# A Fully Reversible Data Transform Technique Enhancing Data Compression of SMILES Data

Shagufta Scanlon, Mick Ridley

# A Fully Reversible Data Transform Technique Enhancing Data Compression of SMILES Data

Shagufta Scanlon and Mick Ridley

University of Bradford, Bradford, UK
s.a.scanlon@student.bradford.ac.uk, m.j.ridley@bradford.ac.uk

**Abstract.** The requirement to efficiently store and process SMILES data used in Chemoinformatics creates a demand for efficient techniques to compress this data. General-purpose transforms and compressors are available to transform and compress this type of data to a certain extent, however, these techniques are not specific to SMILES data. We develop a transform specific to SMILES data that can be used alongside other general-purpose compressors as a pre-processor and post-processor to improve the compression of SMILES data. We test our transform with six other general-purpose compressors and also compare our results with another transform on our SMILES data corpus, we also compare our results with untransformed data.

**Keywords:** SMILES, Data Transform, Data Compression.

## 1 Introduction

The Simplified Molecular Input Line Entry System (SMILES) language was developed to represent two-dimensional molecular structures in a concise and compact way allowing for storage and processing improvements. General-purpose compressors allow for further reductions in storage and processing costs [23], [8].

With the continuous expansion of chemical databases [15] and the need for efficient storage and searching of molecular structure representations, such as SMILES [23], [8], storage and processing costs of these representations need to be further improved.

Data can be transformed by exploiting the specific information contained in the data to its advantage. Transformed data can be used alongside general-purpose compression techniques to further improve compression results [21], [4], [20].

We make the following contributions in this paper:

- We present our SMILES-specific transform designed to enhance the compression of SMILES data when used with other general-purpose compressors.
- We provide results from using general-purpose compression techniques on a breakdown of different SMILES transform scenarios and a combination of techniques used in our SMILES transforms.

- We provide a comparison of the results from our SMILES transforms with Word Replacing Transforms (WRT) [21], [12], [20] optimized for different compression algorithms and also on untransformed data.

The remainder of this paper is structured as follows. Section 2 discusses molecular structure representations and how compression has been used on SMILES data. Sections 3 to 5 introduce the SMILES chemical language, general-purpose compression and data transformation techniques, respectively. Section 6 illustrates our SMILES-specific transform design, grammar and architecture. Section 7 provides details of the experiments conducted, including the data collected, testing environment, metrics computed, experiment methodology and results. The results shown demonstrate the benefits of using compression over transformed data. The actual transformation results themselves have been omitted to help maintain the focus of this paper, which is to improve the compression of SMILES data using SMILES-specific transforms, and also due to space limitations. Section 8 discusses the key findings and concludes this paper.

## 2 Molecular Structure Representations

Several molecular structure representations have been implemented over the years. The most popular linear representations include SMILES, which is human-readable, and International Chemical Identifier (InChI), which is machine-readable. Other linear representations include Wiswesser Line Notation (WLN), Representation of Organic Structures Description Arranged Linearly (ROSDAL), SYBYL Line Notation (SLN), Modular Chemical Descriptor Language (MCDL) and InChIKey [14], [18], [10]. Other types of representations include MOL and SDF formats for single and multiple molecules, respectively. MOL and SDF consist of coordinates and connections between atoms [14]. The authors in [8] suggested that SMILES representations use between 50% and 70% less storage space in comparison to their equivalent connection tables [8].

The authors in [14] developed a molecular structure representation using barcodes in SMILES format. Barcodes were chosen to handle errors and SMILES was chosen due to its human-readability and compact nature in comparison to other molecular representations, such as MOL and SDF. However, they argued that in the future with the potential increase in error correction levels required, the compression of SMILES would be inevitable to avoid an increase in data storage. They discussed the use of Lempel-Ziv-Welch (LZW), a lossless dictionary-based compression algorithm, and found that data reduction with LZW can be achieved approximately by a factor of two. As well as compression, the authors also discussed the use of a more compact representation to SMILES instead, known as the Automatic Chemical Structure (ACS). ACS provides a condensed representation of a molecular structure by assigning unique identifiers to molecular substructures or fragments. However, as molecular information from a SMILES representation is omitted from the ACS encoded barcode, backward mapping from ACS to SMILES is important [14].

## 3 SMILES Data

SMILES is a linear chemical notation language originally developed by Weininger in 1988 [23] and further extended by Daylight Chemical Information Systems [8]. It is used by practitioners in the Chemoinformatics field to represent a linear form of two-dimensional molecular structures [23].

Table 1 provides a summary of the basic representation rules for SMILES notations and some examples. SMILES notations are essentially represented using ASCII characters in linear format with no spaces. Molecular structures can have several different and valid SMILES representations and SMILES representations are not unique. Although, unique representations can be generated using a canonicalization algorithm [23], [8].

Atomic symbols are used to represent atoms. Non-organic atoms can be represented inside square brackets, also included in the brackets are the number of Hydrogen atoms and charges declared for the atoms. The range of elements within the organic subset can be depicted with the square brackets omitted, this indicates that Hydrogen atoms are present for these atoms even without the square brackets. Aliphatic and aromatic atoms are represented in uppercase and lowercase characters, respectively [23], [8].

