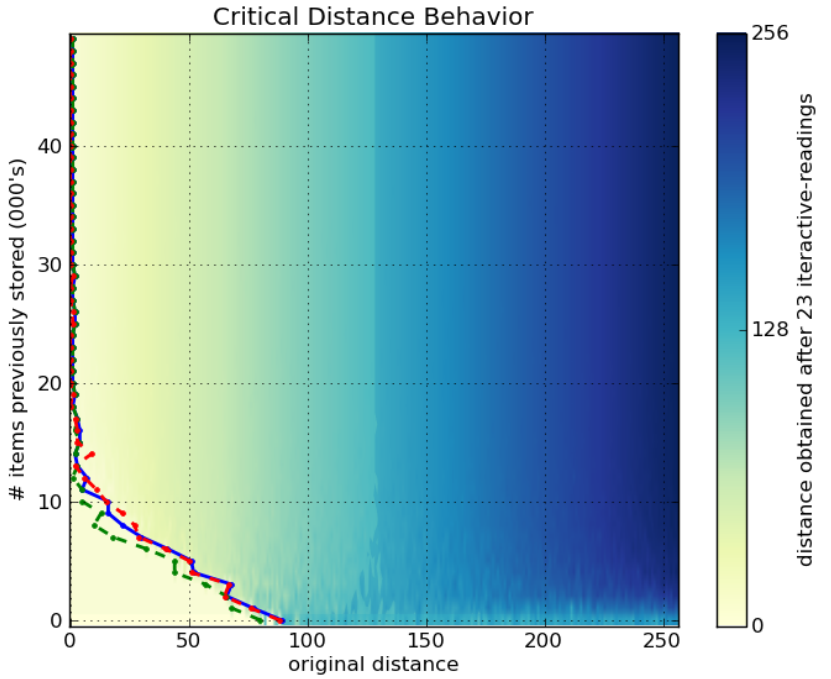

Figure 75: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 23 iterative-reading

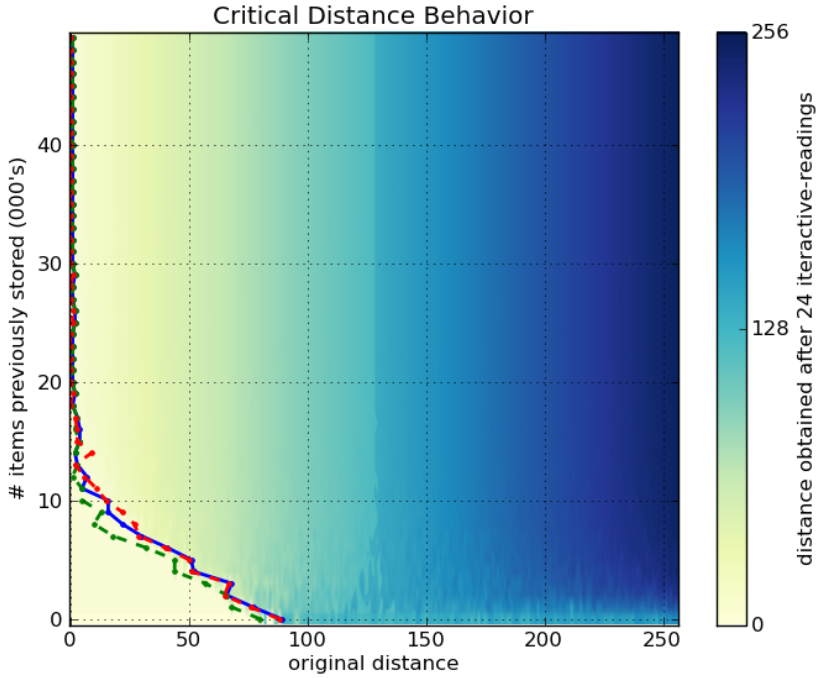

Figure 76: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 24 iterative-reading

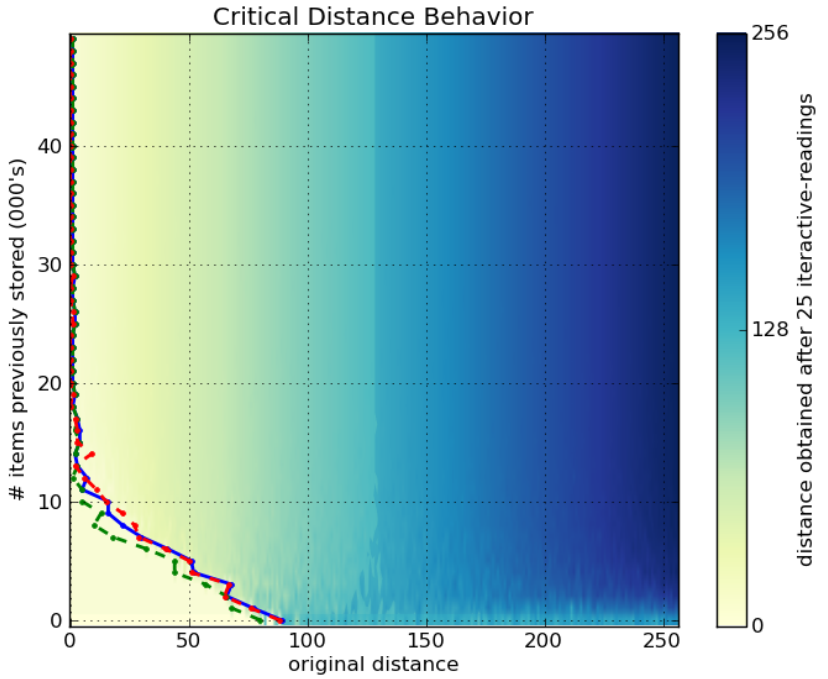

Figure 77: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 25 iterative-reading

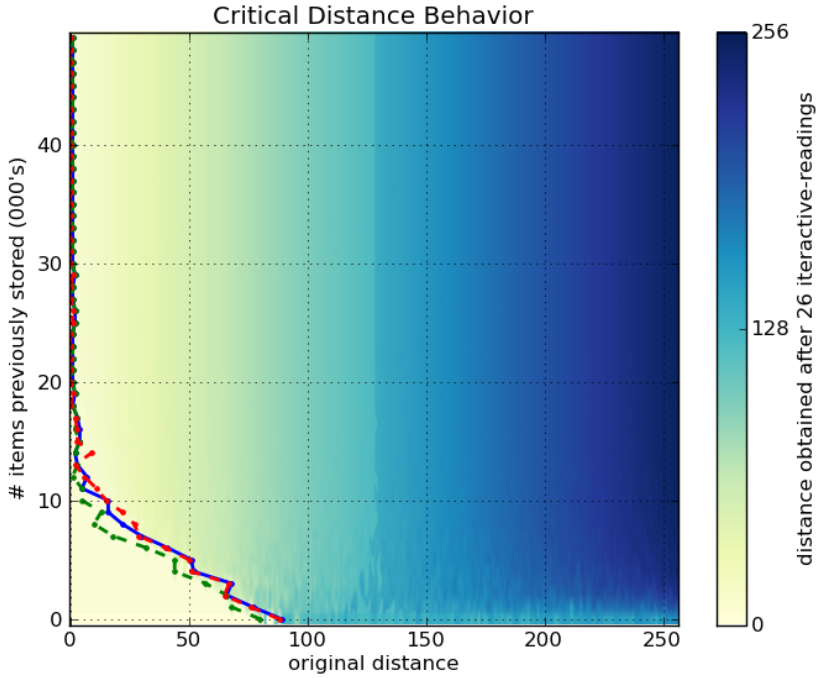

Figure 78: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 26 iterative-reading

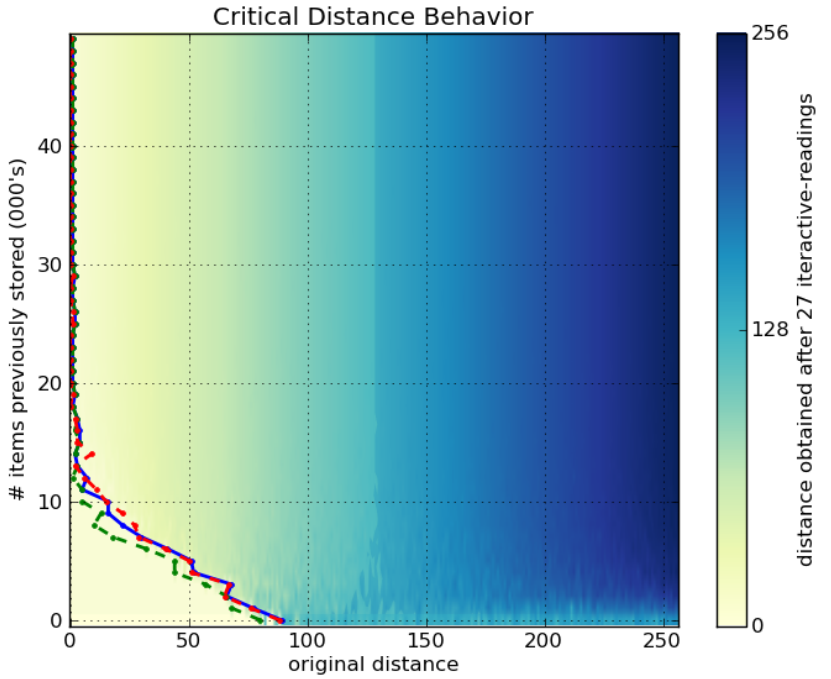

Figure 79: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 27 iterative-reading

