Software Impacts 17 (2023) 100558

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

STNWeb: A new visualization tool for analyzing optimization algorithms

Camilo Chacón Sartori [a], Christian Blum [a,*], Gabriela Ochoa [b]

[a] Artificial Intelligence Research Institute (IIIA-CSIC), Campus of the UAB, Bellaterra, Spain
[b] University of Stirling, Stirling, UK

ARTICLE INFO

Keywords:
Algorithm analysis
Visualization
Behavior of optimization algorithms
Web application

ABSTRACT

STNWeb is a new web tool for the visualization of the behavior of optimization algorithms such as metaheuristics. It allows for the graphical analysis of multiple runs of multiple algorithms on the same problem instance and, in this way, it facilitates the understanding of algorithm behavior. It may help, for example, in identifying the reasons for a rather low algorithm performance. This, in turn, can help the algorithm designer to change the algorithm in order to improve its performance. STNWeb is designed to be user-friendly. Moreover, it is offered for free to the research community.

Code metadata

Current code version	*v1.1*
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2023-306
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/4324580/tree/v1
Legal Code License	*GNU General Public Licence (GPLv3)*
Code versioning system used	*Git/GitHub*
Software code languages, tools, and services used	*Backend: Python, R - Frontend: TypeScript*
Compilation requirements, operating environments & dependencies	Docker
If available Link to developer documentation/manual	https://github.com/camilochs/stnweb/tutorial
Support email for questions	cchacon@iiia.csic.es

1. Introduction

By visualizing information, we turn it into a landscape that you can explore with your eyes, a sort of information map. And when youre lost in information, an information map is kind of useful.

[David McCandless]

Visual representations of complex concepts aid us in comprehending digital information more effectively. This holds for many areas of Computer Science and Artificial Intelligence. However, the research community on combinatorial optimization has so far not been very productive in what concerns the development of visual tools, even though there is an increasing need to assist in the comparison of optimization algorithms with new tools. During the last decades, the standard for comparing optimization algorithms was based on collecting numerical data of optimization runs of different algorithms and comparing these by means of tables and classical data charts (e.g., line plots, bar plots, and scatter plots). In addition, it has become a standard to complement this kind of algorithm comparison through a statistical analysis of the data. During the last years, however, an increasing number of researchers have realized that, for obtaining a proper understanding of the behavior of optimization algorithms such as metaheuristics [1,2], there is a need for incorporating additional graphical tools. Moreover, graphical tools are not only required but must also be user-friendly.

Even though, as mentioned above, the research community on optimization algorithms has not been very productive concerning the development of visual tools, the visualization of optimization algorithm behavior has been attempted a few times, including [3–6]. These methods utilize dimensionality reduction to map search spaces to two or three dimensions in order to enable a rudimentary tracking of the

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.

E-mail addresses: cchacon@iiia.csic.es (C. Chacón Sartori), christian.blum@iiia.csic.es (C. Blum), gabriela.ochoa@stir.ac.uk (G. Ochoa).

https://doi.org/10.1016/j.simpa.2023.100558
Received 7 July 2023; Received in revised form 26 July 2023; Accepted 26 July 2023

search progress. However, the currently best tool for this purpose was introduced only recently in [7,8]. This tool – labeled *Search Trajectory Networks (STNs)* – utilizes directed graph objects with nodes and arcs for visualization and assists in analyzing the search progress. It provides a way to display multiple trajectories of multiple optimization algorithms applied to the same problem instance in a graphical way. This approach relies on graph-based visualization and comes with several R scripts provided by the authors of [8] which are available for free at https://github.com/gabro8a/STNs. The initial implementation of the STNs tool, in the form of R scripts, presented various challenges to potential users. The most notable hindrance was the complexity of use, as the generation of the final graphics requires multiple manual steps, including the execution of a range of R scripts. In order to overcome this issue, but also for the enhancement of the tool with additional features, we created STNWeb [9], a web application that streamlines the previously mentioned process. STNWeb can be executed locally by downloading it as a Docker from Code Ocean (see code metadata), or it can be executed online under the following permanent URL:

https://www.stn-analytics.com/

A simple example of an STN graphic comparing two different algorithms applied to a problem instance of the well-known *multi-dimensional knapsack problem* is shown in Fig. 1. In particular, this graphic displays the trajectories of 10 different runs of each of the two algorithms to the problem instance. Vertices in this graphic correspond to solutions to the problem instance. However, this is not necessarily the case, as explained below. The meaning of colors, vertex shapes, and vertex sizes is as follows:

- The trajectories of the runs of different algorithms are shown in different colors, as indicated in the legend of each STN graphic. The 10 trajectories of a *CMSA* algorithm, for example, are shown in Fig. 1 in blue, while the ones of an *LNS* algorithm are painted in green.
- Starting points of trajectories are indicated by means of yellow squares. Note that, in Fig. 1, the 10 runs of CMSA all start from different initial solutions, while the 10 runs of LNS all start from the same initial solution.
- Trajectory endpoints are either shown as dark-grey triangles or as red dots. The former is the case if the respective endpoint does not correspond to a best-found solution (concerning all algorithm runs), while the latter (red dot) is the case if the respective endpoint corresponds to a best-found solution.
- Light-grey dots indicate solutions forming part of the trajectories of at least two different algorithms.
- Finally, the size of a vertex/dot indicates the number of algorithm trajectories passing through it: the larger the vertex size, the higher the number of algorithm trajectories.

The STN graphic in Fig. 1 compares two different algorithms applied 10 times to the same problem instance. It shows that there clearly exists an area of attraction in the search space, especially for the CMSA algorithm. Five of the 10 CMSA trajectories are attracted to the area of the large light-grey dots. However, even though many trajectories pass through these two solutions, they are clearly not the best solutions of that area, because none of the trajectories actually stops at one of the two solutions. The LNS algorithm is less attracted to this area of the search space as only one of its trajectories passes through it. This type of information is basically neglected in optimization research nowadays. Nevertheless, it can be very useful for understanding, for example, why an algorithm works especially well, or why an algorithm does not work well at all.

2. STNWeb architecture

The architecture of STNWeb consists of three components: the frontend, the backend, and the search space partitioning strategy. Thanks to its user-friendly interface, the frontend allows users to quickly enter

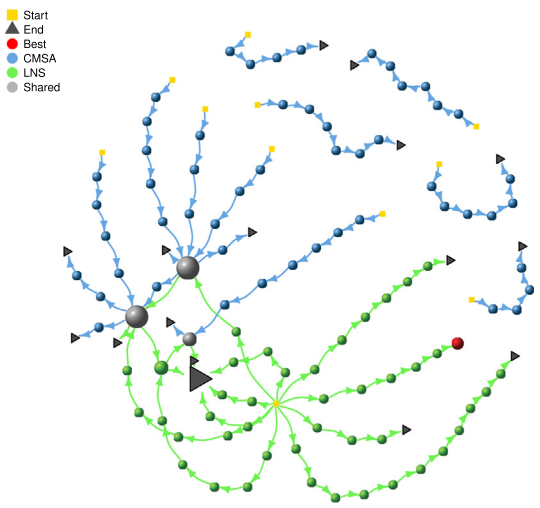


Fig. 1. Example of an STN graphic. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

all the necessary information about the algorithms to be analyzed. The backend invokes the partition strategy component through a Rest API to generate visualizations. The search space partitioning strategy component contains specialized algorithms that can partition the search space into locations that potentially include many solutions in order to uncover information that is not otherwise evident. Fig. 2 shows how these three components interact. Next, we provide a description of each component.

2.1. *STNWeb frontend*

We used Angular[1] and Typescript[2] to build the web application and Bootstrap[3] for styling. The web is designed to be a single-page app. The web app allows users to select the type of problem (discrete or continuous optimization problem), as well as the search space partitioning algorithm and its configuration. Users can also customize the resulting graphics by adjusting the color and size of the nodes. Once the settings are selected, users must upload the data files containing the trajectory data for each algorithm. After completing the configuration form, a user must click the Generate button to produce the visualization. To learn how to properly format and structure the data files containing the trajectory data of the algorithms, please consult the STNWeb tutorial (see https://github.com/camilochs/stnweb).

2.2. *STNWeb backend*

The backend was developed using both Python[4] and R.[5] Python was used to create the Rest API using Flask,[6] a micro web framework, and the partitioning strategy component, while R was used to generate visualizations in PDF format. Upon receipt of the user's configuration and the algorithm data files for comparison, the backend verifies the correct format of each file and applies the selected partitioning algorithm. A unified file is generated and subsequently passed on to the R scripts to create the visualization graphs.