Neighboring atoms are attached to each other by single bonds or aromatic bonds, which can be excluded, unless otherwise specified by double or triple bonds. Branches are parenthesized and can be characterized as nested or stacked. To symbolize a cyclic structure, a single bond must be broken inside a cyclic ring. Cyclic ring opening and closing are determined by numbers assigned following the ring opening and ring closing atomic symbols. A period characterizes the separation of disconnected structures [23], [8].

**Table 1.** Generic SMILES Representation Rules and Examples [23], [8]

| Generic SMILES Rules | Example Representations |
|---|---|
| Non-Organic Atoms | [S], [H+] |
| Aliphatic Organic Atoms | B, C, N, O, P, S, F, Cl, Br, I |
| Aromatic Organic Atoms | b, c, n, o, p, s |
| Single Bonds | C-C, CC |
| Double Bonds | C=C |
| Triple Bonds | C#N |
| Aromatic Bonds | c:c, cc |
| Nested or Stacked Branches | C=CC(CCC)C(C(C)C)CCC |
| Ring Closures | C1CCCCC1 |
| Disconnections | [Na+].[0-]c1ccccc1 |

## 4 General-Purpose Data Compressors

General-purpose compressors can be used to universally compress data of textual format [4], [20]. Table 2 shows a summary of the compression algorithms and techniques used in each general-purpose compressor used in our experiments.

Lossless compression techniques preserve the integrity of data by ensuring the original data is fully reconstructed from the compressed data on decompression [20]. Two common forms of lossless compression techniques are statistical and dictionary-based approaches. Statistical approaches to compression include the Huffman encoding algorithm which is a technique that involves the substitution of frequent characters with shorter codewords [4], [20]. Dictionary-based approaches involve the substitution of words or phrases with their corresponding indices in the dictionary. The Lempel-Ziv compression schemes which include LZ77, LZ78, LZW and LZMA to name a few, are all dictionary-based approaches [4], [19], [20].

Within dictionary-based approaches, static dictionaries can be suitable when information about the data being processed is already known and can be prepared in advance. They have the potential to improve compression results as the technique is one-pass, so an information gathering phase prior to compression would not be required, and the information would be readily available in the dictionary to process. However, the fixed nature of the dictionaries have the potential to negatively impact storage costs. A semi-static dictionary approach uses two-passes, the first to gather statistics from the data source and prepare the dictionary and the second to actually compress the data. The adaptive dictionary technique dynamically rebuilds the dictionary during compression [4], [20].

**Table 2.** General-Purpose Compressors

| Compressors | Compression Algorithms/Techniques |
|---|---|
| 7Zip [1], [19] | Back-End: LZMA (Default), LZMA2, PPMd, BZip2, DEFLATE |
| BZip2 [19] | Uses Burrows-Wheeler Block Transformation (BWT) and Huffman |
| GZip [19] | Based on DEFLATE (LZ77 and Huffman) |
| PPMd [22], [7], [20] | Based on Prediction by Partial Matching (PPM), Adaptive Statistical Technique using Context Modelling and Prediction |
| PPMVC [22], [12], [20] | Based on PPM with Variable-Length Contexts |
| ZPAQ [24], [17] | An Extension to PPM, Back-End: LZ77 (Default), BWT |

## 5 Data Transform Techniques

Data transformation can be added as an extra pre-processing and post-processing step during data compression to enhance compression results. Where general-purpose compressors can be used to treat all types of data as text and compress them accordingly, modelling a data transformation technique on a specific type of data can provide far better compression results when used alongside other compressors [21], [4], [20].

Data-specific transforms have been developed for this purpose for different types of data. For example, to transform textual data, WRT was developed by Skibiński [21], [12], [20] to pre-process English language text based on matching words in the dictionary and replacing them with codewords. It uses capital conversion techniques, dictionary sorting according to word frequency, q-gram replacement whereby frequent q consecutive characters are replaced with shorter symbols, End-of-Line (EOL) symbol conversion to spaces, data protection techniques such as using a data filter, and encloses words with spaces. WRT was extended to Two-Level Word Replacing Transform (TWRT) to add data dictionaries that were specific to the type of data being processed, for example a dictionary containing commands from a Java programming language [21], [20]. WRT transforms optimized for BWT, LZ77, PAQ and PPM [12], [20] have been compared with our SMILES transforms in our experiments.

## 6 SMILES-Specific Data Transform

We have developed a transform language specific to SMILES notations in order to improve the compression of SMILES data. The design and techniques used in this transform was aimed at providing a good balance between data storage and processing costs, whilst maintaining data integrity. As with the use of general techniques such as the substitution of n-grams and the addition of prefixes, SMILES-specific techniques such as substituting atomic symbols with their equivalent atomic numbers

was used. The atomic symbol conversion to atomic numbers, used in stage three of our transform, has an advantage as it allows the practitioner to process the data in its transformed state on decompression.