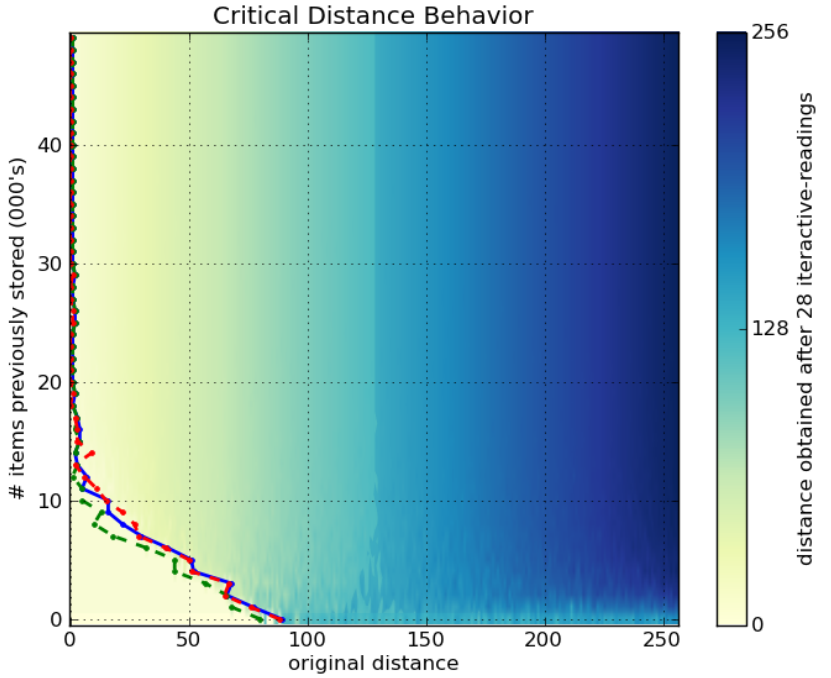

Figure 80: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 28 iterative-reading

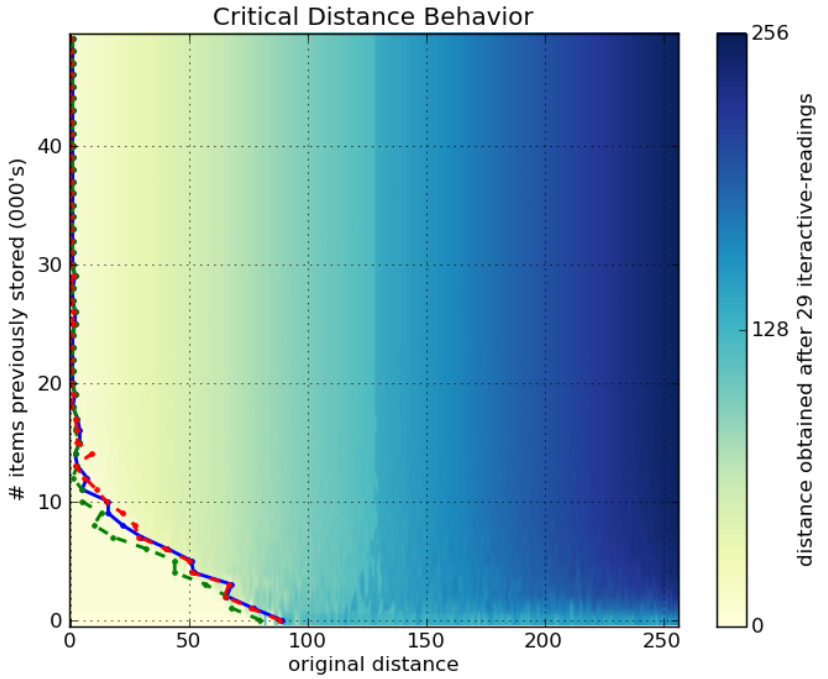

Figure 81: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 29 iterative-reading

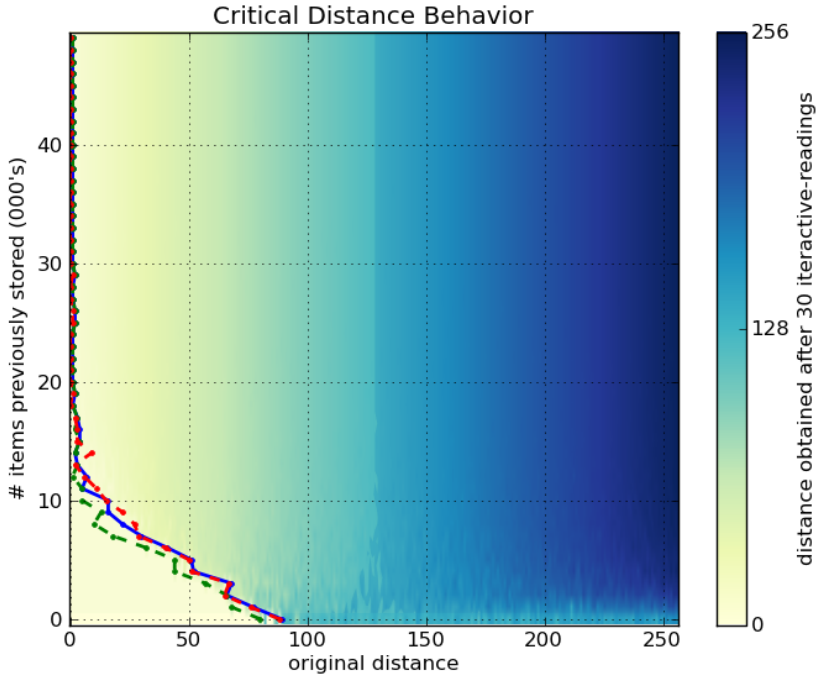

Figure 82: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 30 iterative-reading

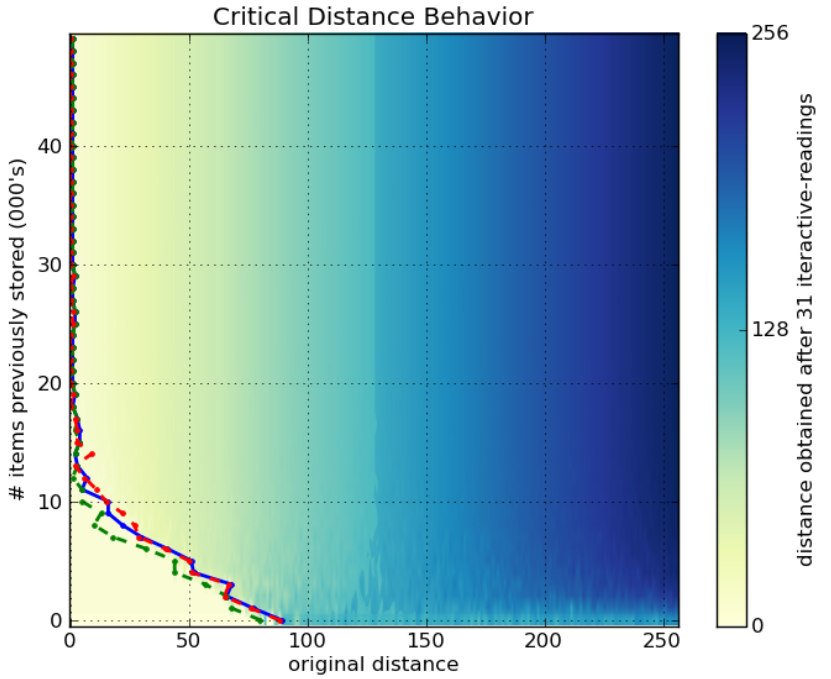

Figure 83: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 31 iterative-reading

