Improving the efficiency of Euclidean TSP solving in Constraint Programming by predicting effective nocrossing constraints

Elena Bellodi

Alessandro Bertagnon

Marco Gavanelli

Riccardo Zese

Department of Engineering - University of Ferrara

{elena.bellodi,alessandro.bertagnon,marco.gavanelli,riccardo.zese}@unife.it

36TH ITALIAN CONFERENCE ON COMPUTATIONAL LOGIC (CILC 2021)

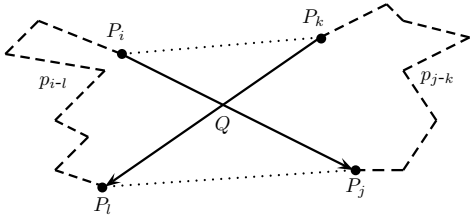
PARMA, SEPTEMBER 7^{th} - 9^{th}, 2021

Università degli Studi di Ferrara

DE: Department of Engineering Ferrara

Euclidean Traveling Salesperson Problem (ETSP)

- Traveling Salesperson Problem (TSP)
- Euclidean TSP (ETSP)
  - Each node i to be visited is associated with a point $P_i = (x_i, y_i)$ in the Euclidean plane, and the cost function $w(P_i, P_j)$ is the Euclidean distance.
- Flood (1956) [Flo56]
  - Let c^* be an optimal tour of a Metric TSP. Then, for each $e_{i,j}, e_{k,l} \in c^*$ such that $\{i, j, k, l\}$ are all different, the segments $\overline{P_i P_j} \cap \overline{P_k P_l} = \emptyset$.



Avoiding crossings

- The `nocrossing` constraint [BG20] imposes that a pair of segments in the TSP should not cross each other (except, possibly, in one endpoint).
- In the **successor** representation, it is defined as follows:

$$\texttt{nocrossing}(i, Next_i, j, Next_j) = \left(\overline{P_i P_{Next_i}} \cap \overline{P_j P_{Next_j}}\right) \subset \{P_i, P_j\}$$

  where i and j are ground variables.
- This constraint should be imposed for each of the $\frac{n(n-1)}{2}$ pairs of nodes, i.e. a quadratic number of constraints !
- Some `nocrossing` constraints never perform any pruning, and only introduce overhead ☹

Evaluating the performance of each `nocrossing` constraint

- Indicators - for each `nocrossing` constraint
  - $N_{activations}$: number of activations of the constraint
  - N_{pruned}: number of value deletions from the domain of the variables involved
  - *Number of failures* (and therefore backtracks) generated as a result of the deletion of values.

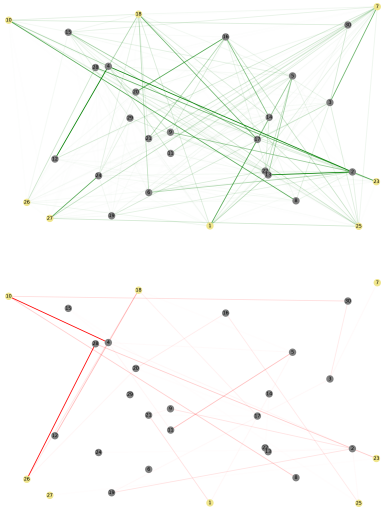


Figure: Graphical representation of the N_{pruned} (*green*) and *Number of failures* (*red*) indicators of each nocrossing constraint in a Euclidean TSP instance. The darker the color of the line, the higher the value.

DE: Department of Engineering Ferrara

Evaluating the performance of each `nocrossing` constraint

$$\texttt{RTIO} = \frac{N_{pruned}}{N_{activations}}$$

- Low `RTIO` $\Rightarrow$ the constraint wakes up many times without being able to perform pruning (overhead)
- High `RTIO` $\Rightarrow$ the constraint can perform a much stronger pruning compared to the number of activations

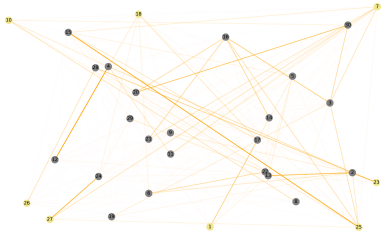


Figure: Graphical representation of the `RTIO` of each `nocrossing` constraint in a Euclidean TSP instance. The darker the color of the line, the higher the `RTIO` value.

Evaluating the performance of each `nocrossing` constraint

- A constraint belonging to a certain instance $\mathcal{I}$ is labeled as *useful* if its `RTIO` is greater than the arithmetic mean calculated on the `RTIO` of all the constraints belonging to instance $\mathcal{I}$, otherwise it is labeled as *useless*.
- The relation we wish to learn could be seen as a function mapping each pair of points (in a generic Euclidean TSP instance) to the set of classes $\{\textit{useful}, \textit{useless}\}$.

