Journal of
open research software

A Python Package for Well-Separated Pair Decomposition

SOFTWARE METAPAPER

DOMAGOJ MATIJEVIĆ

]u[ubiquity press

ABSTRACT

Well-separated pair decomposition (WSPD) is a well-known geometric decomposition used for encoding distances, introduced in 1995 by Paul B. Callahan and S. Rao Kosaraju in a seminal paper. We implemented this remarkable decomposition following the nontrivial algorithm for computing the partial fair split tree of a point set presented in the original article. Our implementation is done in C++. In addition to that, we made further effort to publish it as a Python package on PyPI. By doing so, we made our software easily accessible on Windows, Linux, or macOS to researchers and students worldwide.

CORRESPONDING AUTHOR:
Domagoj Matijević
School of Applied Mathematics and Computer Science, University of Osijek, Croatia
domagoj@mathos.hr

KEYWORDS:
Well-separated pair decomposition; computational geometry

TO CITE THIS ARTICLE:
Matijević D 2024 A Python Package for Well-Separated Pair Decomposition. *Journal of Open Research Software,* 12: 1. DOI: https://doi.org/10.5334/jors.465

(1) OVERVIEW

INTRODUCTION

Many different problems in geometry involve Euclidean distances defined by pairs of n points in a given point set. For example, the closest pair problem, the all-nearest-neighbors problem, the problem of determining k smallest distances among $\binom{n}{2}$ pairwise distances, etc. The solution to such problems often relies on some 'clever' search over a quadratic space of pairwise distances. The natural question is whether one can efficiently approximate the space of pairwise distances in subquadratic space and, by doing so, help provide an efficient solution to many of the problems mentioned above. The answer to that question was given in a seminal work by Paul B. Callahan and S. Rao Kosaraju in 1995 ([1]), which showed that the quadratic space of all pairwise distances could be encoded in $O(n)$ space and $O(n \log n)$ construction time, for a fixed dimension d. The decomposition they introduced is well-separated pair decomposition (WSPD), a well-known decomposition in computational geometry.

Many exact and approximate algorithms for different proximity problems are based on a WSPD: the diameter problem, the spanner problem, the minimum spanning tree, etc. (for the complete list of problems, see [2]). An extensive overview of the a WSPD and its applications is given in the book [3]. A WSPD has even been listed among problems in the Encyclopedia of Algorithms ([4]), which presents solutions to the most important algorithmic problems. Surprisingly, no efficient implementation of a WSPD can be found in public software libraries available for students and researchers.

This paper presents the software package that implements a WSPD following the nontrivial algorithm of the partial fair-split tree presented in [1] that guarantees the construction time of $O(n \log n)$ and a WSPD of $O(n)$ size. To our knowledge, our implementation is the first open-source and publicly available implementation of WSPD that carefully follows the original algorithm in [1]. Recently published implementation of a WSPD in ParGeo C++ library (see [5]) uses simple fair-split kdtree and does not ensure theoretic bounds on the size of a WSPD.

Moreover, in order to make it convenient for researchers and students and available for everyday use, we created Python bindings to our C++ code and published it to PyPI. Python module can be run on Windows, macOS, and Linux for all Python versions starting from 3.7.

Formal problem statement

For the sake of the completeness, we provide the formal definition of a WSPD. Let S be a set of n points in $\mathbb{R}^d$. A pair of sets $A, B \subset S$ are said to be well-separated if

$$d(A,B) > s \cdot \max\{\text{diam}(A), \text{diam}(B)\},$$

for any given separation constant $s > 0$, where $d(A, B) = \min_{a \in A, b \in B} d(a, b)$ and diam(·) denotes the diameter of a set of points.

A well-separated pair decomposition of $S \subset \mathbb{R}^d$, for a given $s > 0$, is a sequence $(A_1, B_1), \dots, (A_k, B_k)$, where $A_i, B_i \subseteq S$, such that

1. A_i, B_i are well-separated with respect to the separation constant s, for all $i = 1, \dots, k$;
2. for all $p \neq q \in S$ there exists a unique pair (A_i, B_i) such that $p \in A_i, q \in B_i$ or $q \in A_i, p \in B_i$.