### 6.1 SMILES Transform Properties

The following highlights the properties held by the developed transform:

- Storage Costs Reduction – Ensures a reduction in storage costs when the transformed data is compressed with other general-purpose compressors.
- Processing Time Reduction – Compression and decompression times are reduced when compression is used over the transformed data.
- No Ambiguity – Handles any ambiguous SMILES tokens, particularly with aromatic elements.
- Fully Reversible – Data is fully reversible with data integrity preserved.

### 6.2 SMILES Transform Phases

The following shows the techniques used in the different phases of the transform:

1. N-Grams[1] – Frequent SMILES tokens from two to eleven characters in length are replaced with either other frequently used characters already used in the data, or with other unused characters.
2. Number Prefixes – Prefixes are added to SMILES tokens of numerical format using either other frequently used characters in the data or other unused characters. Note that this step is carried out prior to the next phase in order to avoid ambiguity and conflicts with atomic numbers.
3. Atomic Numbers and Prefixes – SMILES tokens that contain atomic symbols are matched and converted to their corresponding atomic numbers in the periodic table to ensure that there is no ambiguity. Atomic numbers are also prefixed with either other frequently used characters in the data ensuring no ambiguity or other unused characters. Aromatic elements are also converted to their corresponding atomic numbers to ensure consistency and are transformed last preserving the no ambiguity property.

### 6.3 SMILES Transform Representations

Table 3 illustrates the grammar used for our SMILES transform. Note that 2-grams have only been used for testing the first transformed representation to reduce entropy. In order to distinguish between aromatic elements and other elements, it is assumed that SMILES tokens that contain any aromatic elements would only do so in a manner that they do not conflict and cause direct ambiguity with any other elements. For example, the atomic symbol for Cobalt is Co, if this symbol was to appear in a

[1] The concept of using N-Grams was taken from [5], [20] and [21].

SMILES string, then for our transform it has been assumed that this would be a two-letter symbol for Cobalt and not symbols for aliphatic Carbon, C, and aromatic Oxygen, o.

**Table 3.** SMILES Transform Grammar Applied to Different Scenarios[2]

| Transform Scenarios | SMILES Tokens | 1st Transform Grammar | 2nd Transform Grammar | 3rd Transform Grammar |
|---|---|---|---|---|
| 2-Grams<br>3 ... 11-Grams | CC<br>CCC ...<br>CCCCCCCCCCC | ¬<br>¬ ...<br>¬ | ()<br>() ...<br>() | |
| 2-Grams<br>3 ... 11-Grams | =C<br>C=C ...<br>C1=CC=CC=C1 | \|<br>\| ...<br>\| | []<br>[] ... <br>[] | |
| Number Prefixes (n) | n | £n | ==n | |
| Element (e) Conversion and Atomic Number (n) Prefixes | e | ;n, ~n, :n, <n, >n | *n, **n, ***n, ****n, *****n | [[n, [[[n, [=[n, (=)n, [=]n |

The following are examples of SMILES data and their equivalent transform representations to illustrate the transform scenarios further:

- 7-Grams:
  - SMILES: C1CC(CNCCCCCCC)CCC1CNCCCCCCC
  - 1st Transform: C1CC(CN¬)CCC1CN¬
  - 2nd Transform: C1CC(CN())CCC1CN()
  - SMILES: C(N)(=O)OC(C#C)(C1=CC=CC=C1)C2=CC=C(Cl)C=C2
  - 1st Transform: C(N)(=O)OC(C#C)(|C=C1)C2=CC=C(Cl)C=C2
  - 2nd Transform: C(N)(=O)OC(C#C)([]C=C1)C2=CC=C(Cl)C=C2
- Number Prefixes:
  - SMILES: C1=CC=CC(=C1)CCN(C)N=O
  - 1st Transform: C£1=CC=CC(=C£1)CCN(C)N=O
  - 2nd Transform: C==1=CC=CC(=C==1)CCN(C)N=O
- Atomic Numbers:
  - SMILES: CCC(Br)(CC)C(=O)NC(=O)NC(C)=O
  - 1st Transform: ~6~6~6(:35)(~6~6)~6(=~8)~7~6(=~8)~7~6(~6)=~8

[2] The prefixes added to the atomic numbers in the second transform grammar was based on the star encoding '*' scheme proposed in [16] and also described in [20].

- 2nd Transform: *6*6*6(**35)(*6*6)*6(=*8)*7*6(=*8)*7*6(*6)=*8
- 3rd Transform: [[6[[6[[6([[[35)([[6[[6)[[6(=[[8)[[7[[6(=[[8)[[7[[6([[6)=[[8

### 6.4 SMILES Transform Architecture

Figure 1 illustrates the SMILES transform architecture developed in Java. The transform parses the input SMILES tokens, stores the converted n-grams into a dictionary, uses regular expressions for adding prefixes to numbers, and stores the converted atomic elements along with their prefixes in dictionaries. The dictionaries created are static as the information is already known to us regarding which n-grams to use and the atomic numbers to use to replace the atomic symbols. It is intended that this technique will be extended to use semi-static or adaptive dictionaries in future work to further improve compression.

