Search Trajectory Networks (STNs) Web Application

(version v1.1)

Tutorial

Camilo Chacón Sartori, Christian Blum

May 24, 2023

1 Introduction

The comparison of stochastic, iterative optimization algorithms is generally a difficult task. In fact, such comparisons are mostly based on numerical result tables and certain types of charts (bar, pie, etc.). Search trajectory networks (STNs) are a rather new tool in order to assist researchers in the task of comparing different algorithms. In fact, the new web tool STNWeb that makes use of STNs generates eye-catching visualizations to compare multiple optimization algorithms (see figure 1).

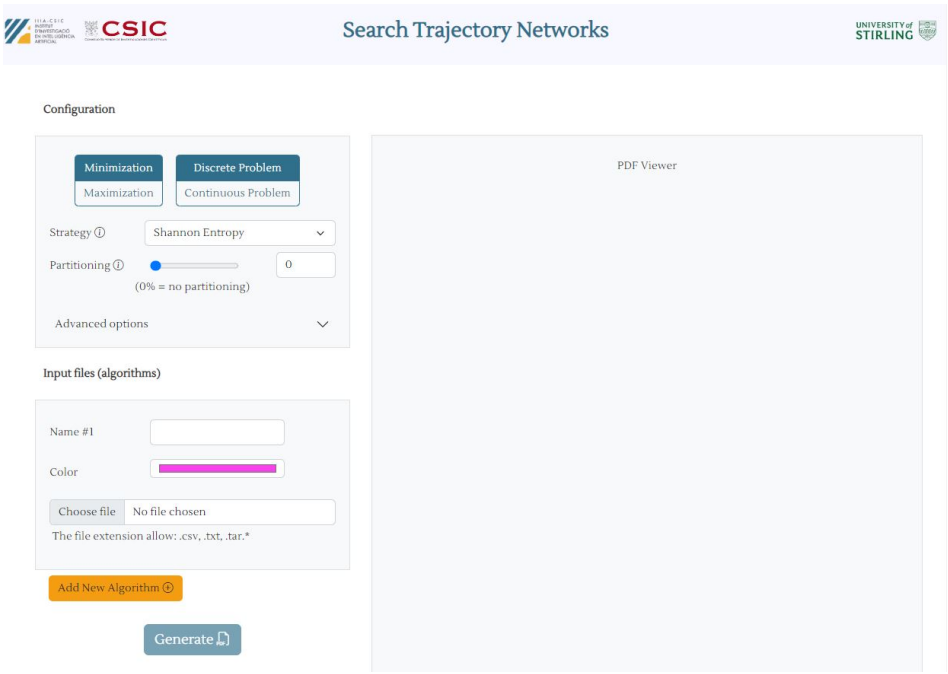


Figure 1: STNWeb main page.

2 Input files

The STNWeb currently can deal with the results of algorithms for two kinds of optimization problems: discrete and continuous problems. Both require slightly different input file formats, which are explained below. Please note that only files with extensions `.txt` and `.csv` are permitted. Moreover, the tool also allows for compressed versions (`.tar.gz`) of such files.[1]

[1] Compressed files are recommended for very large `.txt` and `.csv` files.

Before we start with the description of the input files, note that the visual comparisons of STNWeb are instance-based, that is, STNWeb allows to make a comparison of multiple runs of multiple algorithms for a single problem instance. The results of each algorithm (one or multiple runs) must be supplied by means of a separate input file.

2.1 Discrete optimization problem

Listing 1 shows an example of the content of an input file for a discrete problem. Note that the file has three columns which are separated by commas (tabulators are also allowed for separating columns). The three columns contain the following data:

1. The first column indicates the algorithm run. Therefore, the example input file shows three runs of an algorithm.
2. The second column contains a fitness value of the solution which is shown in string format in the third column. Note that fitness values within a run are decreasing, which means that this data stems from an algorithm applied to a minimization problem.
3. Finally, note that the last column accepts any string, as long as strings are of the same length. In the case of our example we have binary strings, which indicates that the tackled problem is a binary optimization problem. In general, such strings may be produced by ordering the variables of the problem in a fixed way and concatenating the values of these variables in the corresponding solutions.

```
1,1000,010110100101010101
1,950,01011010010101010 1
1,931,01011010010101010 1
2,1200,010110100101010101
2,1100,010110100101010101
2,860,01011010010101010 1
3,1100,010110100101010101
3,999,01011010010101010 1
3,800,01011010010101010 1
3,790,01011010010101010 1
```

Listing 1: Example of an input file for a discrete minimization problem.