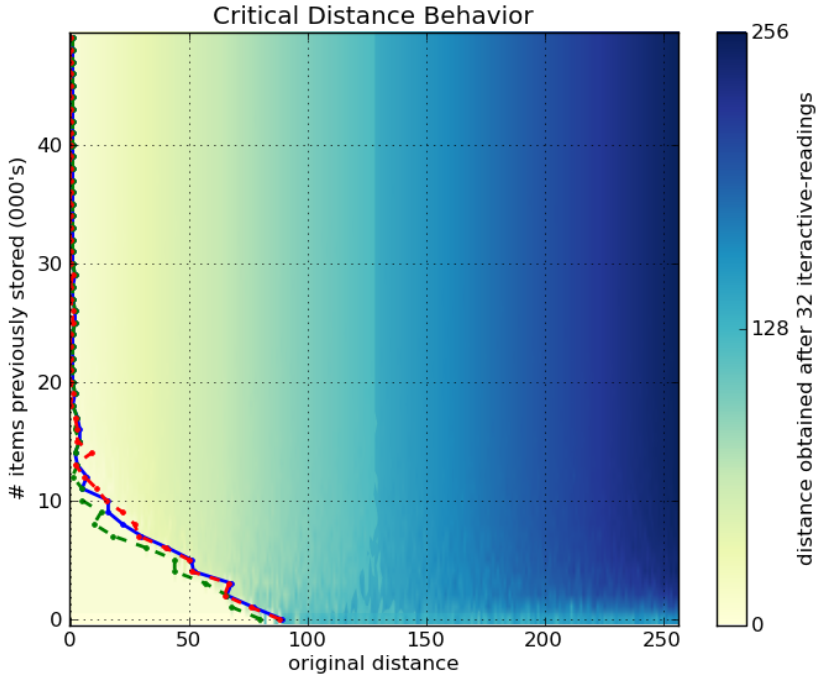

Figure 84: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 32 iterative-reading

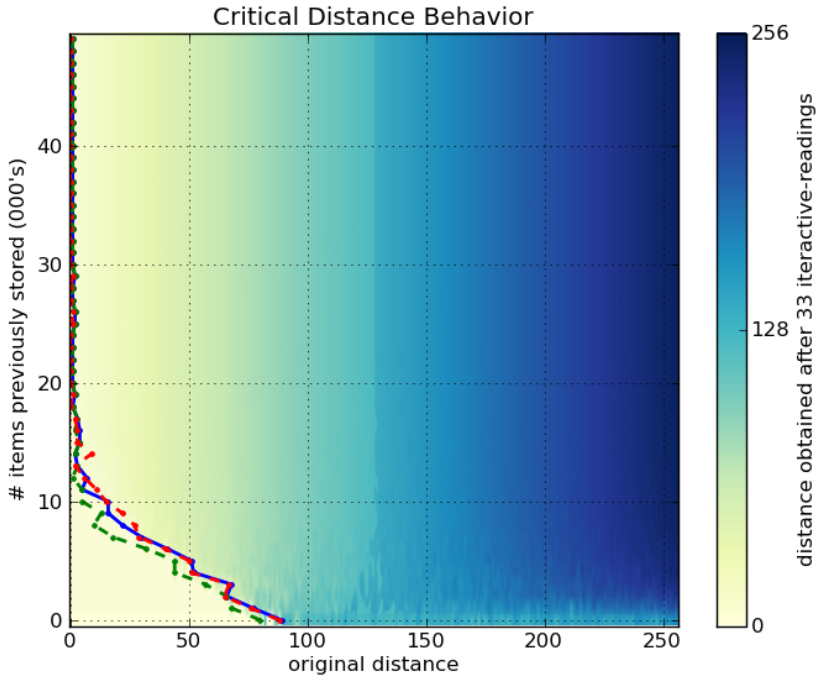

Figure 85: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 33 iterative-reading

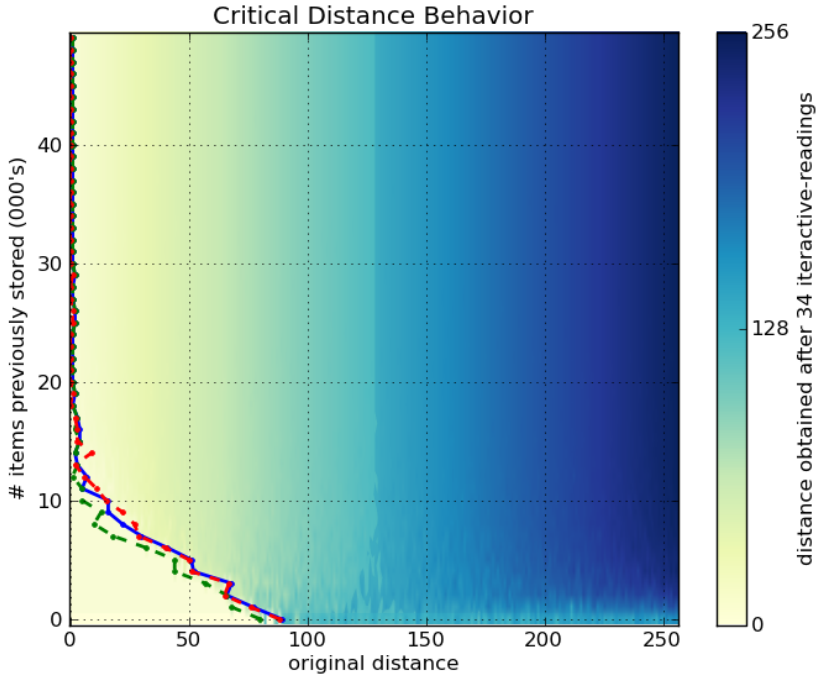

Figure 86: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 34 iterative-reading

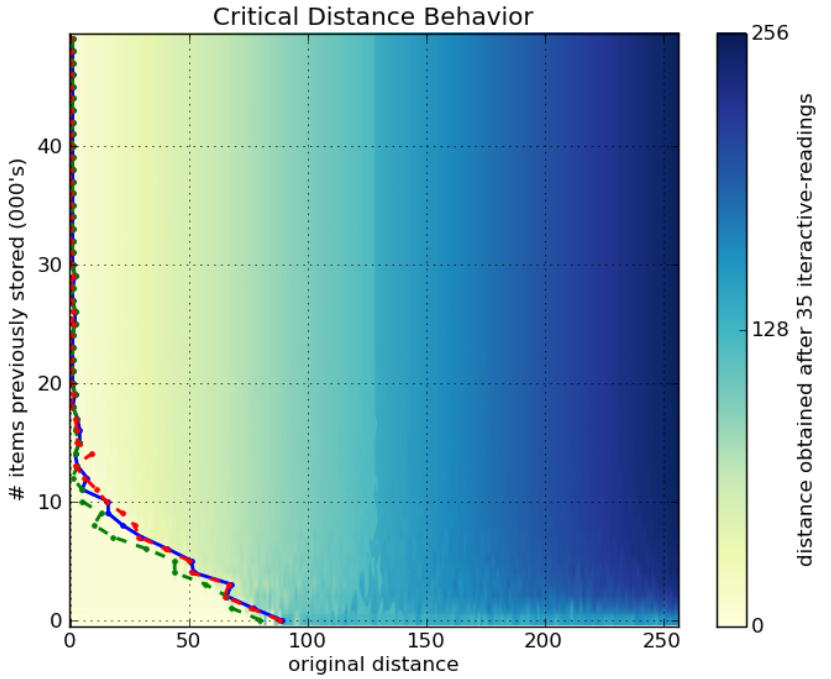

Figure 87: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 35 iterative-reading

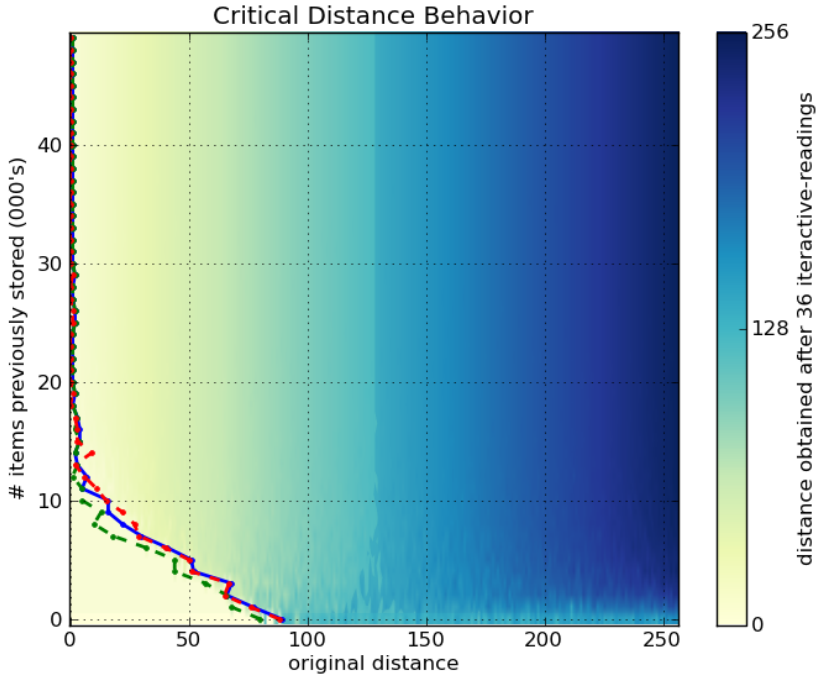

Figure 88: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 36 iterative-reading

