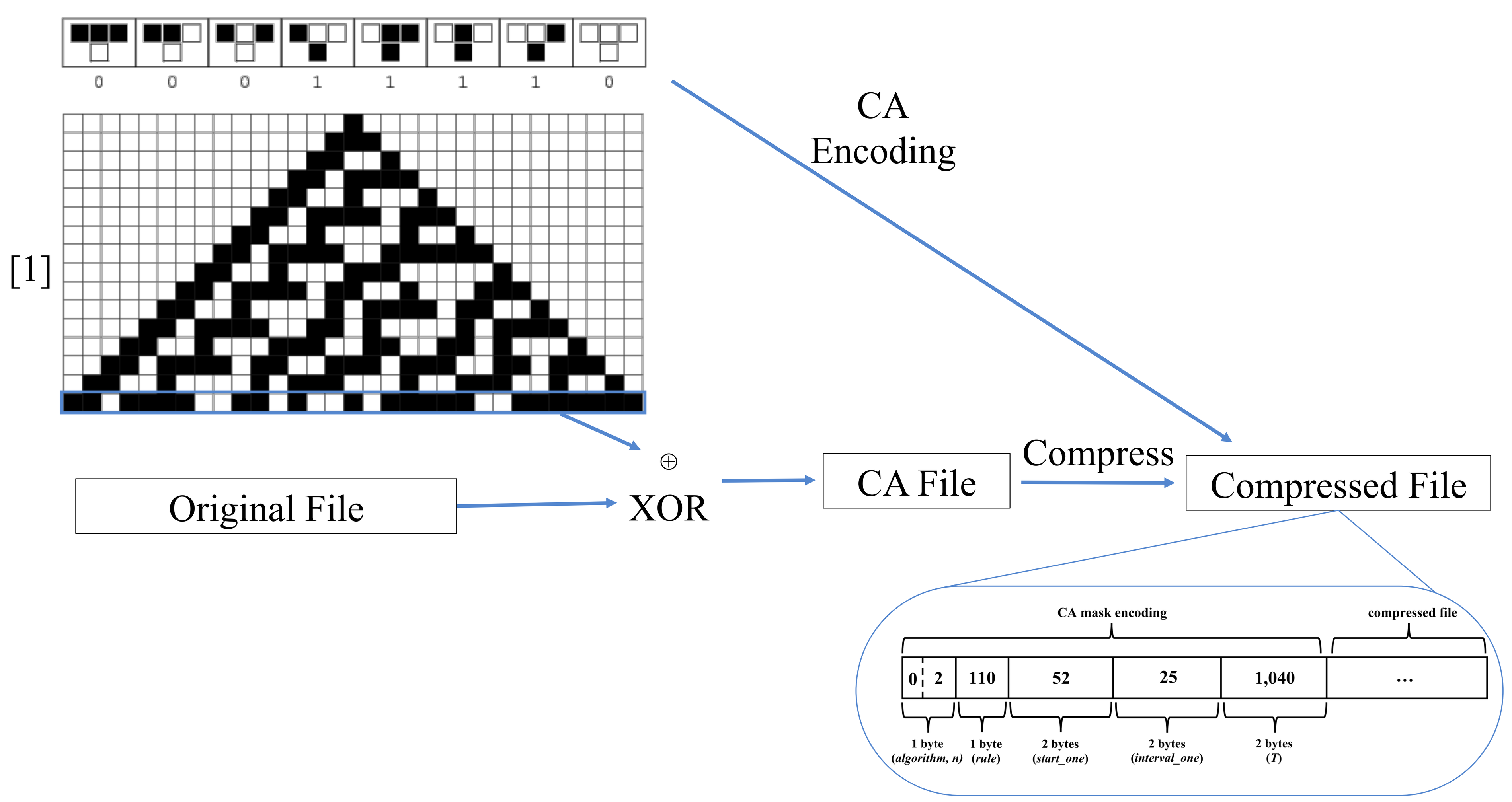
Improving File Compression Using Elementary Cellular Automata