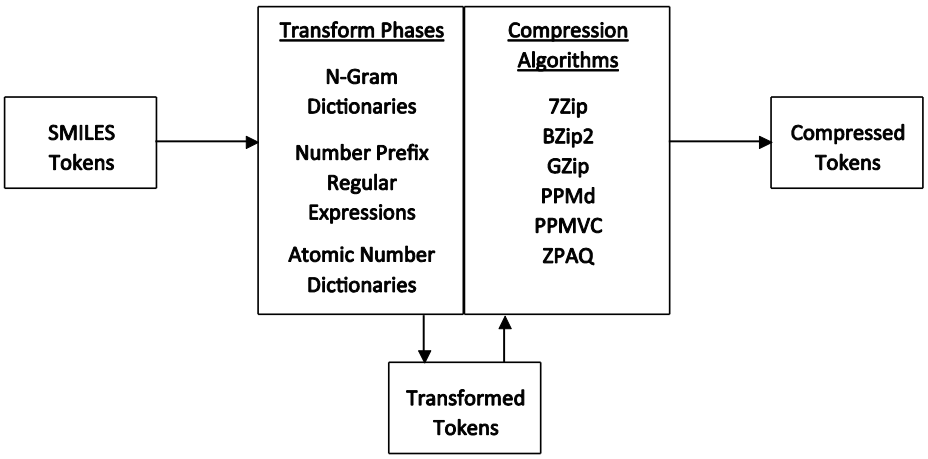


**Fig. 1.** SMILES Transform Architecture

## 7 Experiments

### 7.1 Data Collection

The experiments in this study were carried out on SMILES data extracted from 41 publicly available toxicology datasets. The datasets[3] were obtained from Bergstrom Melting Point [6], Carcinogenic Potency Database [6], CPDB [13], DSSTox[4] [9], Fathead Minnow Acute Toxicity [6], FDA's Carcinogenicity Studies [6], Fontaine Factor Xa [6], HERG [6], Huuskonen [6], Karthikeyan Melting Point [6], Li Blood-Brain-Barrier Penetration [6], National Toxicology Program [6], Stahl [6] and Tox-

[3] All whitespaces were removed.

[4] SMILES data was taken from the Structure SMILES column rather than the Parent SMILES column. Any incomplete or non-SMILES strings were removed.

Benchmark [2]. The SMILES data from all these files were combined into a single file and then multiplied to create a total of 12 files of sizes ranging from 1.71MB to 20.48MB. These files formed the basis of the data used in our experiments. In total, our data corpus consisted of 384 files. 288 files were used to provide a breakdown of our SMILES transform scenarios, these included 228 n-gram, 24 number prefix and 36 periodic number transformed files. 96 files were used to compare our SMILES transforms with other transformed and untransformed files, these included 36 SMILES and 48 WRT transformed files and 12 untransformed files.

### 7.2 Testing Environment

The following testing environment was used for the experiments:

- Operating System: Windows 7 Professional 64-bit OS.
- Processor: Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz.
- Memory: 6.00GB (5.88GB usable).

### 7.3 Compression Metrics

The following metrics were computed in Java for these experiments:

- Compression Ratios – The size of the compressed files divided by the size of the uncompressed files.
- Compression Times – The average time taken for each compressor to compress the transformed files and untransformed files.
- Decompression Times – The average time taken for each compressor to decompress the compressed files.

### 7.4 Methodology

The compressors used on all transformed files were 7Zip [1], BZip2 [3], GZip [11], PPMd [7], PPMVC [12] and ZPAQ [24]. All compressors were used with their default settings, it is intended that these settings will be changed in future work to provide for more detailed storage and processing results and further comparisons. The following SMILES-specific transforms were tested for all the compression metrics:

- 2-Gram_Chars to 11-Gram_Chars – N-grams using unused characters.
- 3-Gram_Brackets to 11-Gram_Brackets – N-grams reusing existing characters.
- Prefix_Chars – Number prefix addition using unused characters.
- Prefix_Equals – Number prefix addition reusing existing characters.
- Periodic_Chars – Atomic symbol to number conversion using unused characters.
- Periodic_Stars – Atomic symbol to number conversion using unused characters.
- Periodic_Brackets – Atomic symbol to number conversion reusing existing characters.
- BestStorage – This transform provides the best transformed file size and uses 2-gram_chars, prefix_chars and periodic_chars in its transform.

- AvgStorage – This transform provides an average transformed file size and uses 8-gram_chars, prefix_chars and periodic_stars in its transform.
- WorstStorage – This transform provides the worst transformed file size and uses 10-gram_brackets, prefix_equals and periodic_brackets in its transform.

Compression metrics were also computed for the following to compare our SMILES-specific transforms against:

- WRT-BWT [12] – WRT transform optimized for BWT compression.
- WRT-LZ77 [12] – WRT transform optimized for LZ77 compression.
- WRT-PAQ [12] – WRT transform optimized for PAQ compression.
- WRT-PPM [12] – WRT transform optimized for PPM compression.
- Untransformed – Untransformed data.