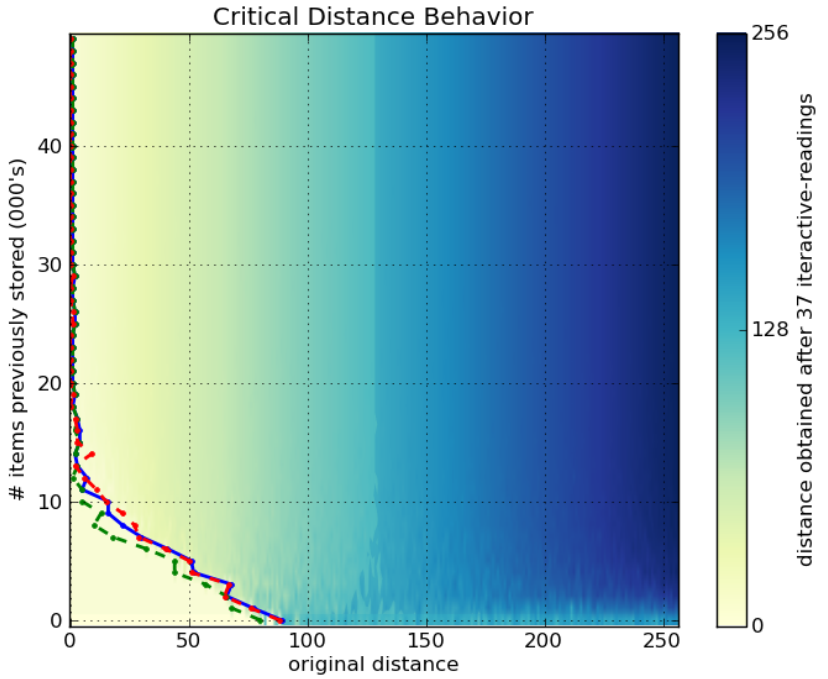

Figure 89: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 37 iterative-reading

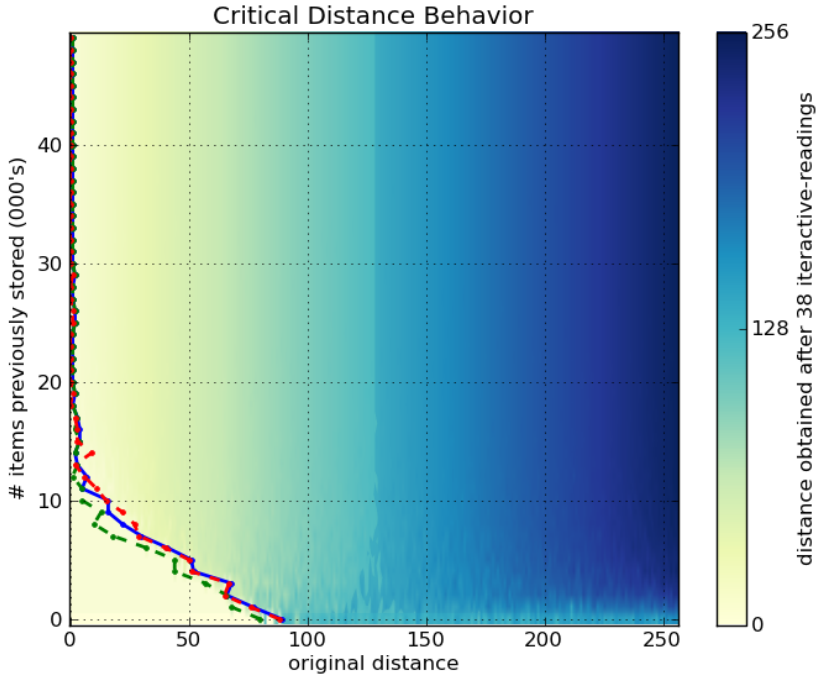

Figure 90: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 38 iterative-reading

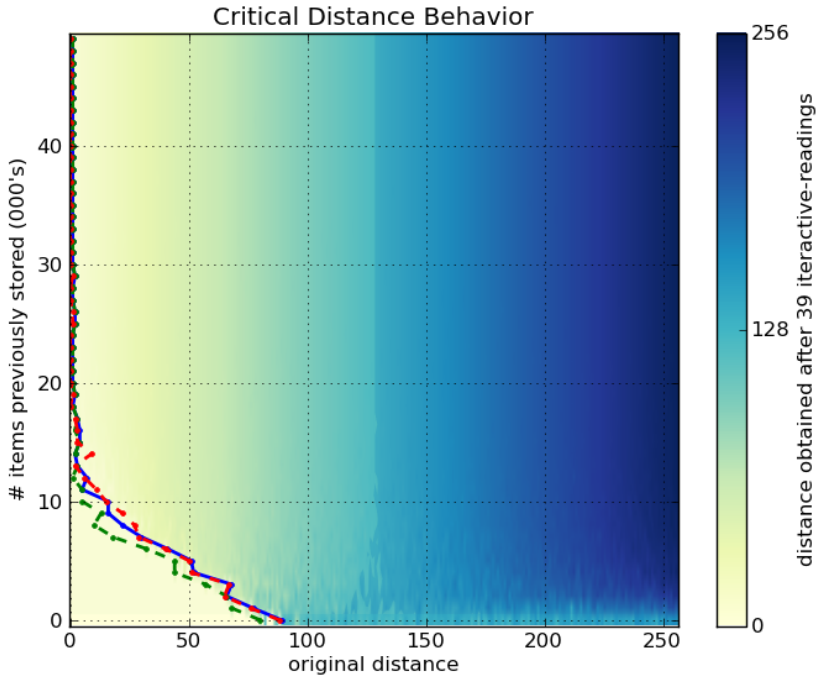

Figure 91: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 39 iterative-reading

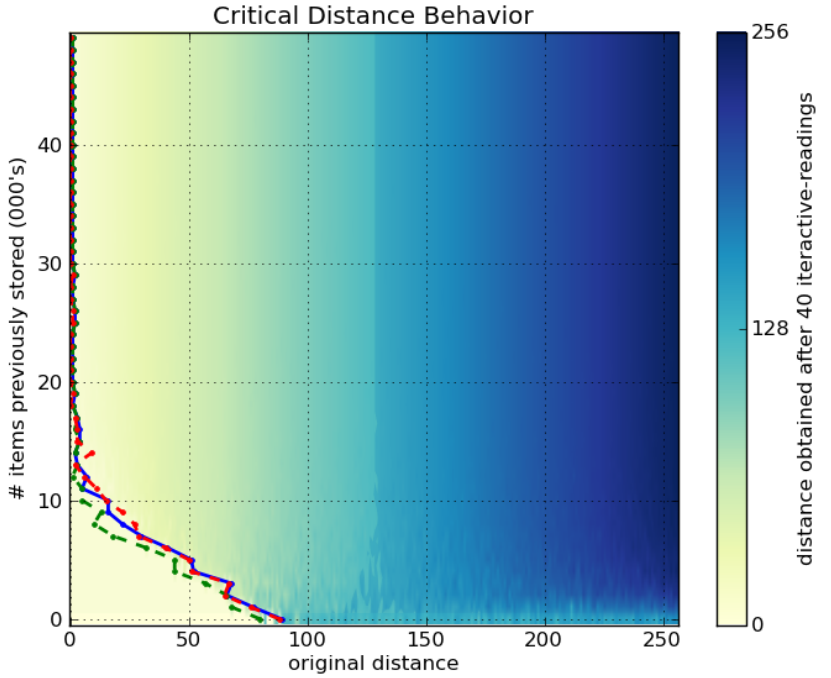

Figure 92: 256-dimensional, 1 million hard-locations, 451 access radius, 1 write of target bitstring, Kanerva's read, 40 iterative-reading

D

APPENDIX D: ADDITIONAL NUMBER OF WRITINGS RESULTS FOR 256-DIMENSIONAL MEMORY

For the sake of completeness, in this appendix we include the entire set of figures generated with 256 dimensions, 6 iterative-readings, ranging from 1 to 9 writings of target bitstring.

