Solving the 10-square Problem in English Language Using Efficient Pruning Strategies

Matevž Kovačič*

Abstract

Word squares are linguistic puzzles consisting of words arranged in a square grid that can be read both horizontally and vertically. The construction of a perfect 10-word square in English has been an open problem in recreational linguistics for over a century, as the search space is vast and subject to strict constraints. Previous attempts have relied on the use of proper names, capitalised words, hyphenated words, or tautonyms to find partial solutions to the problem. In this paper, we present a novel algorithm that uses advanced pruning techniques to significantly reduce the search space of the problem and overcome the previous limitations. Using publicly available English word datasets, our method successfully constructs the first proper 10-square in English, demonstrating the potential of algorithmic approaches to complex linguistic challenges.

1 Introduction

Word squares are a fascinating and ancient form of acrostic puzzle in which a series of words are arranged in a square grid so that they can be read both horizontally and vertically. The number of words, which is equal to the number of letters in each word, determines the order of the square. An example of 3-square with words from a standard English dictionary [1]

```
P S I
S A T
I T S
```

The history of word squares goes back to antiquity, with the first-century Sator square from the remains of Pompeii [2] being one of the earliest and best-known examples. This Latin 5-square is not only palindromic, but also contains additional

*Inetis Ltd, Kidričeva ulica 25, SI-3000, Celje, Slovenia

hidden meanings and symbols that have sparked scholarly debate about its origin and purpose.

In modern times, word squares have been constructed in a variety of languages and sizes, with English-language squares extending to order nine. The search for a perfect 10-square in English has been going on since 1897, when the first 9-square was published [3]. Several partial solutions for the English 10-square have been proposed using lists of personal and geographical names [3].

Constructing a 10-square word puzzle is very challenging, mainly because of the need for a large vocabulary [4]. Brute force methods are not practical for solving such puzzles because the number of candidate solutions is $\binom{n}{10}$, where n is the size of the vocabulary. This leads to an exponential increase in the number of possible word combinations and makes the task computationally intractable. Even if basic pruning methods are used to reduce the search space, the sheer volume of possible solutions remains overwhelming and presents a significant obstacle for those trying to solve 10-square word puzzles.

In this paper, we make several contributions to the field of word puzzle construction. First, we introduce advanced pruning strategies to effectively manage the state space of square word puzzles. By adding single characters to partial solutions rather than whole words, we efficiently prune search space and significantly reduce the number of tests required compared to previous approaches. Second, the construction of our solution focuses on the below-the-diagonal part of the word matrix. Here, the transposed elements are automatically set to the appropriate values, exploiting the symmetry of the word square solutions. Finally, we have compiled a comprehensive dataset of about 360K English words of length 10, taken from various publicly available sources. This extensive vocabulary offers the possibility of constructing 10-squares.

Remarkably, our developed algorithm successfully constructed the first perfect 10-word square in the English language without using proper nouns, capitalised words, hyphenated words, or tautonyms, solving a problem that has existed since 1897.

2 Methodology and Implementation

2.1 Custom Dictionary Construction

To create large word squares, an extensive dictionary is required. The estimated size of a vocabulary needed to create a single 10-word square is about 247,718 words of length 10 [4] (or 256,945 if the assumption of independence of characters from their position within the word is omitted).

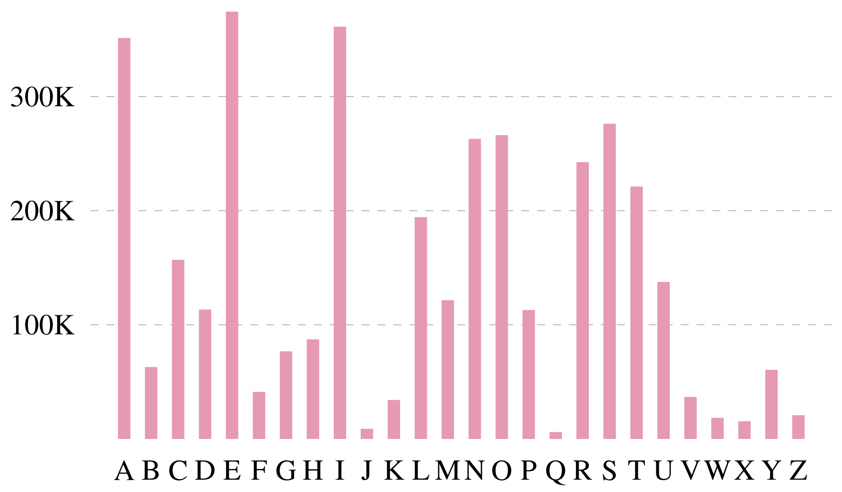

Figure 1: Frequency distribution of characters in English words of length 10.