### 7.5 Results

Figures 2 to 10 illustrate the results and the key findings have been highlighted below:

- SMILES Transform Scenarios – The periodic_brackets transform scenario provided the best compression ratios for all compression algorithms except for PPMVC, where prefix_equals provided the best compression ratios for this algorithm. 2-gram_chars provided the worst compression ratios for all compression algorithms except for PPMVC, where 4-gram_chars gave the worst compression ratios for this algorithm. 2-gram_chars provided the best compression times for 7Zip, GZip, PPMVC, ZPAQ, along with 3-gram_chars for BZip2, and 8-gram_brackets for PPMd. Periodic_brackets gave the worst compression times for all compression algorithms. 6-gram_chars provided the best decompression times for 7Zip, together with 2-gram_chars for BZip2, PPMd, PPMVC and 4-gram_chars for GZip and ZPAQ. Periodic_brackets provided the worst decompression times for all compression algorithms except for GZip, where prefix_chars gave the worst times.
- Overall SMILES Transform Scenarios – For all transform scenarios 7Zip provided the best compression ratios overall. PPMVC and ZPAQ gave good compression ratios. GZip gave the worst compression ratios. BZip2 and PPMd also provided worse compression ratios. PPMd provided the best compression times. BZip2, GZip, PPMVC and ZPAQ gave good compression times. 7Zip provided the worst compression times overall. GZip provided the best decompression times overall. 7Zip and ZPAQ gave good decompression times. BZip2 and PPMd both provided worse decompression times. PPMVC provided the worst decompression times.
- Transform Comparisons – The WorstStorage SMILES transform provided the best compression ratios compared to all the other transforms tested and untransformed data when used with 7Zip. The WRT-LZ77 transform provided the worst compression ratios when used with GZip. The WRT-PAQ transform gave the best compression times when used with PPMd. The transform with the worst compression times was the WorstStorage SMILES transform when used with 7Zip. The untransformed files gave the best decompression times with GZip. The WorstStorage SMILES transform gave the worst decompression times with PPMVC.