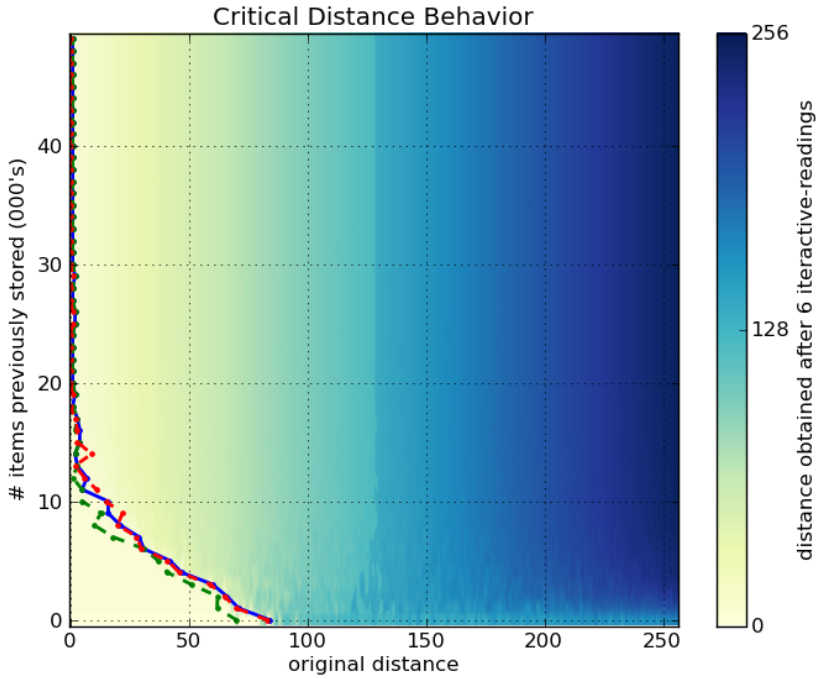

Figure 93: 256-dimensional, 1 million hard-locations, 451 access radius, 1 writes of target bitstring, Kanerva's read, 6 iterative-reading

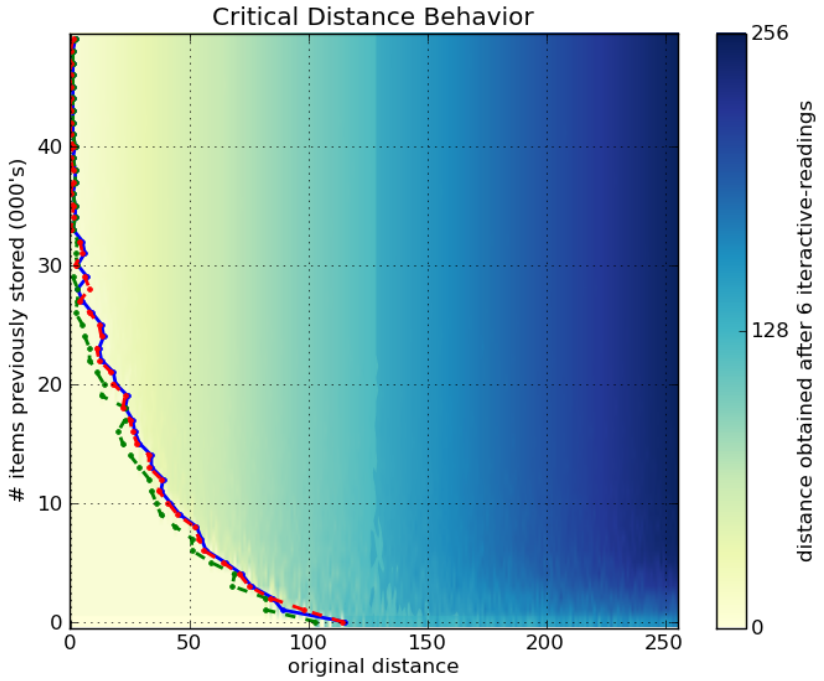

Figure 94: 256-dimensional, 1 million hard-locations, 451 access radius, 2 writes of target bitstring, Kanerva's read, 6 iterative-reading

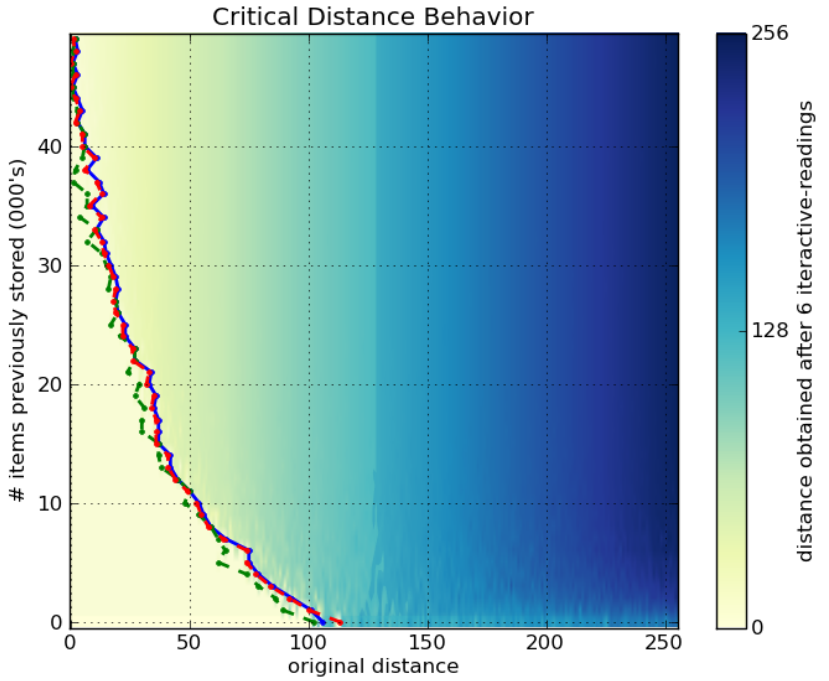

Figure 95: 256-dimensional, 1 million hard-locations, 451 access radius, 3 writes of target bitstring, Kanerva's read, 6 iterative-reading

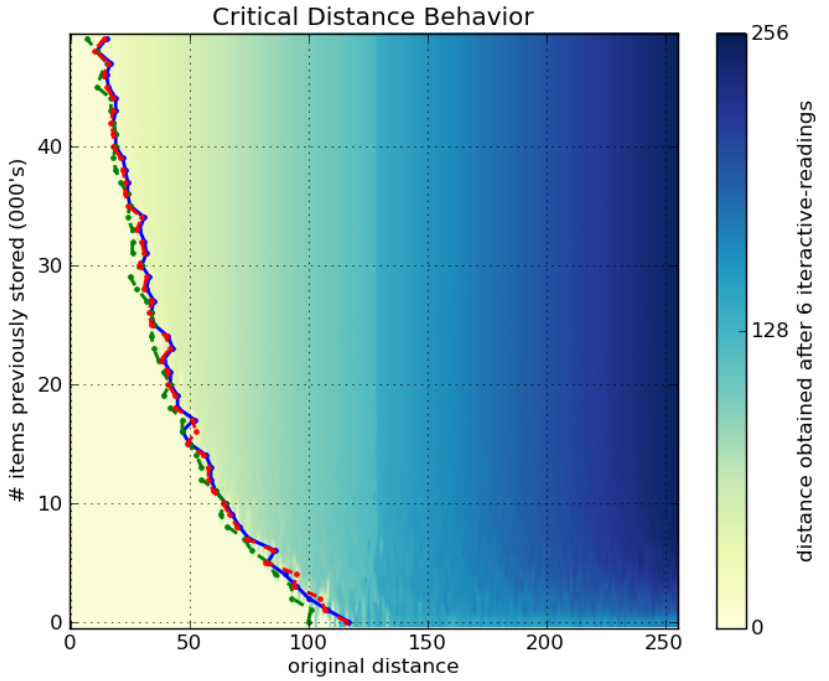

Figure 96: 256-dimensional, 1 million hard-locations, 451 access radius, 4 writes of target bitstring, Kanerva's read, 6 iterative-reading

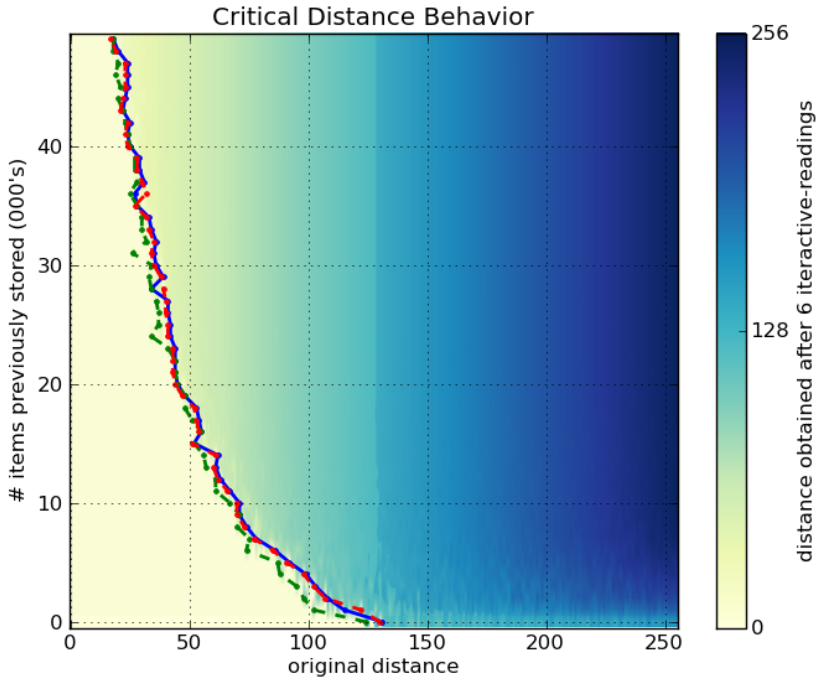

Figure 97: 256-dimensional, 1 million hard-locations, 451 access radius, 5 writes of target bitstring, Kanerva's read, 6 iterative-reading