John Albury, Richard Wales, and Annie S. Wu
Department of Computer Science, University of Central Florida

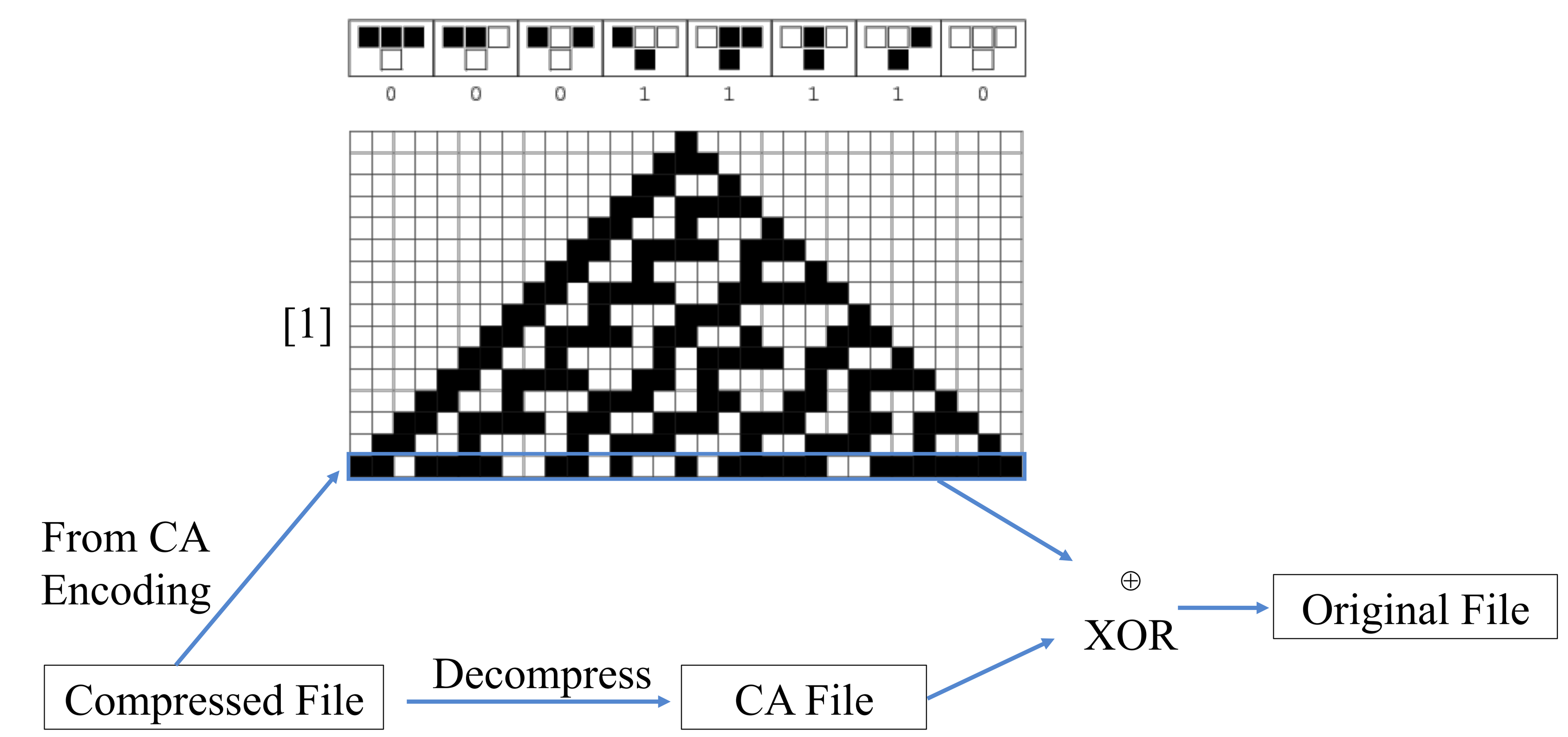
Abstract

We present a novel technique for pre-processing files that can improve file compression rates of existing general purpose lossless file compression algorithms, particularly for files on which these algorithms perform poorly. The elementary cellular automata (CA) pre-processing technique involves finding an optimal CA state that can be used to transform a file into a format that is more amenable to compression than the original file format. This technique is applicable to multiple file types and may be used to enhance multiple compression algorithms. Evaluation on files that we generated, as well as samples selected from online text repositories, finds that the CA pre-processing technique improves compression rates by up to 4% and shows promising results for assisting in compressing data that typically induce worst-case behavior in standard compression algorithms.