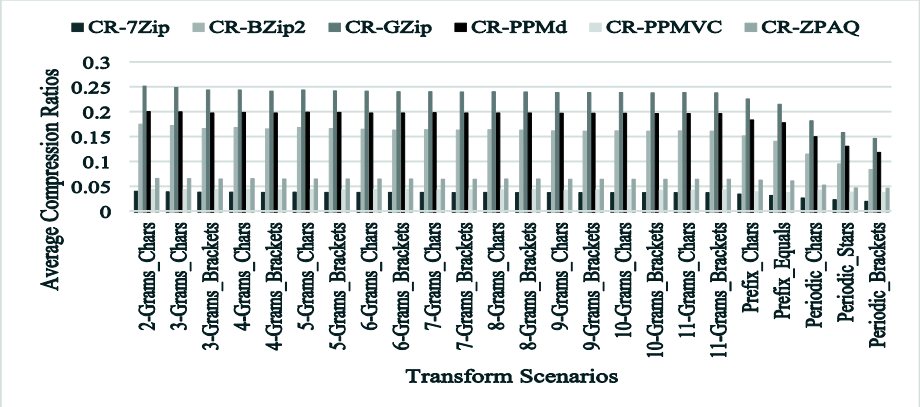


**Fig. 2.** Average Compression Ratios per SMILES Transform Scenarios

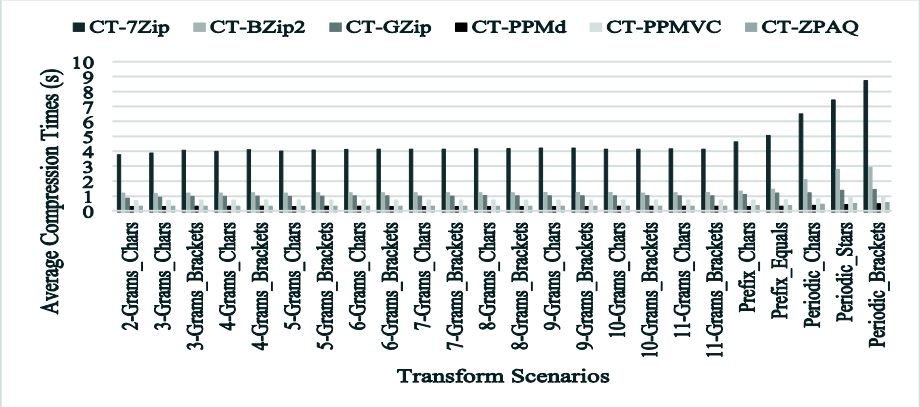


**Fig. 3.** Average Compression Times per SMILES Transform Scenarios

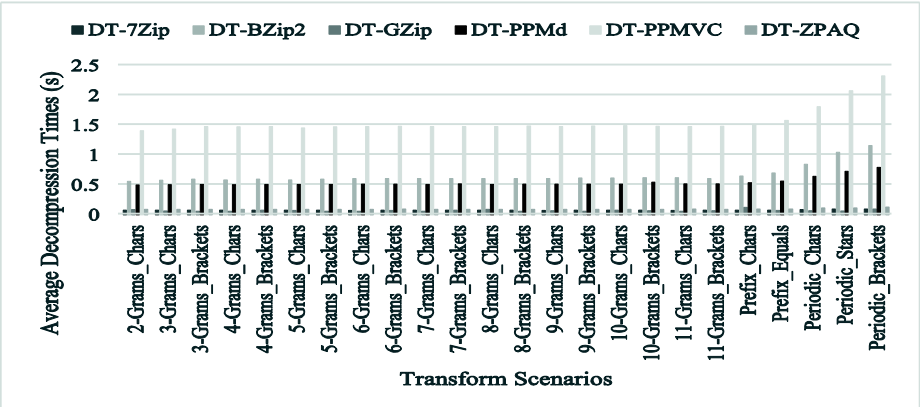


**Fig. 4.** Average Decompression Times per SMILES Transform Scenarios

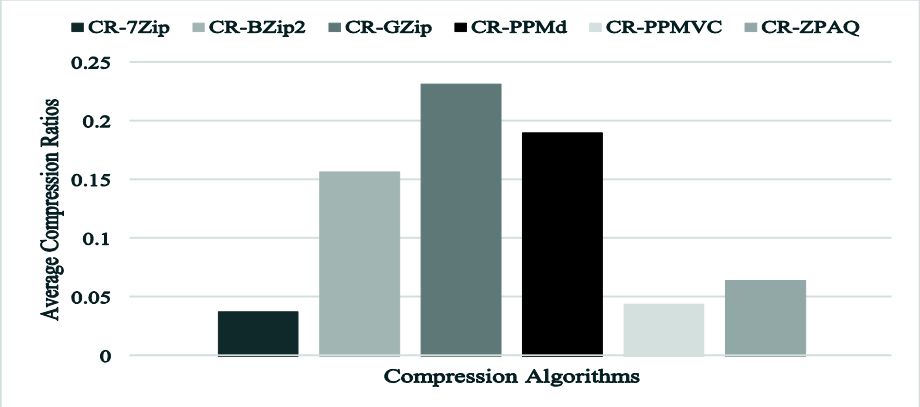


**Fig. 5.** Overall Average Compression Ratios for all Transform Scenarios per Algorithm

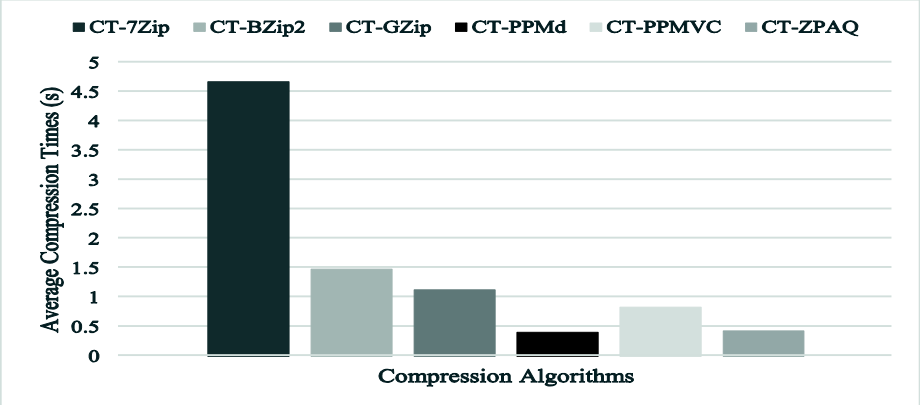


**Fig. 6.** Overall Average Compression Times for all Transform Scenarios per Algorithm

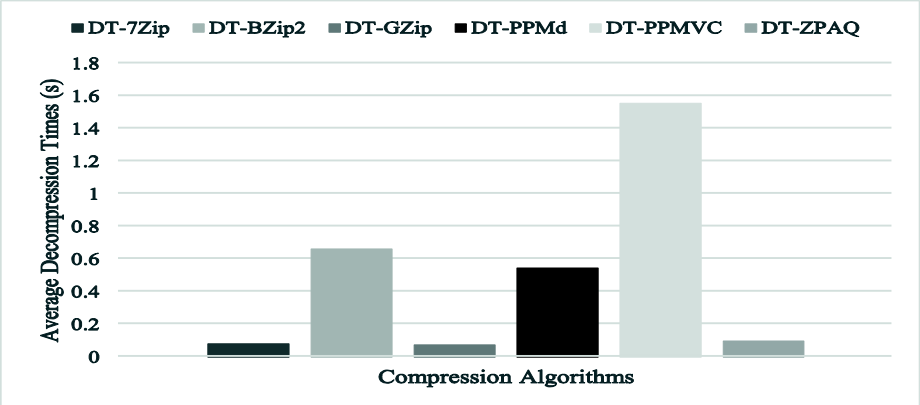


**Fig. 7.** Overall Average Decompression Times for all Transform Scenarios per Algorithm

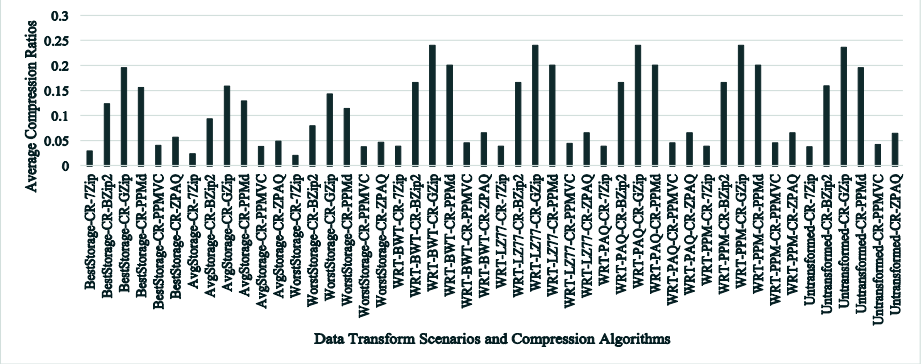


**Fig. 8.** Average Compression Ratios Comparing SMILES Transforms with WRT Transforms and Untransformed Data per Algorithm