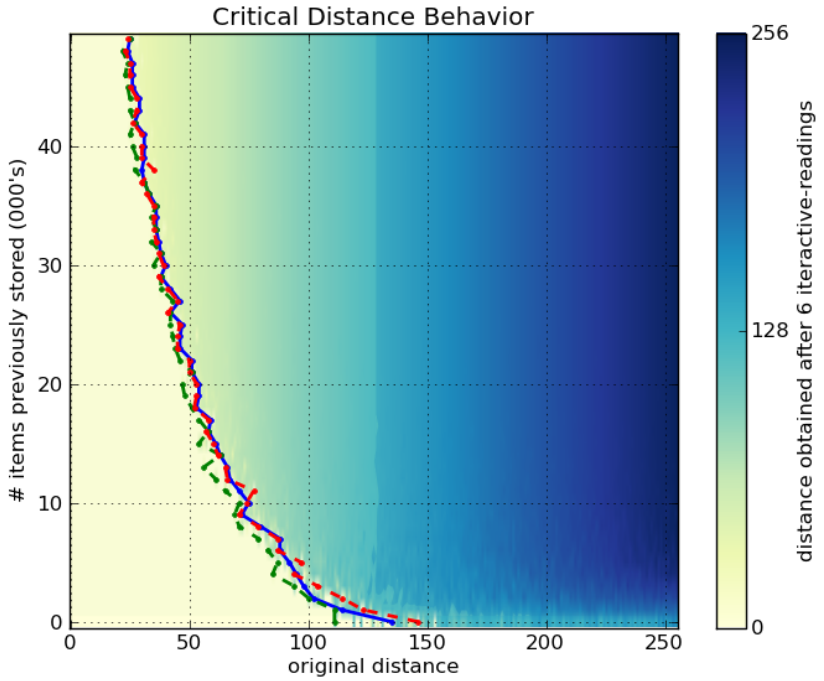

Figure 98: 256-dimensional, 1 million hard-locations, 451 access radius, 6 writes of target bitstring, Kanerva's read, 6 iterative-reading

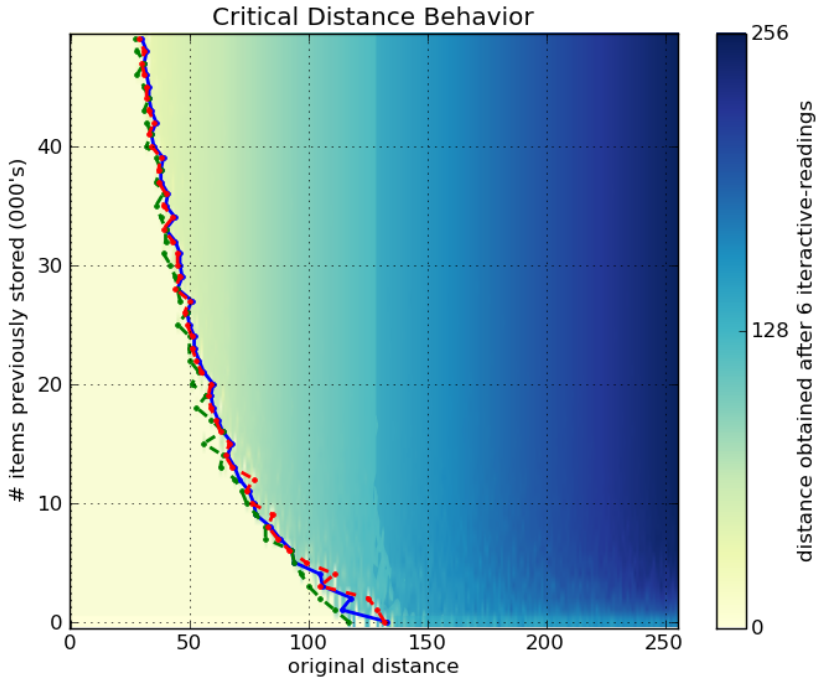

Figure 99: 256-dimensional, 1 million hard-locations, 451 access radius, 7 writes of target bitstring, Kanerva's read, 6 iterative-reading

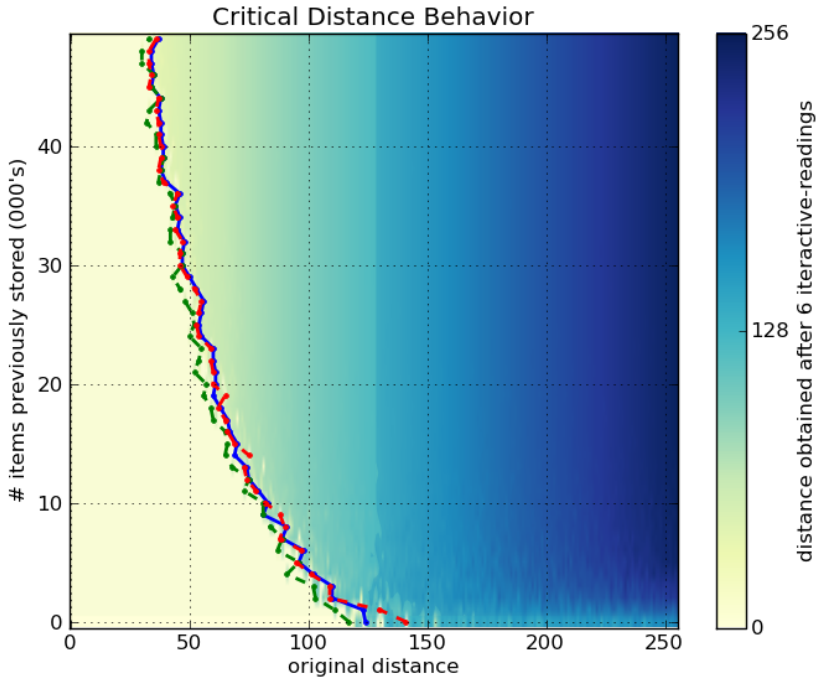

Figure 100: 256-dimensional, 1 million hard-locations, 451 access radius, 8 writes of target bitstring, Kanerva's read, 6 iterative-reading

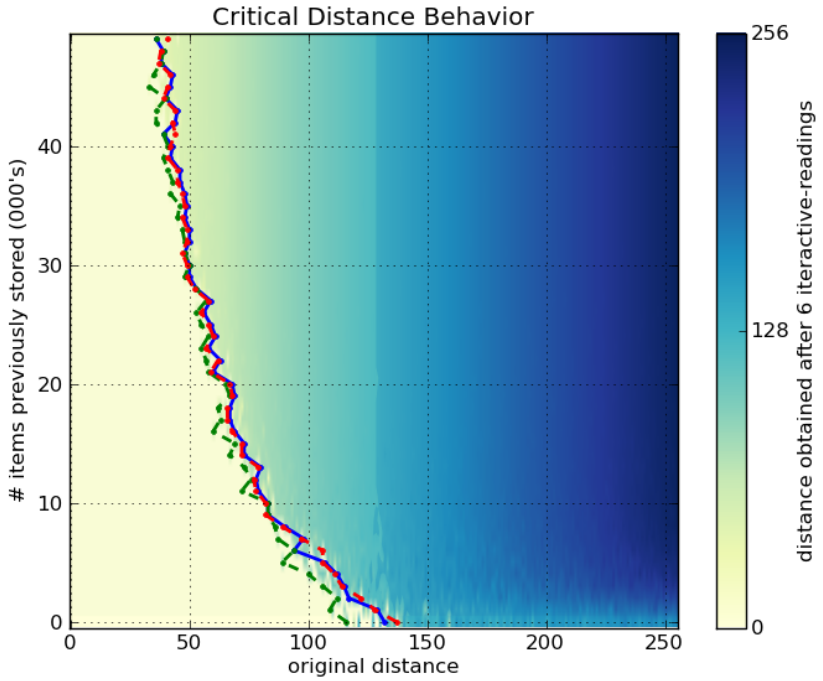

Figure 101: 256-dimensional, 1 million hard-locations, 451 access radius, 9 writes of target bitstring, Kanerva's read, 6 iterative-reading

E

APPENDIX E: ADDITIONAL NUMBER OF WRITINGS RESULTS FOR 1000-DIMENSIONAL MEMORY

For the sake of completeness, in this appendix we include the entire set of figures generated with 1000 dimensions, 6 iterative-readings, ranging from 1 to 9 writings of target bitstring.