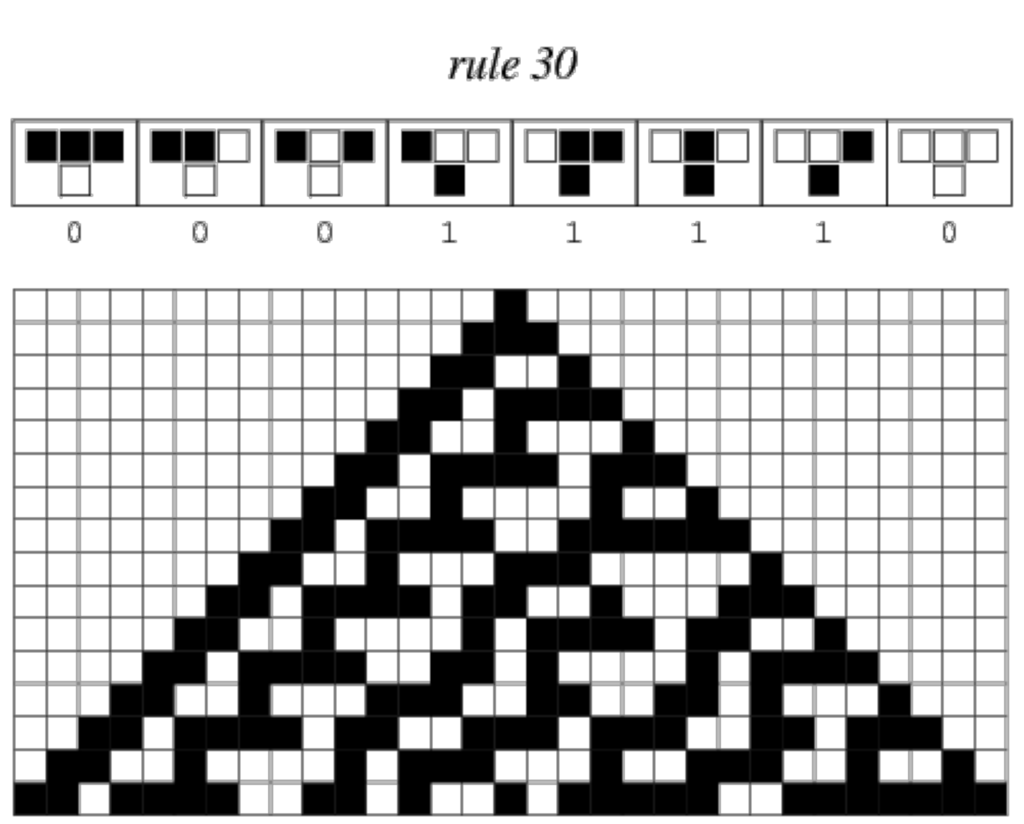
File Compression



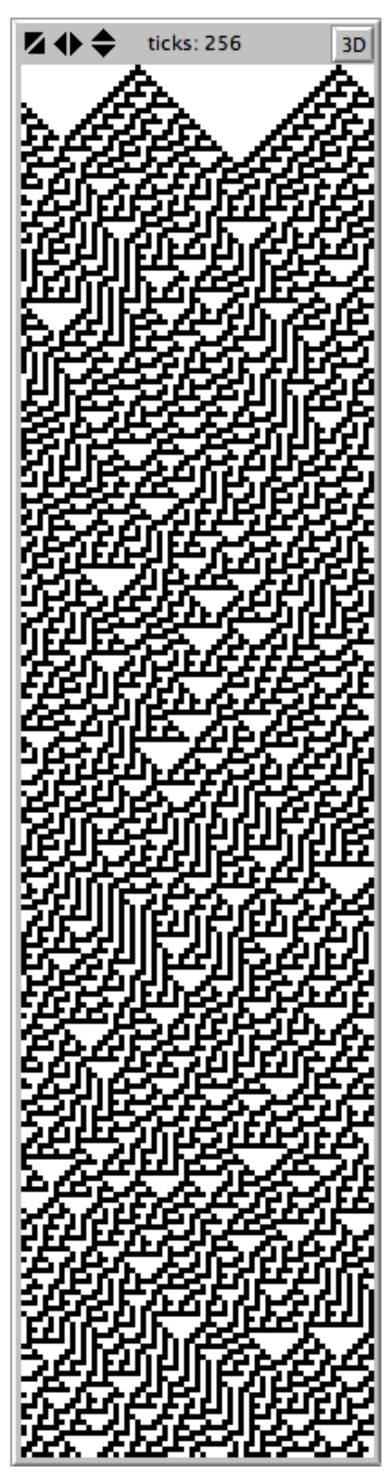
File Decompression



Cellular Automata



An example of an elementary cellular automaton, adapted from [1]