2.2 Continuous optimization problem

There is a slight difference between the input file formats for discrete and continuous optimization problems. For an example of the continuous case, consider Listing 2.

```
1 12.079147503496035  0.17057415953587007,-0.8323403651415218,-1.0167401682782042
1 9.693960361302459 0.009483221295656818,-0.7882374197159615,-0.9476418994386451
1 9.406072694721054 -1.0825483830319764,0.8980604563424185,-1.9622506163707742
1 6.600536472899478 0.09908262786649168,-1.1029656373604055,-1.0443980531855368
1 2.9751997644545725  0.00532599459685712,-1.0584860516992491,-0.07794513807358727
1 2.7347794697938532  -0.003340960717349727,-1.0549782773622605,-0.07263934160449687
1 1.9067422971545618  0.0010138007451910465,-1.0220444934589945,-0.06254471780698309
1 1.5889410543503466  -0.9770198741653673,-0.0025276092799271233,0.05186790322347734
1 0.8018413644040443  -0.0393800880539995,-0.04961148969129788,0.007565797064684743
1 0.7661824816708176  -0.036124989204653346,-0.05024563715860908,0.007661079356715617
1 0.301226619792871 0.03446409957433601,0.018316182606329223,-0.0002493652525031309
1 0.25453762041128414 -0.01101845422374792,-0.011080490678693572,0.03228729141858
1 0.1108304393483337  -0.019520626399893062,-0.00633790500047915,0.011745730015247584
2 22.757999832788997  -0.829453733568293,-1.129413543675997,-3.1201914468754497
2 9.7310986569507 -0.03344920081024352,1.9519905898120509,-2.0703718645983087
2 8.92797335254214  1.018141237903115,-1.088751873653424,-0.1684573165674602
2 5.947124887377072 -0.03403730162219785,1.042938448520033,-2.0281033253548215
2 3.764063556082867 0.9732466617137494,-1.0832258481119486,-0.028979618340608138
2 2.888016128611035 -0.0033198253344832278,-0.0982746827848412,1.0081708267755451
```

Listing 2: Example of an input file for a continuous minimization problem of three dimension.

In particular, commas are used to separate the continuous values of the decision variables in the third column. Moreover, columns are now obligatorily separated by tabs. Also, note that no headers are required in both cases (discrete and continuous).

3 How to use STNWeb?

STNWeb allows for n inputs files, each providing the data of multiple algorithms runs for the same, single instance. Note that all files must either contain the same number of algorithm runs, or the number of algorithms runs to be used for the comparison must be indicated by means of setting the corresponding parameter on the main page of STNWeb. If two algorithms are to be compared in the context of a specific problem instance, two input files must be uploaded.

In addition to uploading the data files, a user must perform five rather simple steps to perform a visual comparison (regardless of the number of algorithms to be compared). These steps are described below.

3.1 Type of the considered problem

First, a user must choose the type of the considered optimization problem (minimization vs. maximization), and then select whether the problem is a discrete or a continuous problem. See Figure 2 below.

Figure 2: Choosing the type of the optimization problem considered

Depending on the type of the chosen problem (discrete vs. continuous) the further options will change by activating different search space partitioning strategies (see Section 3.2).

3.2 Search Space Partitioning Strategy

Sometimes, algorithm runs are very long. This generally results in the problem that the produced graphics for the visual comparison of the algorithms are too cluttered in order to be analyzed. In order to solve this problem, we offer different methods for the partitioning of the search space (more details in [OMB21]). Note that, if no search space partitioning is required (or desired) the user should simply leave the default setting (which implies no use of search space partitioning).