In practise [3] the actual size of the dictionary tends to be 38% larger than the estimated value. Taking this factor into account, approximately 342,000 words are required to construct a 10-square. The requirements are beyond the capabilities of a single dictionary or data source.

In order to gather enough words for this purpose, we have gathered words from the following sources:

1. English Wiktionary [5]
2. One Billion Word Language Modeling Benchmark Dataset [6]
3. JStor [7]
4. Catalogue of life [8]
5. SemMedDb [9]

Consolidating the data sources and removing words beginning with a capital letter, hyphenated words and words containing non-English characters resulted in a dictionary of about 360,000 words of length 10. In Fig 1 we show the frequency distribution of the characters in the dictionary.

Applying Long's estimation formula [4] with the frequencies of the characters in our dictionary gives an estimate of the size of the dictionary needed for the 10-square of 222,406. Taking into account the 38% increase factor, the size of the vocabulary needed for the 10-square is about 307,000. It is important to note that the consolidated dictionary data is noisy and contains misspelled words and words from other languages. Since manual cleaning of the dictionary was not feasible, we chose to manually filter out misspelled words only if they appear in the solutions of our word square construction algorithm. The generated dictionary, especially if we take into account that it also contains non-valid words, is just about the estimated size needed to generate one 10-word square.

3 Algorithm and Problem Definition

In this section we present an efficient algorithm for solving the word-square problem and give a clear definition of the problem itself.

The word square problem is a combinatorial puzzle in which the goal is to find all the word squares from a given vocabulary V containing N words of size n. Words arranged in a square grid can be read both horizontally and vertically. A word square can be represented as $n \times n$ symmetric matrix $W = (w_{ij})$ of characters, where the characters in each row and column form valid words from the vocabulary.

We develop a backtracking search algorithm that effectively navigates through the space of partial solutions using advanced pruning rules to minimise the exploration of partial solutions that cannot lead to a valid solution. The algorithm includes the following main steps:

Initialisation: Start constructing the solution in the upper left corner of the word matrix.

Traversal path: Iterate through the cells of the matrix using a specific traversal path that starts at the first column of each row, moves diagonally upwards until it reaches the main diagonal element, and then continues with the next row. In the last row, the algorithm enumerates all remaining diagonals and ends in the bottom right cell of the matrix (as shown in Fig 2). This is the only traversal path that ensures that for each new cell to be examined, both the immediate left and the upper neighbouring cell are already defined, which is crucial for the application of pruning rules (see Sect 3.1).

Candidate selection: For each cell w_{ij}, select valid candidate characters based on the pruning rules (described in detail in Sect 3.1).

Backtracking: If no valid candidate characters can be placed in a cell, backtrack to the previous cell on the matrix traversal path or stop if the current cell is the top left matrix cell.

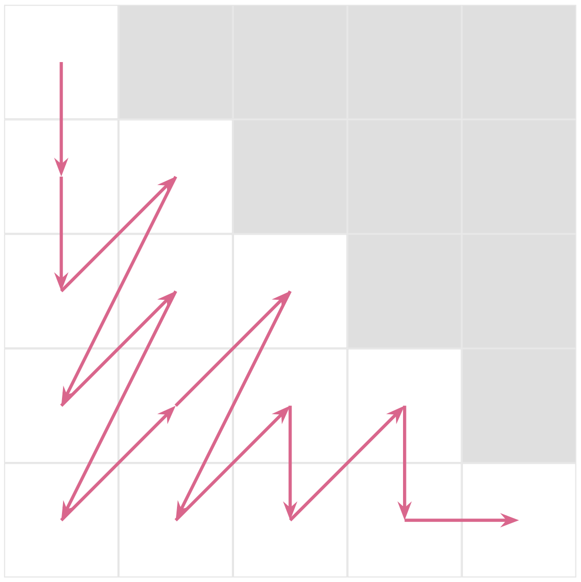

Figure 2: Traversal path of the matrix algorithm ensuring that the immediate left and upper neighbouring cells are defined for the application of the pruning rules.

> *Validating the transposed cell:* When you have found a valid character C for a cell w_{ij}, also check whether the transposed cell w_{ji} can accommodate C by applying the same pruning rules.
>
> *Progression:* If the partial solution with $w_{ij} = w_{ji} = C$ is considered valid, proceed to the next cell on the matrix traversal path.