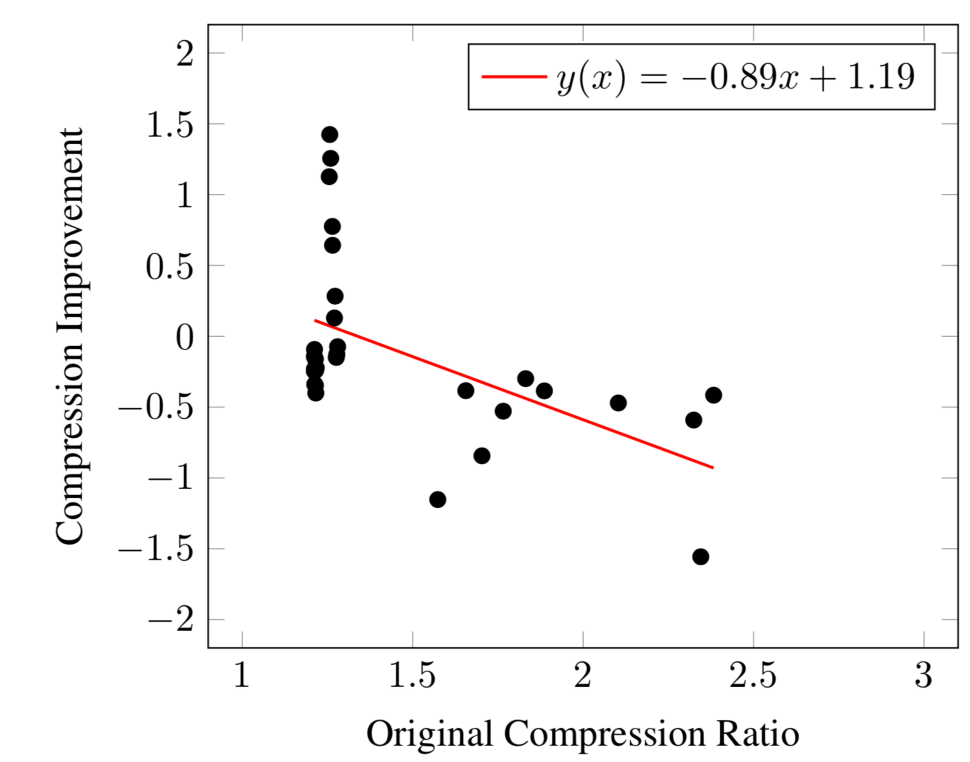
References

[1] Wolfram, S. *A New Kind of Science*. Champaign, IL: Wolfram Media, 2002

Results

File	Original Size	bzip2	gzip	xz	bzip2 %Imp.	bzip2 Δ_{avg}	bzip2 Δ_{best}	bzip2 T_{best}	gzip %Imp.	gzip Δ_{avg}	gzip Δ_{best}	gzip T_{best}	xz %Imp.	xz Δ_{avg}	xz Δ_{best}	xz T_{best}
key1	1,675B	1,414B	1,305B	1,436B	100%	+0.573%	+0.990%	21,628	0%	-0.444%	-0.307%	N/A	0%	-0.557%	-0.557%	N/A
key2	1,675B	1,407B	1,302B	1,436B	80%	+0.398%	+0.781%	11,104	0%	-0.545%	-0.461%	N/A	0%	-0.334%	-0.279%	N/A
key3	1,679B	1,408B	1,306B	1,436B	10%	-0.246%	+0.071%	9,686	0%	-0.536%	-0.383%	N/A	0%	-0.418%	-0.279%	N/A
key4	1,675B	1,407B	1,305B	1,432B	60%	+0.028%	+0.711%	11,759	0%	-0.513%	-0.307%	N/A	0%	-0.559%	-0.559%	N/A
key5	1,675B	1,415B	1,303B	1,436B	100%	+0.572%	+0.919%	10,317	0%	-0.545%	-0.537%	N/A	0%	-0.306%	-0.279%	N/A
key6	1,675B	1,409B	1,306B	1,440B	50%	+0.007%	+0.284%	10,084	0%	-0.467%	-0.383%	N/A	0%	-0.278%	-0.278%	N/A
key7	1,675B	1,411B	1,304B	1,436B	60%	+0.106%	+0.709%	3,098	0%	-0.567%	-0.460%	N/A	0%	-0.557%	-0.557%	N/A
key8	1,679B	1,410B	1,304B	1,432B	90%	+0.404%	+0.780%	4,101	0%	-0.514%	-0.383%	N/A	0%	-0.599%	-0.559%	N/A
key9	1,675B	1,406B	1,304B	1,432B	90%	+0.220%	+0.711%	13,817	0%	-0.383%	-0.307%	N/A	0%	-0.599%	-0.559%	N/A
key10	1,679B	1,416B	1,308B	1,440B	70%	+0.282%	+0.777%	4,750	0%	-0.420%	-0.306%	N/A	0%	-0.556%	-0.556%	N/A
Average	1,676B	1,410B	1,305B	1,436B	71%	+0.234%	+0.673%	10,034	0%	-0.493%	-0.383%	N/A	0%	-0.476%	-0.446%	N/A
random1	2,048B	1,661B	1,539B	1,664B	90%	+0.144%	+0.241%	16,377	0%	-0.429%	-0.390%	N/A	100%	+2.12%	+2.404%	13,306
random2	2,048B	1,658B	1,537B	1,692B	40%	+0.024%	+0.543%	8,145	10%	-0.416%	+0.325%	8,145	100%	+4.16%	+4.492%	9,122
random3	2,048B	1,658B	1,536B	1,612B	90%	+0.241%	+0.543%	19,694	10%	-0.312%	+0.391%	15,875	10%	-0.149%	+0.248%	15,874