Features

We have identified the following 15 features for each $\texttt{nocrossing}(i, Next_i, j, Next_j)$ constraint in the dataset:

- `XPIN`, `YPIN`, `XPJN`, `YPJN`: normalized coordinates of the two points;
- `DISN`: Euclidean distance calculated between points P_i and P_j;
- `LEVI`, `LEVJ`: a numeric value suggesting how deep in the interior of the figure are points P_i and P_j;
- `CXNI`, `CYNI`, `CXNJ`, `CYNJ`: normalized coordinates of the nearest point to P_i and P_j respectively;
- `NDTI`, `NDTJ`: Euclidean distance of the closest point to P_i and P_j respectively;
- `NBHD`: number of points contained in the circle having as diameter the segment connecting points P_i and P_j;
- `NMST`: indicates whether the segment $\overline{P_iP_j}$ belongs to a MST.

Università degli Studi di Ferrara | DE: Department of Engineering Ferrara

Machine learning

- The dataset of labelled constraints is suitable for the application of supervised machine learning algorithms
- We want to predict which of the constraints will be useful and which will be useless in a new unseen instance of the Euclidean TSP.
- Random Forest
  - Random Forest (RF) approach is known for its good computational performance and scalability;
  - The algorithm used in the experimental validation is the one available in the WEKA[1] workbench for machine learning.

[1] https://www.cs.waikato.ac.nz/ml/weka/

Machine learning

1. A training dataset of various Euclidean TSP instances, where each constraint has been labelled according to its own `RTIO`, is used to learn a random forest classifier;
2. Given any new instance of the problem, for whose constraints the label are unknown, apply the classifier to find the only pairs of points that are classified as *useful* for imposing the `nocrossing` constraint;
3. Run the instance together with the selected `nocrossing` constraints to solve the Euclidean TSP.

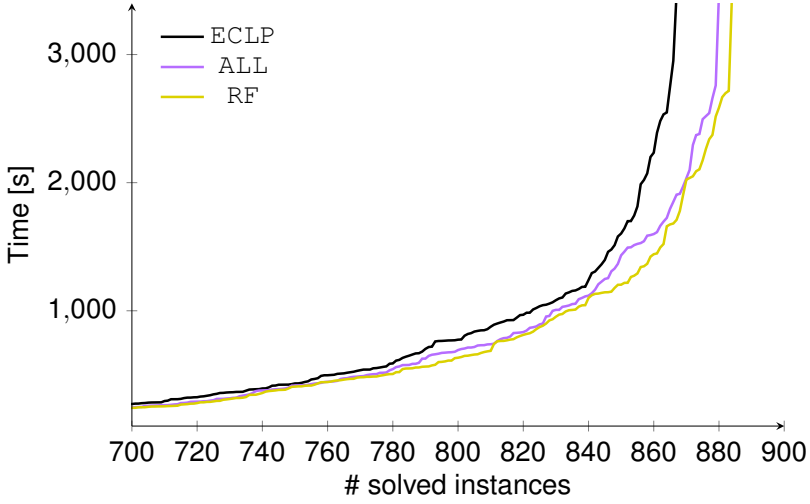
Experiments

- Random Forest Classifier Dataset: 517,120 `nocrossing` constraints (from 1024 instances of Euclidean TSP). Split 66% training, 34% test.

Class	TP Rate (Recall)	FP Rate	Precision	F-Measure	ROC Area	PR Area
0	0.957	0.301	0.932	0.944	0.945	0.984
1	0.699	0.043	0.790	0.742	0.945	0.818

- Euclidean TSP Solver Experiments on 1024 randomly-generated TSPs [JM07], varying the size from 15 to 30 nodes. We compared three constraint models:
  - `ECLP` : a basic constraint model, including the `circuit` and `alldifferent` constraints required by the successor representation plus the objective function;
  - `ALL` : the constraint model imposing the `nocrossing` constraint for all pairs of nodes;
  - `RF` : the model imposing only the `nocrossing` constraints predicted as useful by the classifier.

Experiments



All algorithms are implemented in the ECL^{i}PS^e CLP language [SS12]. The time limit for each run was set to 3480s.

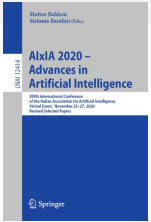
Università degli Studi di Ferrara | DE: Department of Engineering Ferrara

Conclusion and Future work

- Conclusion
  - We proposed to predict and select a subset of useful `nocrossing` constraints in Euclidean Traveling Salesperson Problems (TSPs) formulated in Constraint Programming (CP) by means of machine learning techniques.
  - Experimental results are encouraging, but we believe that much more efficiency could be obtained in future research.
- Future work
  - Testing other supervised learning techniques (e.g. neural networks).
  - Expanding the dataset of experiments (using different search strategies).
  - Enlarging the set of features collected during the creation of the dataset.
  - Use dynamic strategies (removing during search less effective constraints).

For further information...

Bellodi E., Bertagnon A., Gavanelli M., Zese R. (2021) **Improving the Efficiency of Euclidean TSP Solving in Constraint Programming by Predicting Effective Nocrossing Constraints**. In: Baldoni M., Bandini S. (eds) AIxIA 2020 - Advances in Artificial Intelligence. AIxIA 2020. Lecture Notes in Computer Science, vol 12414. Springer.



Università degli Studi di Ferrara | DE: Department of Engineering Ferrara

Thank you for your attention.