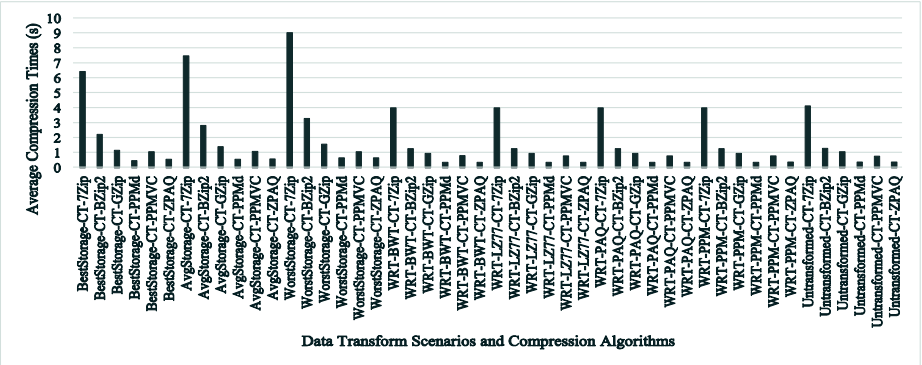


**Fig. 9.** Average Compression Times Comparing SMILES Transforms with WRT Transforms and Untransformed Data per Algorithm

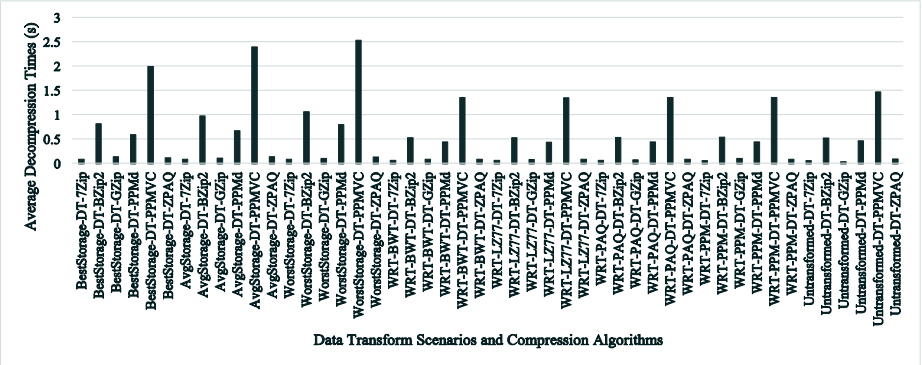


**Fig. 10.** Average Decompression Times Comparing SMILES Transforms with WRT Transforms and Untransformed Data per Algorithm

## 8 Discussions and Conclusions

In this paper we presented a SMILES-specific transform designed to enhance the compression of SMILES data when used with other general-purpose compression techniques. We tested general-purpose compression techniques on a breakdown of different SMILES transform scenarios and a combination of SMILES transforms. We compared our results with WRT transforms optimized for different compression algorithms and also on untransformed data.

The following can be concluded from the results:

- In terms of the SMILES transform scenarios tested, the results demonstrated that generally the number prefix and atomic element to atomic number transforms provided better compression ratios than n-gram substitution. However, n-gram substitution provided better compression times than number prefix and atomic element to atomic number transforms, except for PPMd. Decompression times were also generally better than the number prefix and atomic element to atomic number transforms, except for GZip and PPMd.
- It can also be noted that overall substituting data with existing characters in the data provided slightly better compression ratios than using unused characters, except for PPMVC. Whilst, many compression and decompression times for n-gram substitutions overall were better when using unused characters, the results also showed that using existing characters in some cases was better. Although, for number prefix and atomic number transforms using unused characters generally provided slightly better compression times than using existing characters. Using unused characters also provided slightly better decompression times mainly for number prefix and atomic number transforms, however, some also displayed slightly better results when using existing characters.
- The results also illustrate that generally ZPAQ provided good and balanced compression ratios, compression times and decompression times for all SMILES transform scenarios tested.
- Comparing the SMILES transforms developed with WRT optimized transforms and untransformed data, it can be concluded that all the SMILES transforms tested provided better compression ratios compared to all the WRT optimized transforms and untransformed data. Also, the WorstStorage SMILES transform provided better compression ratios than the AvgStorage and BestStorage SMILES transforms. However, all the WRT optimized transforms and untransformed data provided better compression and decompression times compared to the SMILES transforms.
- Overall, 7Zip and PPMVC provided the best compression ratios for all transforms and untransformed data. GZip gave the worst compression ratios overall. PPMd and ZPAQ provided the best compression times overall and 7Zip gave the worst compression times. 7Zip, GZip and ZPAQ provided the best decompression times, whilst PPMVC provided the worst decompression times overall.