random4	2,048B	1,655B	1,539B	1,672B	20%	-0.199%	+0.302%	8,161	20%	-0.201%	+0.715%	26,789	100%	+2.727%	+2.871%	1,256
random5	2,048B	1,664B	1,540B	1,700B	50%	+0.054%	+0.361%	2,082	0%	-0.416%	-0.260%	N/A	100%	+3.741%	+4.471%	22,515
random6	2,048B	1,664B	1,538B	1,696B	100%	+0.331%	+0.781%	16,321	10%	-0.280%	+0.780%	16,321	100%	+4.222%	+4.481%	2,956
random7	2,048B	1,658B	1,533B	1,644B	100%	+0.211%	+0.362%	8,156	20%	-0.189%	+0.783%	8,719	100%	+0.827%	+1.217%	15,951
random8	2,048B	1,664B	1,537B	1,620B	80%	+0.228%	+0.541%	9,248	0%	-0.429%	-0.325%	N/A	0%	-0.247%	0.000%	N/A
random9	2,048B	1,658B	1,537B	1,620B	50%	+0.018%	+0.422%	13,456	10%	-0.377%	+0.260%	13,456	0%	-0.025%	0.000%	N/A
random10	2,048B	1,663B	1,542B	1,636B	60%	+0.174%	+0.421%	1,876	0%	-0.324%	-0.259%	N/A	100%	+0.538%	+0.733%	3,790
Average	2,048B	1,660B	1,538B	1,656B	68%	+0.123%	+0.452%	10,352	8%	-0.337%	+0.202%	14,884	71%	+1.791%	+2.092%	10,596
ast500hr	786B	463B	450B	472B	70%	+0.389%	+1.296%	1,340	0%	-1.733%	-1.556%	N/A	0%	-1.186%	0.000%	N/A
fs417	2,018B	1,071B	1,023B	1,120B	70%	+0.177%	+1.027%	1,787	0%	-0.762%	-0.684%	N/A	0%	-0.571%	-0.357%	N/A
genetic	1,873B	1,016B	984B	1,072B	90%	+0.581%	+1.378%	14,242	0%	-0.732%	-0.610%	N/A	0%	-0.746%	-0.746%	N/A
mind6	3,216B	1,350B	1,364B	1,440B	0%	-0.741%	-0.370%	N/A	0%	-0.770%	-0.660%	N/A	0%	-0.264%	-0.208%	N/A
unifid	1,200B	506B	479B	556B	0%	-1.581%	-1.581%	N/A	0%	-1.649%	-1.461%	N/A	0%	-1.439%	-1.439%	N/A
xargs	4,227B	1,763B	1,748B	1,812B	20%	-0.068%	+0.227%	9,512	0%	-0.572%	-0.515%	N/A	0%	-0.607%	-0.607%	N/A
goddard	896B	558B	528B	632B	0%	-0.968%	-0.538%	N/A	0%	-1.288%	-0.947%	N/A	0%	-1.203%	-0.632%	N/A
ast-dorn	2,613B	1,542B	1,563B	1,632B	60%	+0.149%	+0.908%	4,775	0%	-0.627%	-0.512%	N/A	0%	-0.674%	-0.674%	N/A