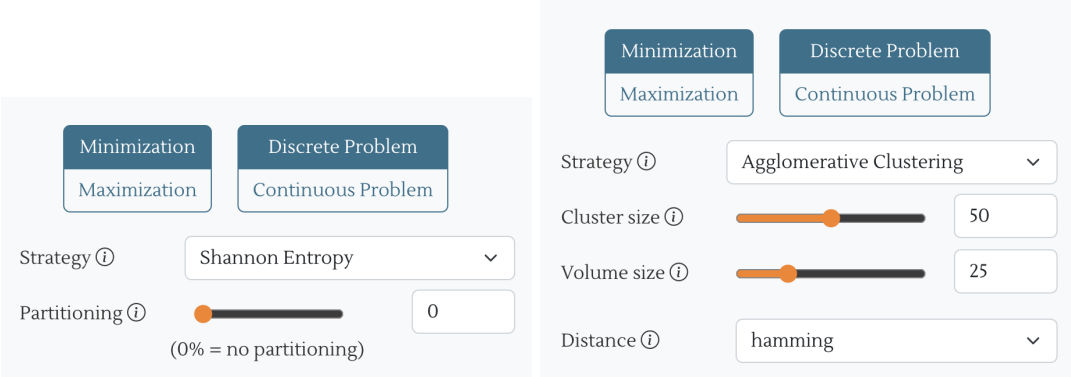
3.2.1 Discrete cases

For discrete cases, one can choose among two different search space partitioning strategies (see Figure 3):

- *Shannon Entropy.* The partitions are created according to the Shannon entropy values calculated for each decision variable. Variables are sorted according to these values and the less significant variables are discarded. Configuration parameters:

  1. *Partitioning.* The percentage of variables to be removed after ordering them (ascending) with respect to their Shannon Entropy value.

- *Agglomerative Clustering.* Here the partitions are created creating clusters based on the distance between points/solutions in the search space, cluster size and cluster volume. Configuration parameters:



(a) (b)

Figure 3: Search space partitioning strategies for discrete problems: (a) Shannon Entropy (b) Agglomerative Clustering.

1. *Cluster size.* Maximal cluster size in terms of the percentage of all solutions a cluster may maximally contain.
2. *Volume size.* Maximal cluster size in terms of the percentage of the covered search space volume spanned by the solutions of a cluster.
3. *Distance.* Metric for the calculation of solution, resp. cluster, distances. Available options for the discrete case are the Euclidean and the Manhattan distance.

3.2.2 Continuous cases

For continuous cases, the user can choose among the two following search space partitioning strategies (see Figure 4):

- *Standard Partitioning.* This form of partitioning separates the search space into hypercubes. Note that this form of search space partitioning is not indicated/practical for problems with more than five decision variables. Moreover, this form of partitioning requires a problem with box constraints with the same lower and upper bounds for each dimension. Configuration parameters:
  1. *Hypercube.* A factor that determines the size of the hypercubes. A lower value for the factor implies smaller hyper-cubes and, therefore, a lower degree of search space partitioning.
  2. *Minimum Bound.* The lower bound of the box constraints.
  3. *Maximum Bound.* The upper bound of the box constraints.
- *Agglomerative Clustering.* Similar to the discrete case, except for the available distance measures.

3.3 Advanced options

Advanced options are optional for modifying visual elements and other information (see Figure 5). They affect the appearance of the visual output of STNWeb. Advanced options are the following ones:

1. *Value best-known solution.* Value of the best-known solution for the analyzed problem instance (not necessarily to be found in the analyzed algorithm runs/trajectories).
2. *Number of runs.* The number of runs/trajectories to be utilized from each input file.
3. *Vertex size.* The weighting factor for the relative size of solutions (search space locations) in the STN graphics to be generated.
4. *Arrow size.* The weighting factor for the relative size of arcs among search space locations in the STN graphics to be generated.
5. *Tree layout.* The standard layout of STN graphics are produced by the Fruchterman-Reingold algorithm. The tree layout is optional.