The algorithm generates all valid word squares by effectively enumerating all valid solutions and efficiently discarding partial solutions that cannot lead to a valid solution. In the worst case, the algorithm faces the challenge of enumerating an exponential number of possible partial solutions for word squares. However, the pruning heuristic has proven effective in practise, especially for large vocabularies, as it significantly reduces the search for invalid solutions. The algorithm's ability to competently navigate through the space of partial solutions, apply advanced pruning rules and exploit the symmetric property of the word matrix makes it a valuable approach to solving the word-square problem.

3.1 Pruning Strategies within the Algorithm

In this section, we introduce the pruning mechanisms that our algorithm uses to solve the word square problem. These mechanisms are important to improve the performance of the algorithm by minimising the exploration of partial word squares that cannot lead to valid solutions of the problem.

The algorithm uses both near and long-range expansion constraints to effectively prune the search space. The short-range expansion constraints are applied to ensure that a candidate character C can be placed at a specific position within the word matrix, taking into account the symmetric nature of the matrix and vocabulary V. There are four short-range expansion constraints, with constraints 3 and 4 being the symmetric counterparts of constraints 1 and 2. All of the following conditions are shown in Fig. 3:

1. Horizontal expansion in row i: V must contain a word with the prefix $w_{i0}w_{i1}\dots w_{i(j-1)}C$.
2. Vertical expansion in column j: V must contain a word with the prefix $w_{0j}w_{1j}\dots w_{(i-1)j}C$.
3. Horizontal expansion in row j: V must contain a word with the prefix $w_{j0}w_{j1}\dots w_{j(i-1)}C$ (symmetrical counterpart to condition 1).
4. Vertical expansion in column i: V must contain a word with the prefix $w_{0i}w_{1i}\dots w_{(j-1)i}C$ (symmetrical counterpart to condition 2).

In addition to the short-range constraints, a long-range expansion constraint is applied if $i > j$. In such a case the intersection element of w_{ij} and its transpose w_{ji}, namely w_{ii}, must contain a matching character X so that V contains words:

1. $w_{i0}\dots w_{i,(j-1)}Cw^{j-i-1}X$
2. $w_{0i}\dots w_{(j-1)i}Cw^{j-i-1}X$

where w^n denotes any sequence of characters of length n. The long-range constraint is shown in Fig 4.

Our algorithm expands partial word-square solutions in the traversal order illustrated in Fig 2 to effectively apply pruning mechanisms. For the traversal step for w_{ij} the pruning tests are performed in the following order:

1. w_{ij} is set to a character C that satisfies the valid short-range expansion constraints in the horizontal and vertical directions for w_{ij}.
2. w_{ji} is set to C and checked to see if the short-range expansion constraints in the horizontal and vertical directions are satisfied for w_{ji}.

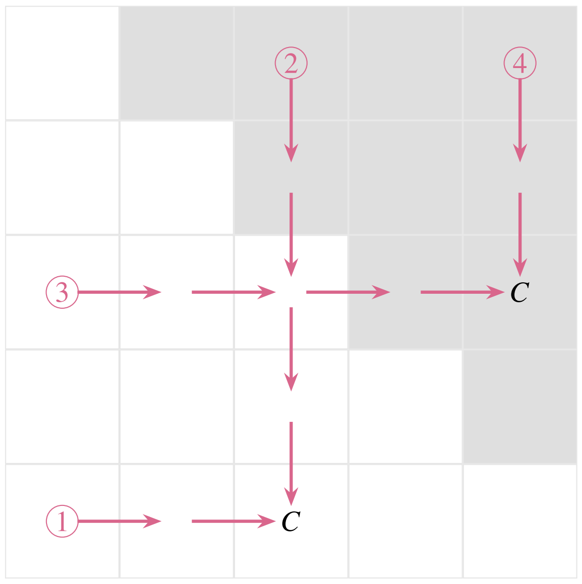

Figure 3: Types of short-range expansion constraints for candidate character C. Note that the matrix traversal path of the algorithm (see Fig 2) ensures that all constraints can be evaluated.

3. If $i > j$, the algorithm evaluates all rows within the interval $(j, i]$ for compliance with the long-range expansion constraints with respect to column j. In case of a violation of the constraints, the algorithm returns to the previous stage of the traversal path using backtracking. It is important to note that although analogous long-range expansion constraints are possible for columns relative to the current row, these were implemented but later removed from the implementation of the algorithm. This decision is based on the observation that the inclusion of these constraints does not help to increase the efficiency of the algorithm.