ast-prog	1,672B	942B	904B	1,000B	20%	-0.361%	+0.425%	12,091	0%	-0.785%	-0.664%	N/A	0%	-0.440%	-0.400%	N/A
taxonomy	3,271B	1,572B	1,508B	1,588B	20%	-0.076%	+0.254%	24,613	0%	-0.643%	-0.531%	N/A	0%	-0.693%	-0.693%	N/A
Average	2,177B	1,078B	1,055B	1,132B	35%	-0.250%	+0.303%	9,766	0%	-0.956%	-0.814%	N/A	0%	-0.782%	-0.576%	N/A

Original Size: original size of file in bytes
bzip2: size of file (in bytes) after being compressed by bzip2
gzip: size of file (in bytes) after being compressed by gzip
xz: size of file (in bytes) after being compressed by xz
%Imp.: percentage of trials (out of 10) where our method results in a net positive effect on compression
Δ_{avg}: average percent improvement in compression when using our method compared with using the standard compression algorithm alone
Δ_{best}: best percent improvement in compression when using our method compared with using the standard compression algorithm alone
T_{best}: time step at which the best individual compression improvement is found (that is, the value of T when Δ_{best} is found)



As shown above, the compression improvement our method offers seems to have an inverse relationship with the compression ratio of the standard compression algorithm for the file being tested. Typically, standard compression algorithms perform poorest on random-like data, and this holds true for the files we test as well (the SSH keys, key1 through key10, and the randomly-generated text files, random1 through random10). These random-like files also show the highest and most consistent improvements when using our pre-processing method compared to the non-random files. Thus, this method could have intriguing implications for compressing random-like data and other types of data that typically induce worst-case behavior in standard compression algorithms.