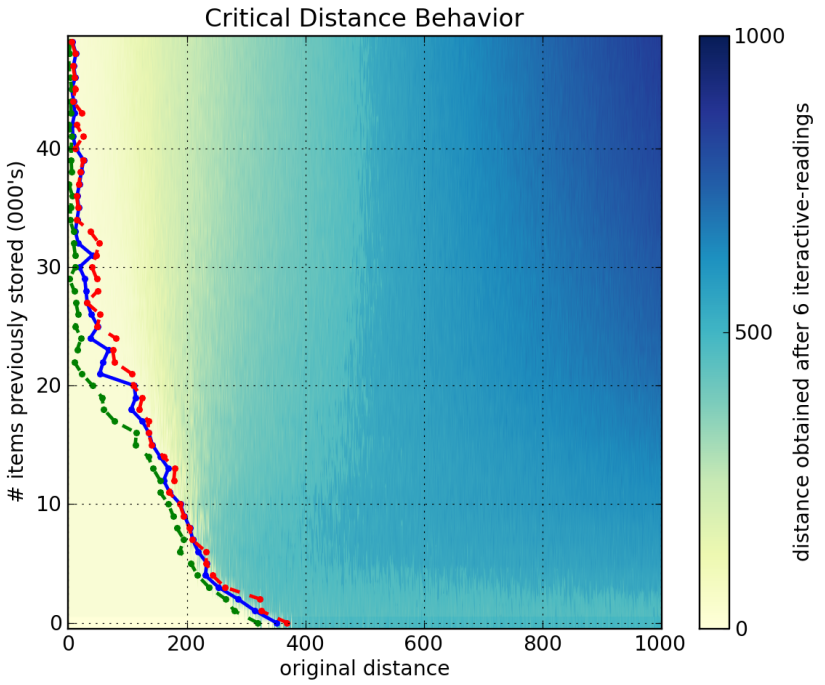

Figure 102: 1000-dimensional, 1 million hard-locations, 451 access radius, 1 writes of target bitstring, Kanerva's read, 6 iterative-reading

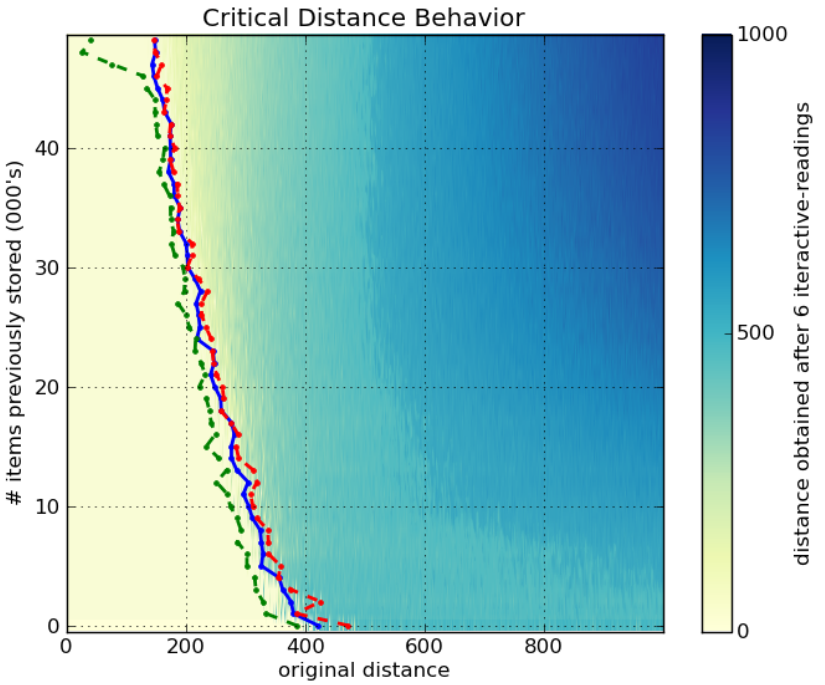

Figure 103: 1000-dimensional, 1 million hard-locations, 451 access radius, 2 writes of target bitstring, Kanerva's read, 6 iterative-reading

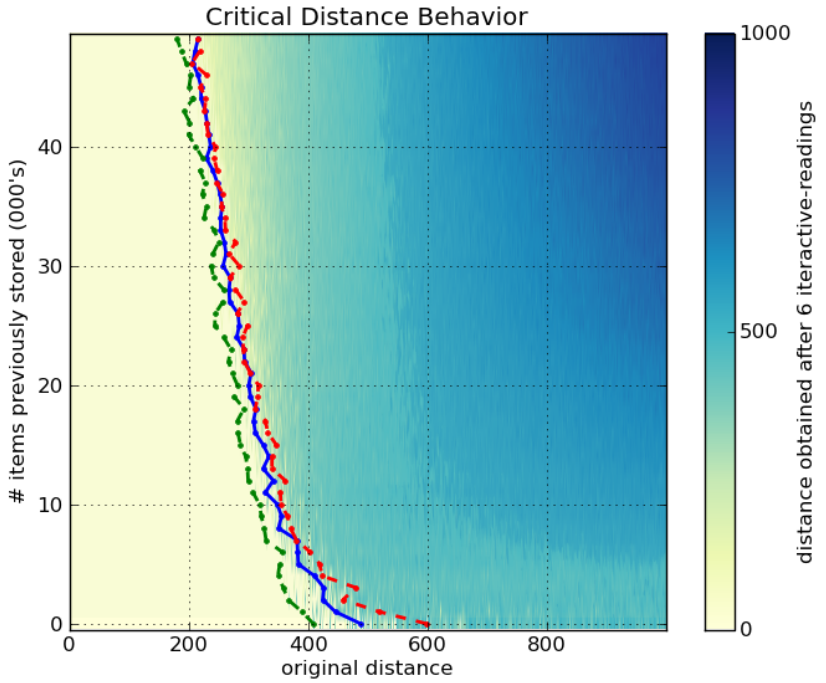

Figure 104: 1000-dimensional, 1 million hard-locations, 451 access radius, 3 writes of target bitstring, Kanerva's read, 6 iterative-reading

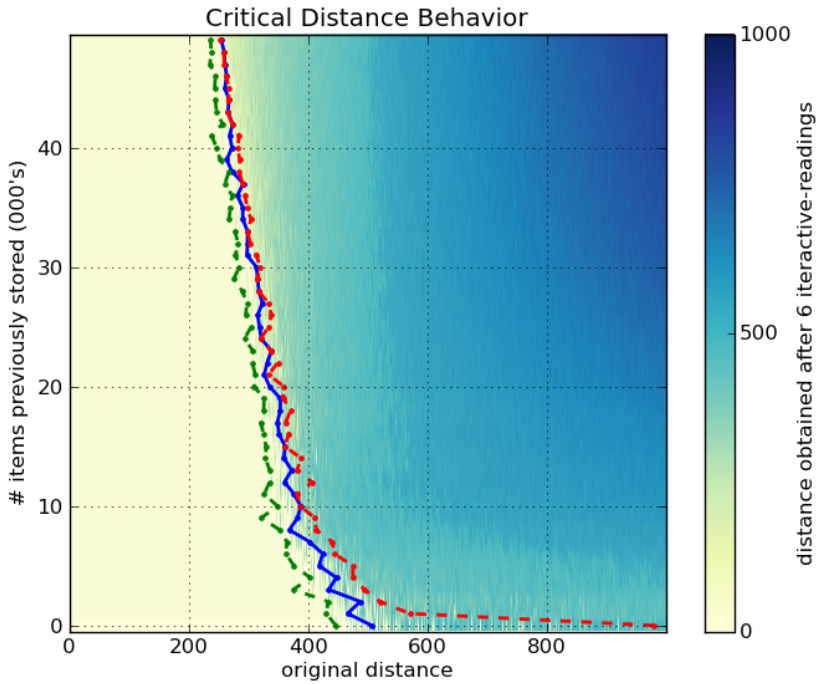

Figure 105: 1000-dimensional, 1 million hard-locations, 451 access radius, 4 writes of target bitstring, Kanerva's read, 6 iterative-reading

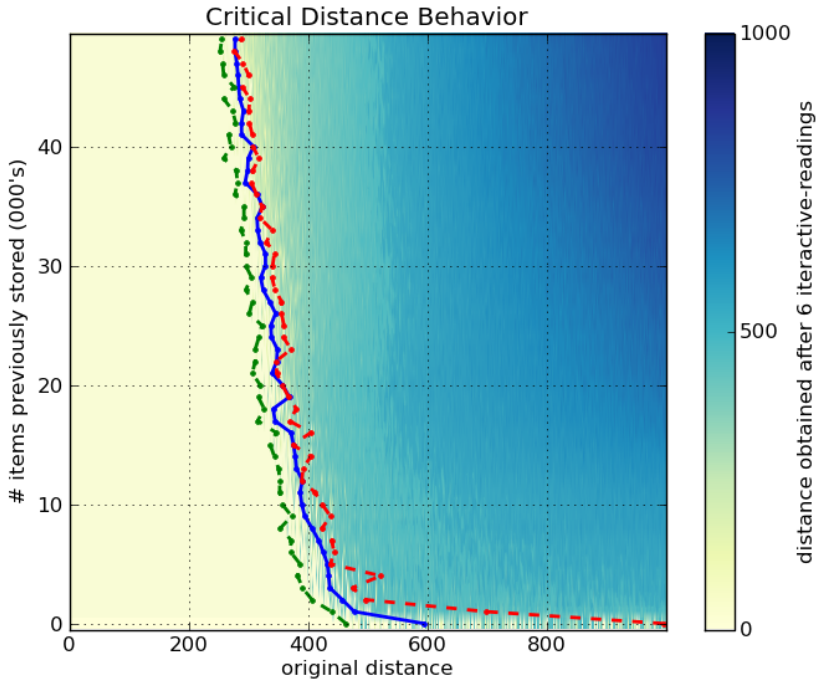

Figure 106: 1000-dimensional, 1 million hard-locations, 451 access radius, 5 writes of target bitstring, Kanerva's read, 6 iterative-reading