1 https://angular.io/
2 https://www.typescriptlang.org/
3 https://getbootstrap.com/
4 https://www.python.org/
5 https://www.r-project.org/
6 https://flask.palletsprojects.com

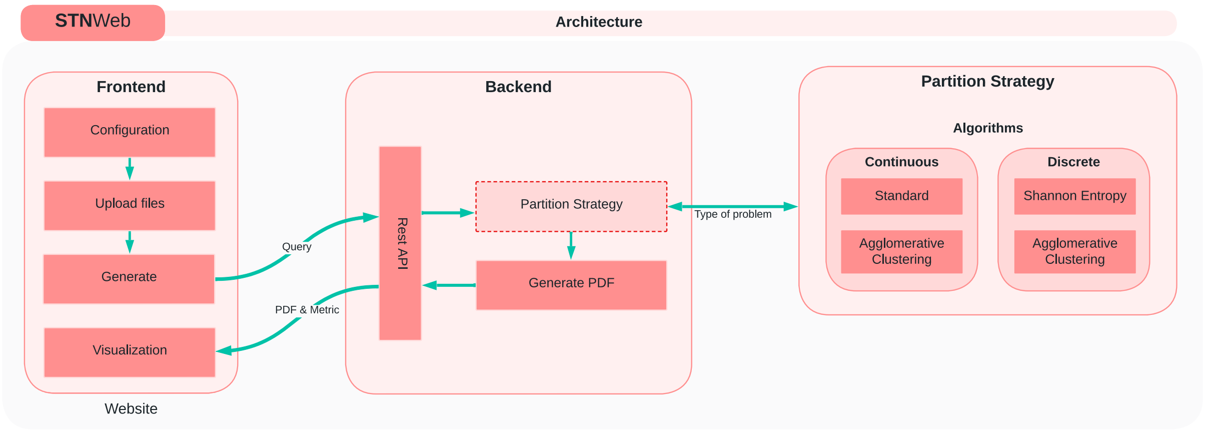
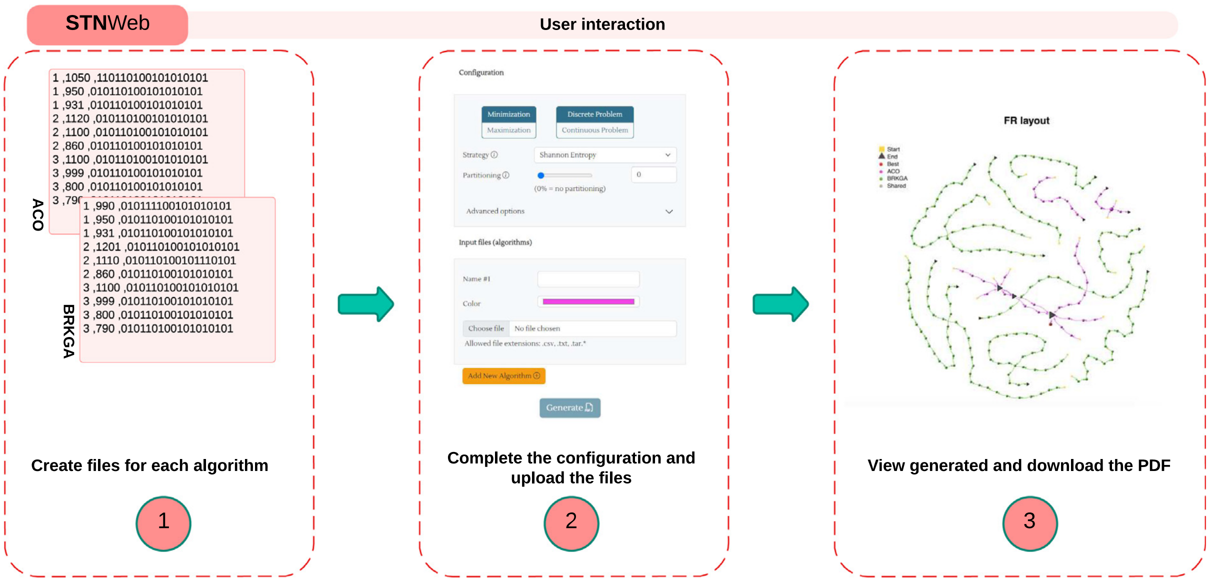