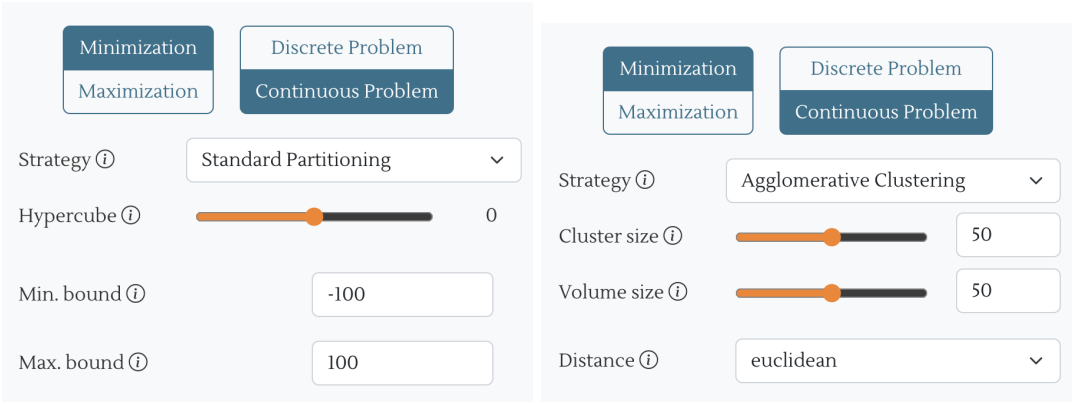


Figure 4: Search space partitioning strategies for continuous problems: (a) Standard Partitioning (b) Agglomerative Clustering.

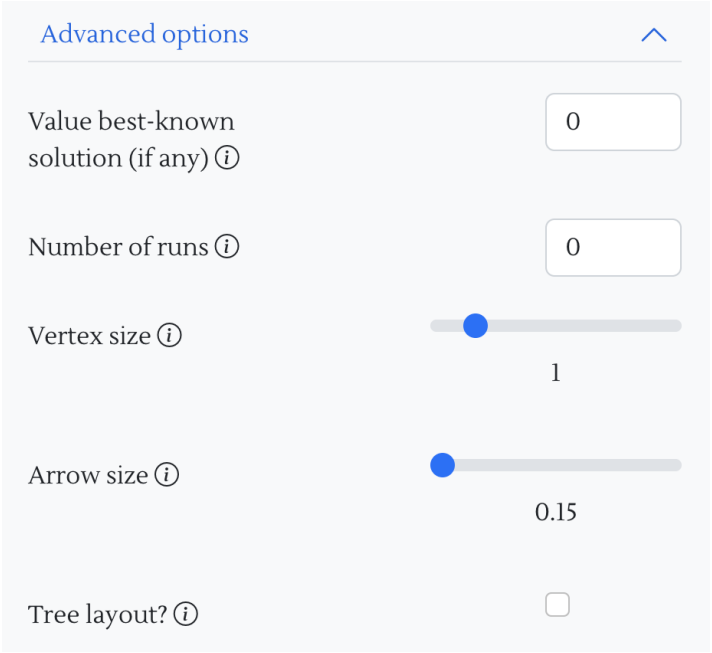


Figure 5: Advanced options.

3.4 Adding algorithms and input files

The first required action for adding an algorithm (see Figure 6) consists in choosing a name. Then, the corresponding input file can be uploaded. In addition, the user can change the color of the algorithm's appearance in the STN graphic to be produced.

Finally, the user has to click on the ADD NEW ALGORITHM button.

3.5 Generating the STN graphic

The last step is to click the GENERATE button. After that, please wait for the PDF to be generated and loaded in the PDF viewer (right panel).

Moreover, once the PDF has been generated and embedded in the right panel, you can download it or download some metrics of the displayed STN in form of a spreadsheet (see Figure 7).