By testing the long- and short-range expansion constraints immediately after adding each character to the partial word square, our algorithm ensures maximum pruning of the search space. This approach is more efficient and effective than the common practise of adding whole rows or columns to the partially defined word square solution, which increases the number of tests performed and prevents the symmetry constraints from being tested for all characters in the added word. This is because not all transposed elements of the word matrix are defined after a whole row or column has been added to the partial word square solution. Therefore, our traversal is more efficient and effective for creating word squares.

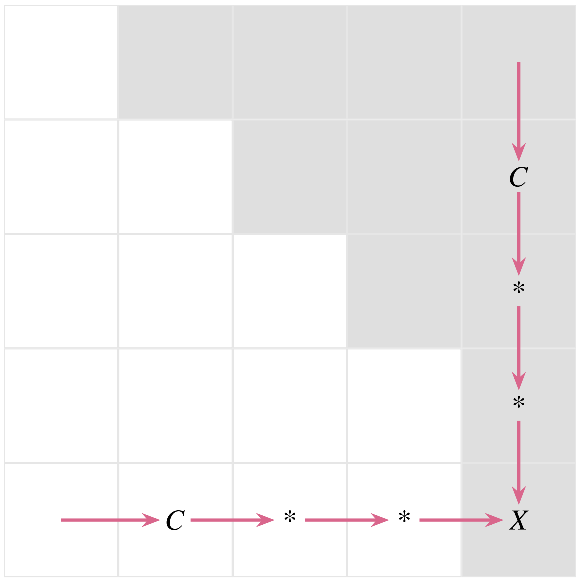

Figure 4: Long-range expansion constraints. The existence of X ensures that C may be placed in a word square. The $*$ cells represent arbitrary characters, currently undefined, which must be taken into account when evaluating the long-range constraints.

3.2 Efficient data structures for pruning the search space

The effectiveness and running time of algorithms for constructing word squares are significantly influenced by the use of efficient data structures for pruning the search space. The size and pattern of words in the dictionary also affect the ability to find large word squares.

To test the conditions for expansion over short- and long-range expansion, we use trie or an array of trie sets. A trie allows constant-time access to the set of next possible characters of a word prefix, so that the expansion conditions for the short range can be tested in both horizontal and vertical directions in constant time. Similarly, an array of d trie sets allows constant-time access to the set of next nearest possible characters based on a word prefix, followed by $d < n-1$ arbitrary characters, ensuring constant-time testing for a single long-range expansion constraint. Note that n long range expansion constraints must be performed, resulting in $O(n)$ long range test complexity. Note that the amortised time for the construction of dictionary trie and d trie sets must also be considered.

In summary, our backtracking algorithm constructs word squares by incrementally creating partial solutions while efficiently pruning the search space using diagonal traversal and various pruning mechanisms. By incorporating appropriate data structures, the algorithm achieves a balance between computational complexity and effectiveness in constructing valid word squares. This approach enables the construction of word squares even in situations where brute force methods or methods that add whole rows or columns to partial solutions would not be feasible.

The linear complexity of testing long-range interactions together with the constant-time testing of short-range expansion constraints for each matrix cell helps that the algorithm remains efficient and practical for different dictionary sizes and word-square dimensions.

4 Results and Discussion

We applied our algorithm for constructing word squares to a noisy dictionary of about 360K English words of length 10 obtained from the sources in Sect. 2.1. This dictionary contains misspelled words, proper nouns and non-English words. Since manual cleaning is not feasible, we filtered out misspelled words only if they appeared in the algorithm's solutions.

In parallel, we ran four versions of the algorithm, each with a fixed character in the upper left corner of the matrix, until all the letters of the English alphabet were exhausted. After the algorithm generated solutions, we manually checked them for validity and immediately removed invalid words from the dictionary. The processing time for enumerating all possible solutions for a single fixed letter in the top left corner ranged from a few hours to over a week on a computer with an Intel(R) i5-9400F CPU @ 2.90GHz processor.