Fig. 2. Once completed filling out the configuration form and uploading the algorithm data files (frontend), a user may request the generation of visualizations. The Rest API (backend) will receive the request and invokes the chosen algorithm to partition the space and to create a PDF containing the visualization based on the user's initial configuration. Finally, the generated PDF will be displayed in the embedded viewer.



```
1 ,1050 ,1101101001010101010
1 ,950 ,0101101001010101010
1 ,931 ,0101101001010101010
2 ,1120 ,0101101001010101010
2 ,1100 ,0101101001010101010
2 ,860 ,0101101001010101010
3 ,1100 ,0101101001010101010
3 ,999 ,0101101001010101010
3 ,800 ,0101101001010101010
3 ,79
```

```
1 ,990 ,0101111001010101010
1 ,950 ,0101101001010101010
1 ,931 ,0101101001010101010
2 ,1201 ,0101101001010101010
2 ,1110 ,0101101001011101010
2 ,860 ,0101101001010101010
3 ,1100 ,0101101001010101010
3 ,999 ,0101101001010101010
3 ,800 ,0101101001010101010
3 ,790 ,0101101001010101010
```

Strategy: Shannon Entropy
Partitioning: 0 (0% = no partitioning)
Advanced options
Name #1:
Color:
Choose file: No file chosen
Allowed file extensions: .csv, .txt, .tar.*

Fig. 3. Interacting with STNWeb only requires three simple steps. Here is an example. First, separate data files containing the trajectory data of the considered algorithms must be generated. In the second step, the configuration of the algorithm comparison must be completed on STNWeb, and the data files must be uploaded. Finally, in the third step, a visualization is generated and can be downloaded in PDF format.

2.3. Rest API

The Rest API has the following two entry points:

- `/stn` - The user sends configuration information and files to be analyzed via a POST request. Thus, this entry point returns a PDF containing the visualization.
- `/metric` - Once the `/stn` request is submitted, the set of metrics (supplied by means of a spreadsheet file) can be accessed through the provided entry point for download.

Utilizing a Rest API is crucial for simplifying frontend modifications, as this is one of its primary advantages. Therefore, the entry points will remain unchanged, regardless of any changes made to the user interface which is crafted to provide a seamless and straightforward user experience (see Fig. 3).

2.4. Search space partitioning strategy

Sometimes – either due to characteristics of the problem instance or due to algorithm characteristics – search trajectories are very long. This inevitably leads to very cluttered STN visualizations, making the algorithm analysis basically impossible. To address this issue, we provide various methods for partitioning (or dividing) the search space into junks or locations that contain more than one solution. More details about this issue can be found in [8]. By using these partitioning techniques, the STN graphics are focused on their essential structure which shows the algorithm behavior. However, search space partitioning is not always necessary and, therefore, it is not of obligatory use in STNWeb. In fact, the default setting works without search space partitioning.

It is also worth mentioning that search space partitioning depends on the type of optimization problem that is solved. In particular, STNWeb offers partitioning schemes that are limited to a certain type of optimization problem (such as the partitioning scheme based on *Shannon Entropy*, which is limited to discrete optimization problems). However, it offers also one partitioning scheme based on *Agglomerative Clustering* which can be applied both to discrete and continuous optimization problems.

3. Impact overview

STNWeb potentially has an important impact on the community of designers of iterative, stochastic optimization algorithms. This is because it is the first tool that assists – in a visual way – in the comparison of multiple runs of multiple algorithms. Moreover, STNWeb is user-friendly and free of use to everybody. In particular, the tool often helps to understand why a certain algorithm design works especially well or, on the contrary, may not work well at all. A good example of what can be learned from the STN graphics generated by our tool is shown in Fig. 1. This graphic compares 10 runs of each of two different algorithms (CMSA and LNS) for the same problem instance. The following can be observed:

- Even though LNS obtains the best solution found by all runs (red dot), there is no common area of attraction for the LNS trajectories. This generally indicates a low robustness of the algorithm.
- There are not many overlaps between the LNS trajectories, which also indicates low robustness.
- Five out of 10 CMSA runs are attracted by a specific area of the search space, as indicated by the large grey dots that show a high overlap between CMSA trajectories. This indicates that the robustness of CMSA is higher than the one of LNS.

As a result of this STN graphic, an algorithm designer obtains a clear indication that the robustness of LNS needs to be improved. Another information that can be obtained from STN graphics is, for example, if an algorithm gets stuck in certain local optima. This may trigger the algorithm designer to improve the diversification mechanism of the developed algorithm. Due to these benefits, after the initial publication of the STN methodology in 2021 [8], some researchers have already started using it for enhancing their algorithmic studies. Examples include [10–15]. In [13], for example, the author used STN graphics especially for studying the overlap between trajectories of different optimization algorithms. In contrast, in [12] the authors used STN graphics for explaining why their hybrid algorithm outperformed the standard algorithm variant on some problem instances, while it was outperformed by the standard algorithm on other problem instances. In a similar way, the author of [14] made use of STN graphics for being able to give a more detailed explanation of why some of his algorithm variants outperformed others on problem instances of a certain structure.