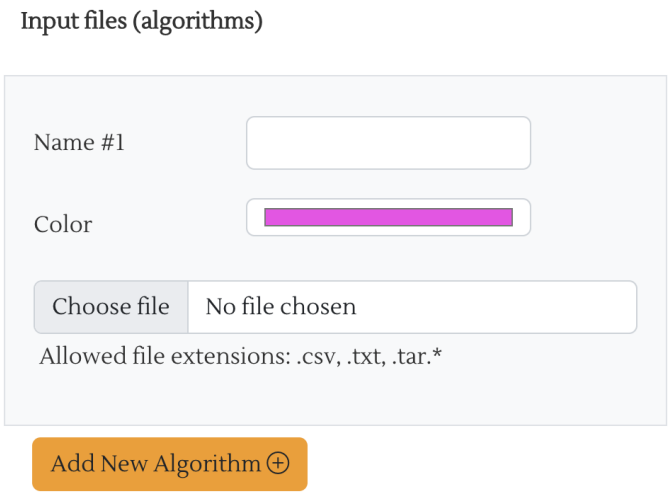


Figure 6: The panel for adding a new algorithm for comparison.

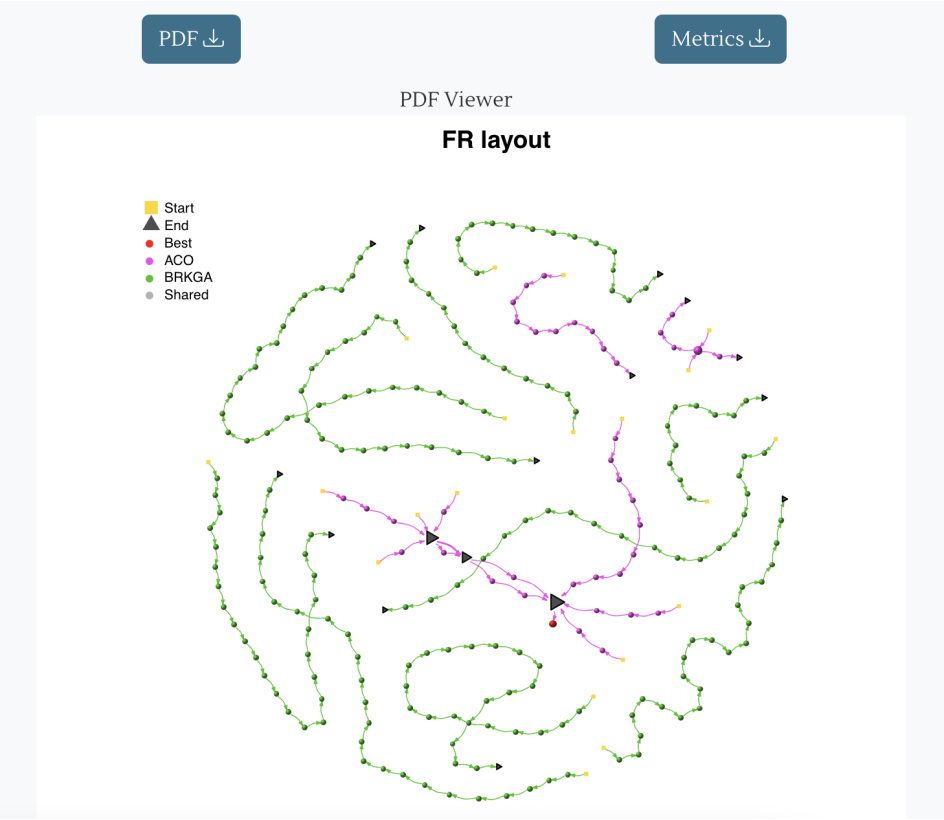


Figure 7: An example of a comparison of two algorithms using STNWeb.

4 Conclusions

In this tutorial, we presented STNWeb, a new visualization web application to enrich analytics in optimization research. The tutorial dealt with the five steps required to generate a visually attractive and useful graphic to be used in your research. For more information on STNs, please see [OMB21].

References

[OMB21] Gabriela Ochoa, Katherine M. Malan, and Christian Blum. Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. *Applied Soft Computing*, 109:107492, 2021.