In general, it can be concluded that transforms developed for specific data can enhance compression when used with other compressors. Transforming files prior to

compression can result in larger file sizes, however, as the results have shown, compression can still be enhanced despite this. We must also not forget that transforming files incurs additional pre-processing and post-processing times prior to compression and after decompression in order to return the file back to its original state. However, in our transform it is possible to process or use the files to a certain extent without de-transforming them. This can be achieved simply because the atomic element number to atomic symbol mapping, and vice-versa, is based on the information available in the periodic table. Users can refer to the periodic table for references to the atomic number in question and process these files without de-transformation.

A final point, it is important to note when designing transform or compression techniques that providing better storage impacts compression and decompression times, as can be seen with our SMILES transforms, and vice-versa, providing better compression and decompression times can impact compression storage. It is better to try and balance storage with performance if possible.

Future work will involve further research into data transformations and compression. The transform techniques developed will be further improved, and these experiments will be extended to provide further comparisons with compression techniques and more datasets.

**Acknowledgements.** We would like to thank all the authors of publicly available datasets, compression and transformation tools that made this study possible. We are also very grateful to the anonymous reviewers for their valuable comments and suggestions that helped to improve this paper.

## 9 References

1. 7z Format, http://www.7-zip.org/7z.html
2. Benchmark Data Set for In Silico Prediction of Ames Mutagenicity, http://doc.ml.tu-berlin.de/toxbenchmark/
3. BZip2 for Windows, http://gnuwin32.sourceforge.net/packages/bzip2.htm
4. Carus, A., Mesut, A.: Fast Text Compression Using Multiple Static Dictionaries. J. Inf. Tech. 9, 5, 1013-1021 (2010)
5. Cavnar, W.B., Trenkle, J.M.: N-Gram-Based Text Categorization. In: Proceedings of the Annual Symposium on Document Analysis and Information Retrieval (SDAIR '94), 11-13 Apr 1994, pp. 161-175, Las Vegas, Nevada, USA (1994)
6. Chemoinformatics.org, http://cheminformatics.org/datasets/index.shtml
7. Compression.ru Project, http://www.compression.ru/ds/ (in Russian)
8. Daylight Theory Manual, http://www.daylight.com/dayhtml/doc/theory/index.html
9. DSSTox, http://www.epa.gov/ncct/dsstox/
10. Engel, T.: Basic Overview of Chemoinformatics. J. Chem. Inf. Model. 46, 6, 2267-2277 (2006)
11. The GZip Home Page, http://www.gzip.org/
12. Homepage of Przemysław Skibiński, http://pskibinski.pl/
13. Index of /Data/CPDB, http://www.predictive-toxicology.org/data/cpdb/
14. Karthikeyan, M., Bender, A.: Encoding and Decoding Graphical Chemical Structures as Two-Dimensional (PDF417) Barcodes. J. Chem. Inf. Model. 45, 3, 572-580 (2005)

15. Kristensen, T.G., Nielsen, J., Pedersen, C.N.S.: Using Inverted Indices for Accelerating LINGO Calculations. J. Chem. Inf. Model. 51, 3, 597–600 (2011)
16. Kruse, H., Mukherjee, A.: Preprocessing Text to Improve Compression Ratios. In: Proceedings of the IEEE Data Compression Conference (DCC '98), 30 Mar-1 Apr 1998, pp. 556, Snowbird, Utah, USA (1998)
17. Mahoney, M.V.: Adaptive Weighing of Context Models for Lossless Data Compression. Technical Report CS-2005-16, Florida Institute of Technology, Melbourne, Florida, USA (2005)
18. O'Boyle, N.M.: Towards a Universal SMILES Representation – A Standard Method to Generate Canonical SMILES Based on the InChI. J. Cheminform. 4, 22 (2012)
19. Ratanaworabhan, P., Ke, J., Burtscher, M.: Fast Lossless Compression of Scientific Floating-Point Data. Proceedings of the IEEE Data Compression Conference (DCC '06), 28-30 Mar 2006, pp. 133-142, Snowbird, Utah, USA (2006)
20. Skibiński, P.: Reversible Data Transforms that Improve Effectiveness of Universal Lossless Data Compression. PhD Dissertation, University of Wrocław, Wrocław, Poland (2006)
21. Skibiński, P.: Two-Level Directory Based Compression. In: Proceedings of the IEEE Data Compression Conference (DCC '05), 29-31 Mar 2005, pp. 481-492, Snowbird, Utah, USA (2005)
22. Skibiński, P., Grabowski, S.: Variable-Length Contexts for PPM. Proceedings of the IEEE Data Compression Conference (DCC '04), 23-25 Mar 2004, pp. 409-418, Snowbird, Utah, USA (2004)
23. Weininger, D.: SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. J. Chem. Inf. Comput. Sci. 28, 1, 31-36 (1988)
24. ZPAQ: Open Standard Programmable Data Compression, http://mattmahoney.net/dc/zpaq.html