Note that a WSPD always exists since one could use all singleton pairs $(\{p\}, \{q\})$ for all pairs $p, q \in S$. However, this would yield a sequence of well-separated pairs of size $k = \Theta(n^2)$. The work of [1] showed that, given a set S of n points in $\mathbb{R}^d$ and a separation constant $s > 0$, a WSPD of S with $O(s^d d^{d/2} n)$ many well-separated pairs can be computed in $O(dn \log n + s^d d^{d/2} n)$ time.

IMPLEMENTATION AND ARCHITECTURE

The implementation of a WSPD was done in C++ with the help of STL containers, and no other external libraries were used. To facilitate the practical use of our software, we used the pybind11 library ([6]) to enable Python binding to our C++ implementation. The main advantage of pybind11 is that it allows for automatic conversion between STL containers and Python list, set, and dict data structures. More precisely, we use native Python types `<int, int, float, list>` in Python function `wspd.build_wspd()`, and pybind11 converts them to native C++ types `<int, int, double, stl::vector>` in the corresponding C++ function call. In the opposite direction, the well-separated pairs of a WSPD, once computed in C++, are represented as a composite data type

```
<stl::vector<stl::pair<stl::vector<int>,
stl::vector<int>>>>
```

and pybind11 converts them to Python's `list<tupe<list, list>>`. Type conversions involve copying data during Python/C++ transitions due to memory layout differences. We direct the interested reader to pybind documentation (see [6]) for more details about type conversions.

Furthermore, we used the GitHub Actions CI/CD [7] to compile our project for Windows, Linux, and macOS and released it as a Python package on the Python Package Index PyPI [8] platform.

QUALITY CONTROL

Tests for the WSPD package are performed using different-sized random data and manually constructed inputs where the correct results are known. In Figure 1, left, one can see that our implementation outputs well-separated pairs typically in less than a few seconds

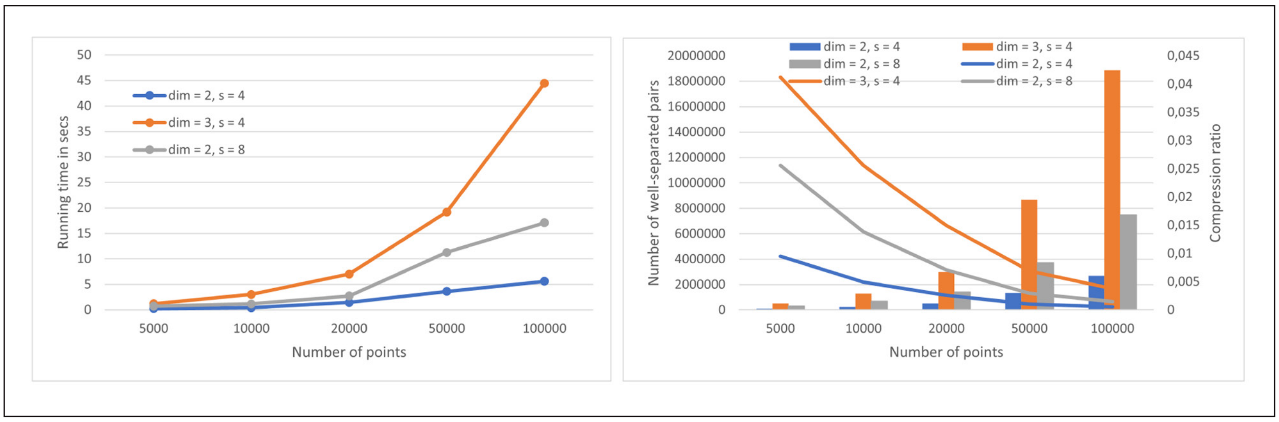

Figure 1 (left) Running times with respect to the number of points is shown. (right) The relation between the total number of well-separated pairs and the number of points is depicted with boxes. The compression ratio, i.e., the total number of wellseparated pairs divided by the total number of pairwise distances $n \cdot (n-1)/2$, is depicted with lines.

for inputs as large as 20 000 points in the plane. Unfortunately, the dependence in the running time on the dimension is exponential, which is also reflected in our experiments. In Figure 1, right, we revealed the total number of well-separated pairs as a function of the input size. Surprisingly, the compression ratio, i.e., the total number of well-separated pairs of sets divided by the total number of pairwise distances $n \cdot (n-1)/2$, decreases in practice as the size of the input set of points increases. Thus, the larger the space of all pairwise distances, the better it gets compressed with the WSPD algorithm.