The algorithm run with the "S" in the upper left corner took 34 hours, evaluating over 800 billion partial solutions. The resulting perfect square you see below excludes proper nouns, capitalised words and hyphenated words:

S	C	A	P	H	A	R	C	A	E
C	E	R	R	A	T	E	A	N	A
A	R	G	O	L	E	T	I	E	R
P	R	O	C	O	L	I	C	I	N
H	A	L	O	B	O	R	A	T	E
A	T	E	L	O	M	E	R	E	S
R	E	T	I	R	E	M	E	N	T
C	A	I	C	A	R	E	N	S	E
A	N	E	I	T	E	N	S	I	S
E	A	R	N	E	S	T	E	S	T

Word	**Description & source**
scapharcae	*adj.* epithet of bacterium *Ornithinibacillus scapharcae* [10]
cerrateana	*adj.* epithet of a plant *Pitcairnia cerrateana* [8]
argoletier	*noun (obsolete, military)* a light mounted soldier; a mounted bowman [11]
procolicin	*noun* a propeptide form of colicin [12]
haloborate	*noun* a type of inorganic compound [13]
atelomeres	*adj.* epithet of a species of moths *Ectropis atelomeres* [8]
retirement	*noun* withdrawal from one's position or occupation or from active working life [14]
caicarense	*adj.* epithet of a plant *Machaerium caicarense* [8]
aneitensis	*adj.* epithet of a tree fern *Alsophila aneitensis* [8]
earnestest	*adj. (now rare)* superlative form of earnest [15]

The solution contains five species epithets, one type of inorganic compound, one type of organic compound, one rare word, one obsolete word and one common English word, with the newest word introduced in 2011.

This is the first correct solution to the 10-word square problem, which was posed in 1897. Although other 10-word squares might exist or future English words might allow for new solutions, an 11-word square is unlikely. We have unsuccessfully attempted to construct an 11-word square from the sources in Sect 2.1, which supports the claim that the estimated size of the dictionary required for an 11-word square is in the order of one million words [4].

5 Data and Code Availability

The implementation of the algorithm for constructing word squares and the dictionary used to construct 10-square from this paper can be found at `https://github.com/matevz-kovacic/word-square`.

References

[1] Merriam-Webster online dictionary. Accessed on 2023-05-08.

[2] Matteo Della Corte. I cristiani a pompeii. *Rendiconti Accademia di Archeologia, Lettere e belle arti di Napoli*, 12:394–400, 1936.

[3] A Ross Eckler. *A History of the Ten-Square*, pages 85–91. A K Peters, Ltd, 2005.

[4] Chris Long. Mathematics of square construction. *Word Ways*, 26(1), 1993.

[5] Tatu Ylonen. Wiktextract: Wiktionary as machine-readable structured data. In *Proceedings of the 13th Conference on Language Resources and Evaluation (LREC)*, pages 1317–1325, Marseille, June 20-25 2022. Retrieved on February 28, 2023.

[6] One billion word language modeling benchmark dataset, 2023. Downloaded on March 14, 2023, from `https://www.statmt.org/lm-benchmark/1-billion-word-language-modeling-benchmark-r13output.tar.gz`.

[7] Jstor database, 2023. Downloaded using Constellate builder on March 7, 2023, from `https://constellate.org/builder`.

[8] O. Bánki, Y. Roskov, M. Döring, G. Ower, L. Vandepitte, D. Hobern, D. Remsen, P. Schalk, R. E. DeWalt, M. Keping, J. Miller, T. Orrell, R. Aalbu, J. Abbott, R. Adlard, E. M. Adriaenssens, C. Aedo, E. Aescht, N. Akkari, et al. Catalogue of life checklist, March 2023. Retrieved from `https://www.catalogueoflife.org/data/download`, Retrieval date 26 March 2023.

[9] SemMedDB, 2023. Retrieved on January 10, 2023, from `https://lhncbc.nlm.nih.gov/ii/tools/SemRep_SemMedDB_SKR/SemMedDB_download.html`.

[10] NR Shin, TW Whon, MS Kim, SW Roh, MJ Jung, YO Kim, and JW Bae. Ornithinibacillus scapharcae sp. nov., isolated from a dead ark clam. *Antonie van Leeuwenhoek*, 101(1):147–154, Jan 2012. Epub 2011 Sep 28.

[11] argoletier. `https://en.wiktionary.org/wiki/argoletier`, 2022. Accessed: 2023-05-06.

[12] N Horn, A Fernández, HM Dodd, MJ Gasson, and JM Rodríguez. Nisin-controlled production of pediocin pa-1 and colicin v in nisin- and non-nisin-producing lactococcus lactis strains. *Applied and Environmental Microbiology*, 70(8):5030–5032, Aug 2004.

[13] Ismail Y Ahmed and C D Schmulbach. Conductance of some haloborate and boronium salts in acetonitrile at 25.deg. *Inorganic Chemistry*, 8(7):1411–1413, Jul 1969.

[14] retirement. `https://www.merriam-webster.com/dictionary/retirement`, 2023. Accessed: 2023-05-06.

[15] earnestest. `https://en.wiktionary.org/wiki/earnestest`, 2022. Accessed: 2023-05-06.