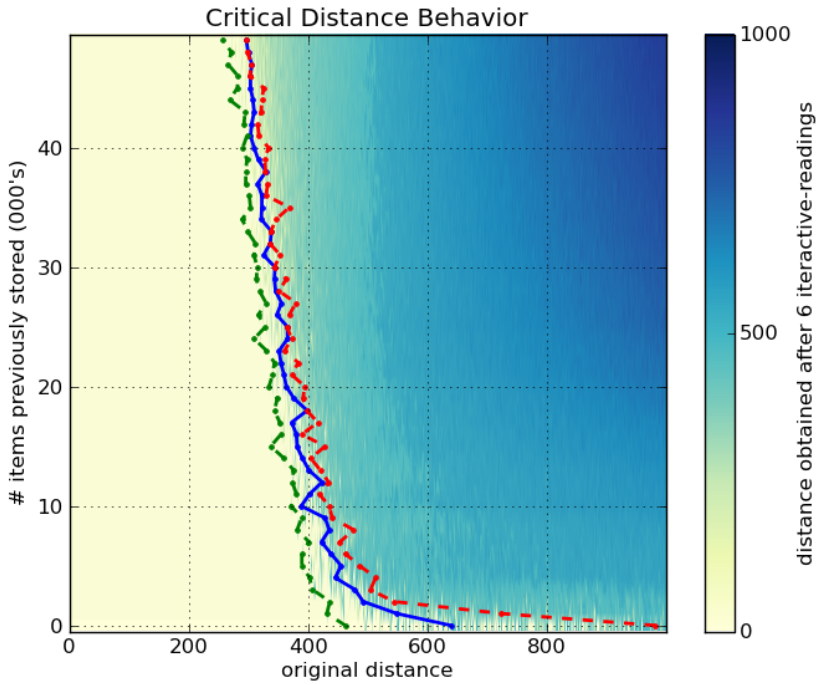

Figure 107: 1000-dimensional, 1 million hard-locations, 451 access radius, 6 writes of target bitstring, Kanerva's read, 6 iterative-reading

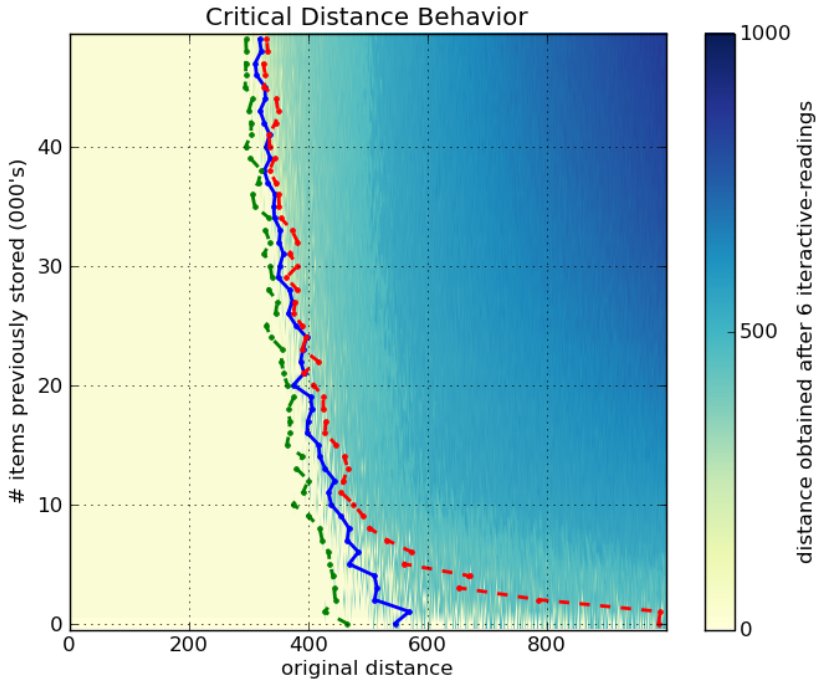

Figure 108: 1000-dimensional, 1 million hard-locations, 451 access radius, 7 writes of target bitstring, Kanerva's read, 6 iterative-reading

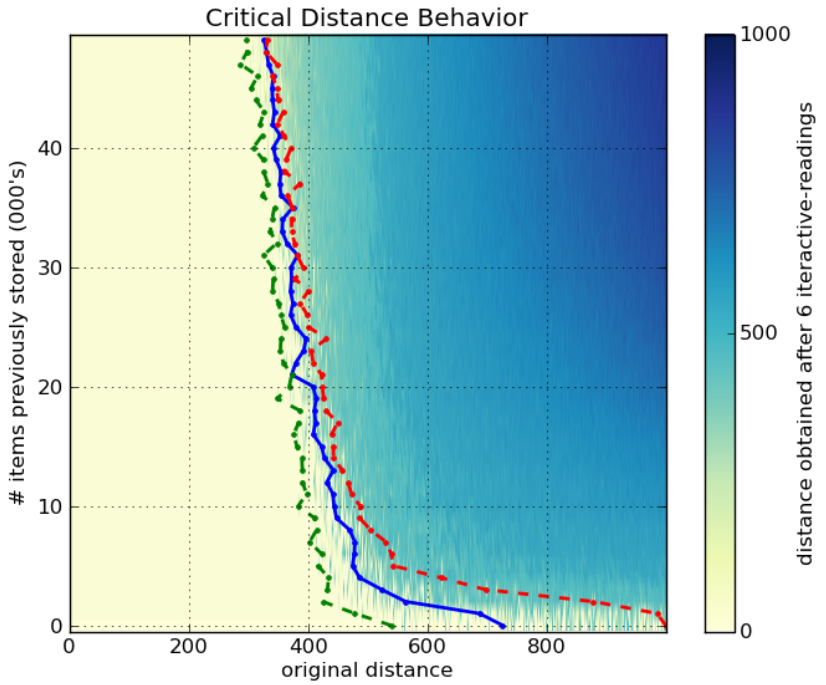

Figure 109: 1000-dimensional, 1 million hard-locations, 451 access radius, 8 writes of target bitstring, Kanerva's read, 6 iterative-reading

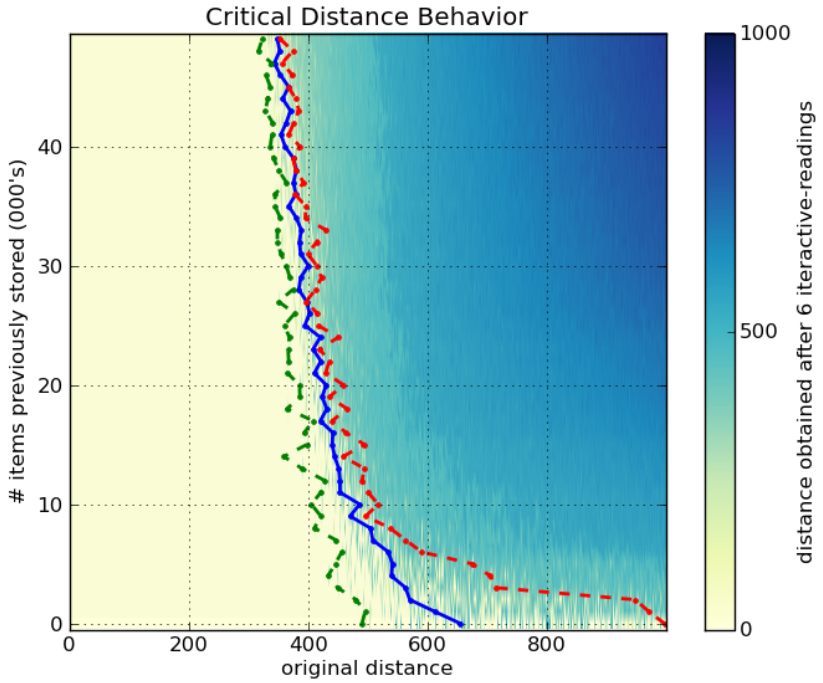

Figure 110: 1000-dimensional, 1 million hard-locations, 451 access radius, 9 writes of target bitstring, Kanerva's read, 6 iterative-reading