(2) AVAILABILITY

OPERATING SYSTEM

The package is platform-independent and it can be run on Linux, macOS and Windows.

PROGRAMMING LANGUAGE

Python versions 3.7 and larger. The software can also be included in other C++ projects.

ADDITIONAL SYSTEM REQUIREMENTS

None

DEPENDENCIES

WSPD is implemented in C++ and uses only standard STL containers and C++ native types. No dependencies on the Python side are required. However, for visualizing a WSPD, you might want to use matplotlib, and for point set generation, you might want to rely on numpy random sampling.

SOFTWARE LOCATION

Archive

Name: Zenodo
Persistent identifier: 10.5281/zenodo.8314954
Licence: Creative Commons Attribution 4.0 International
Publisher: Domagoj Matijević
Version published: Version 1
Date published: 04/09/2023

Code repository

Name: GitHub
Persistent identifier: https://github.com/dmatijev/wspd pip
Licence: MIT
Version published: v0.7.0
Date published: 04/09/2023

LANGUAGE

English

(3) REUSE POTENTIAL

Users can request support by raising an issue on GitHub. We expect our implementation, presented as a C++ code and a Python module installable from the PyPI repository, can be easily included in any existing C++ or Python projects. This project will hopefully bring WSPD, a well-known geometric decomposition, closer to researchers and students by letting them easily use WSPD in their Python and C++ projects.

USE CASE FOR THE SOFTWARE

Figure 2 shows a WSPD computed on a simple example.[1] Though the example is elementary and comprises only eight different points, it already demonstrates the effectiveness of pairwise distance compression, e.g., note that all possible pairwise distances among any number of points placed in different red rectangles are already all efficiently approximated by pairwise distances between red rectangles.

For Python users, installing the WSPD module now amounts to a simple `pip install wspd` command. In the following, we provide a simple example of how the WSPD package can be used, following the example depicted in Figure 2.