In not even two years, the original article on STN technology [8] has already obtained 31 citations (Google Scholar). STNWeb facilitates the use of the STN technology and we expect that, as a result, the body of users will grow significantly in the near future.

4. Conclusion and future work

In this paper, we have described our web application STNWeb which aims to improve optimization research analytics. This web application facilitates the process of using the original STNs tool from [8]. Moreover, it features additional functionalities such as a search space partitioning scheme that can be applied to both discrete and continuous optimization problems. Concerning future work, there are at least four possible ways to enhance our application: (1) Adding 3D technology would enhance the visualizations and would aid in identifying aspects of the algorithm comparison that may have been missed in the 2d visualization; (2) Enabling real-time graph modification by the user could improve the usefulness of the algorithm analysis; (3) We also believe that it is still useful to consider the development of new, additional search space partitioning schemes; and (4) we want to accelerate STNWeb for comparisons of more than 4 algorithms.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This paper was supported by grant TED2021-129319B-I00 funded by MCIN/AEI/10.13039/501100011033.

References

[1] Michel Gendreau, Jean-Yves Potvin, Metaheuristics in combinatorial optimization, Ann. Oper. Res. 140 (1) (2005) 189–213.
[2] Christian Blum, Andrea Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[3] Trevor D. Collins, Applying software visualization technology to support the use of evolutionary algorithms, J. Vis. Lang. Comput. 14 (2) (2003) 123–150.
[4] H. Pohlheim, Multidimensional scaling for evolutionary algorithms – visualization of the path through search space and solution space using Sammon mapping, Artif. Life 12 (2006) 203–209.
[5] Krzysztof Michalak, Low-dimensional Euclidean embedding for visualization of search spaces in combinatorial optimization, IEEE Trans. Evolut. Comput. 23 (2) (2019) 232–246.
[6] Andrea De Lorenzo, Eric Medvet, Tea Tušar, Alberto Bartoli, An analysis of dimensionality reduction techniques for visualizing evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM, 2019.
[7] Gabriela Ochoa, Katherine M. Malan, Christian Blum, Search trajectory networks of population-based algorithms in continuous spaces, in: Proceedings of EvoApps 2020 – International Conference on the Applications of Evolutionary Computation, Springer, 2020, pp. 70–85.
[8] Gabriela Ochoa, Katherine M. Malan, Christian Blum, Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics, Appl. Soft Comput. 109 (2021) 107492.
[9] Camilo Chacon-Sartori, Christian Blum, Gabriela Ochoa, Search trajectory networks meet the web: A web application for the visual comparison of optimization algorithms, in: Proceedings of the 2023 12th International Conference on Software and Computer Applications, ICSCA '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 89–96.
[10] Yuri Lavinas, Claus Aranha, Gabriela Ochoa, Search trajectories networks of multiobjective evolutionary algorithms, in: International Conference on the Applications of Evolutionary Computation (Part of EvoStar), Springer, 2022, pp. 223–238.
[11] Valentina Narvaez-Teran, Gabriela Ochoa, Eduardo Rodriguez-Tello, Search trajectory networks applied to the cyclic bandwidth sum problem, IEEE Access 9 (2021) 151266–151277.
[12] Camilo Chacón Sartori, Christian Blum, Boosting a genetic algorithm with graph neural networks for multi-hop influence maximization in social networks, in: 2022 17th Conference on Computer Science and Intelligence Systems, FedCSIS, IEEE, 2022, pp. 363–371.
[13] Boris Almonacid, AutoMH: Automatically create evolutionary metaheuristic algorithms using reinforcement learning, Entropy 24 (7) (2022) 957.
[14] Teddy Nurcahyadi, An Algorithmic Framework for Making Use of Negative Learning in Ant Colony Optimization (Ph.D. thesis), Universidad Autónoma de Barcelona (UAB), 2022.
[15] Cosmin Constantin Andru, Red de la trayectoria de búsqueda de las metaheurísticas, (Master's thesis), ETSI Informàtica, Universidad Politécnica de Madrid, 2022.