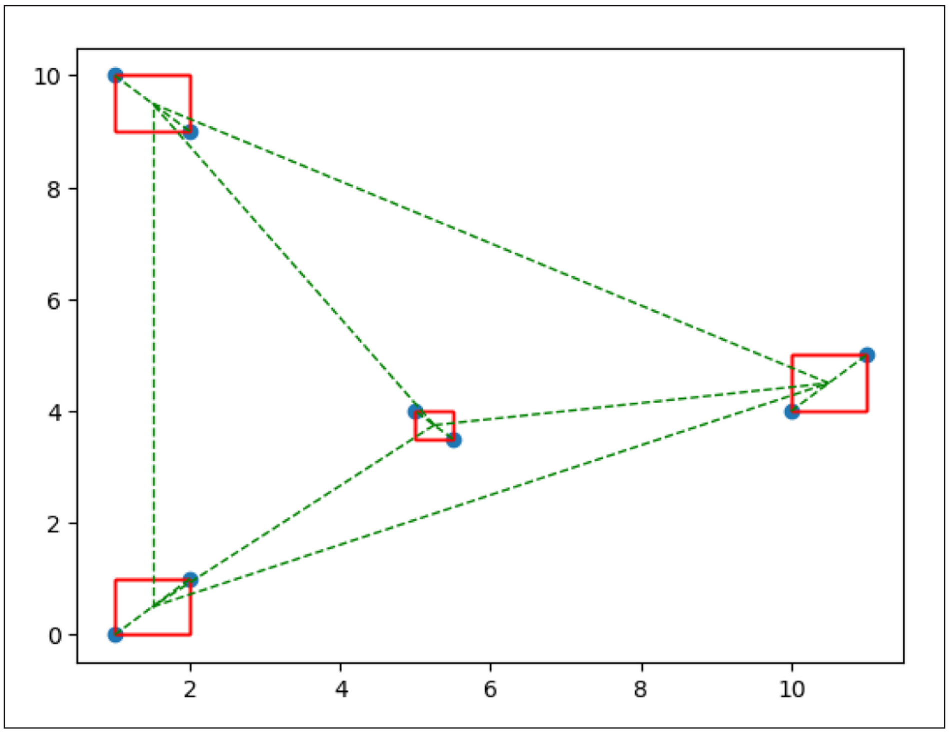

Figure 2 A simple example of a WSPD for eight blue points in the plane computed with separation constant $s = 2$. The WSPD algorithm computed ten well-separated pairs, denoted as green dashed lines. Thus, all pairwise distances between points in different red rectangles will be approximated with corresponding green dashed lines.

```
import wspd
data_pts = [(1,0),(2,1),(1,10),(2,9),(10,4
),(11,5),(5,4),(5.5,3.5)]
nr_pts = len(data_pts) # number of points
dim = len(data_pts[0]) # point dimension

# move points from Python list to point
class objects
data_pts = [wspd.point(p) for p in data_
pts]
# compute WSPD and return well-separated
pairs
dumbbells = wspd.build_wspd(nr_pts, dim,
S, data_pts)
```

As already stated, a well-separated pair decomposition has an application in solving a number of problems. For example, given the well-separated sequence (A_1, B_1)… (A_k, B_k) computed by the `build_wspd()` function, the closest pair problem can be computed in an additional $O(n)$ time by a simple inspection of all pairs (A_j ,B_j), $1 \leq j \leq k$, where both A_j and B_j are singleton sets. A similar approach can be used to compute the k closest pairs problem or for many approximate proximity problems (for the complete list of proximity problems, a WSPD has the direct consequence to, see [2]). Finally, it is worth mentioning that a WSPD has been successfully used in different geometric problems, e.g., the computation of energy-efficient paths in radio networks [9]. A well-separated pairs were used for the precomputation of 'template' paths that were further used in the query phase of their algorithm.

In conclusion, a well-separated pair decomposition is a powerful technique that has proven useful in various applications in geometry. We hope that this open-source project could also lead to further collaboration, as other researchers may contribute improvements and new features to this Python package over time.

NOTE

1. This example is also provided on GitHub under */example* and can be used for further testing of a WSPD.

ACKNOWLEDGEMENTS

We thank Tomislav Prusina, who provided insight and expertise that greatly assisted the development of this software.

COMPETING INTERESTS

The author has no competing interests to declare.

AUTHOR AFFILIATIONS

Domagoj Matijević orcid.org/0000-0003-3390-9467
School of Applied Mathematics and Computer Science, University of Osijek, Croatia

REFERENCES

1. **Callahan PB, Kosaraju SR.** A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, jan 1995; 42(1): 67–90. ISSN 0004-5411. DOI: https://doi.org/10.1145/200836.200853
2. **Smid MHM.** The well-separated pair decomposition and its applications. In Gonzalez TF (ed.), *Handbook of Approximation Algorithms and Metaheuristics, Second Edition, Volume 2: Contemporary and Emerging Applications*. Chapman and Hall/CRC; 2018.
3. **Narasimhan G, Smid MHM.** *Geometric spanner networks.* Cambridge University Press; 2007. DOI: https://doi.org/10.1017/CBO9780511546884
4. **Klein R.** *Well Separated Pair Decomposition*. New York, NY: Springer New York, 2016; 2368–2371. ISBN 978-1-4939-2864-4. DOI: https://doi.org/10.1007/978-1-4939-2864-4_479
5. **Wang Y, Yu S, Dhulipala L, Gu Y, Shun J.** Pargeo: A library for parallel computational geometry. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming,* PPoPP '22. New York, NY, USA, 2022; 450–452. Association for Computing Machinery. ISBN 9781450392044. DOI: https://doi.org/10.1145/3503221.3508429
6. **Jakob W, Rhinelander J, Moldovan D.** pybind11 – seamless operability between c++11 and python; 2017. https://github.com/pybind/pybind11.
7. Github actions documentation. URL: https://docs.github.com/en/actions.
8. Python package index pypi. URL: https://pypi.org/.
9. **Beier R, Funke S, Matijević D, Sanders P.** Energy-efficient paths in radio networks. *Algorithmica*, oct 2011; 61(2): 298–319. ISSN 0178-4617. DOI: https://doi.org/10.1007/s00453-010-9414-0

TO CITE THIS ARTICLE:
Matijević D 2024 A Python Package for Well-Separated Pair Decomposition. *Journal of Open Research Software,* 12: 1. DOI: https://doi.org/10.5334/jors.465

Submitted: 30 March 2023 **Accepted:** 20 December 2023 **Published:** 02 January 2024