References I

- Alessandro Bertagnon and Marco Gavanelli, *Improved filtering for the euclidean traveling salesperson problem in CLP(FD)*, The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 1412–1419.
- M. M. Flood, *The traveling-salesman problem*, Operations Research **4** (1956).
- David S Johnson and Lyle A McGeoch, *Experimental analysis of heuristics for the stsp*, The traveling salesman problem and its variations, Springer, 2007, pp. 369–443.
- Joachim Schimpf and Kish Shen, *Ecl^{i}ps^e - from LP to CLP*, TPLP **12** (2012), no. 1-2, 127–156.

Università degli Studi di Ferrara | DE: Department of Engineering Ferrara

Extended Features 1/3

We have identified the following 45 features for each $\texttt{nocrossing}(i, Next_i, j, Next_j)$ constraint in the dataset:

- 1 - 3: **Cost Matrix Statistics***: Mean (c_{avg}), variation coefficient, skew;
- 4 - 5: **Distance**: Euclidean distance $d_{N_i N_j}$ between points N_i and N_j.
- 6: **Radius***: Mean distance from each node to the centroid;
- 7 - 10: **Centroid Distance**: Euclidean distance from the centroid C to the two extremes N_i (d_{CN_i}) and N_j (d_{CN_j}) respectively.
- 11 - 12: **Levels**: Level of points P_i and P_j.
- 13 - 15: **Cluster Distance Features***: Mean, variation coefficient, skew;
- 16 - 17: **Nearest Neighbour Distance***: Standard deviation and coefficient variation of the normalized nearest neighbour distance;

Extended Features 2/3

- 18 - 21: **Neighbours Distance**: Euclidean distance of the closest point $C(N_i)$ to N_i ($d_{C(N_i)N_i}$) and N_j ($d_{C(N_j)P_j}$) respectively;
- 22: **Neighbourhood size**: Number of points contained in the circle having as diameter the segment connecting points P_i and P_j;
- 23 - 26: **Minimum spanning tree cost statistics***: Sum, mean, variation coefficient, skew;
- 27 - 28: **Shortest Path in MST**: Cost of the shortest path p between N_i and N_j in a minimum spanning tree;
- 29 - 31: **Minimum spanning tree node degree statistics***: Mean, variation coefficient, skew;
- 32 - 33: **MST Degree**: Degree in the minimum spanning tree of P_i and P_j respectively;

Università degli Studi di Ferrara | DE: Department of Engineering Ferrara

Extended Features 3/3

- 34: **Segment in MST** (boolean value) Indicates whether the segment $\overline{P_i P_j}$ belongs to a minimum spanning tree.
- 35 - 36: **Segment crosses**: Number of segments crossing the segment $\overline{P_i P_j}$ and the version normalized, obtained dividing by the total number of arcs;
- 37 - 40: **Crossings**: Total number of crossings between edges exiting from P_i (resp. P_j) and other edges.
- 41 - 45: **Fraction of distinct distances***, with precision to $k \in \{1, 2, 3, 4\}$ decimal digits.
- 46: **Good**: (boolean value) label each constraint as useful (1) or useless (0).

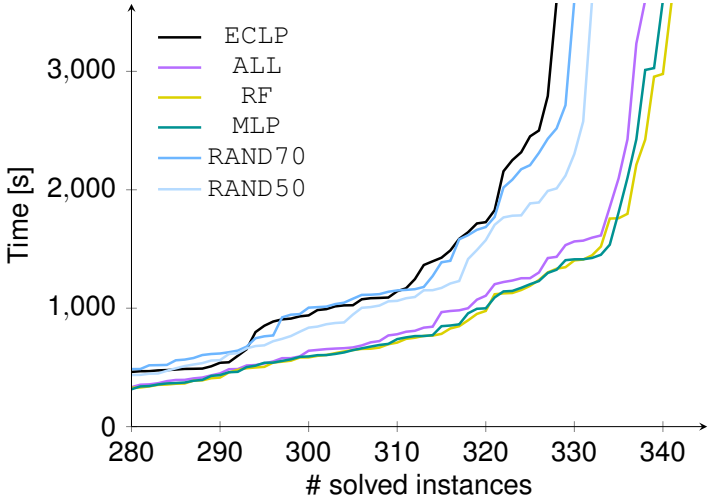
Extended Experiments

Dataset of 434,352 `nocrossing` constraints - 50% labelled as *useful* and 50% as *useless* - collected from 1536 instances of Euclidean TSP (split 66% training, 34% test).

Table: Performance of the RF and MLP classifiers over the fraction of instances used for test. Class 0 corresponds to ***useless*** constraints.

Classifier	Class	Recall	FP Rate	Precision	F-Measure	ROC Area	PR Area	Accuracy
RF	0	0.760	0.155	0.830	0.793	0.883	0.894	
	1	0.845	0.240	0.779	0.811	0.883	0.862	
	W.Avg.	0.802	0.198	0.804	0.802	0.883	0.878	0.802
MLP	0	0.827	0.258	0.704	0.761	0.855	0.871	
	1	0.742	0.173	0.853	0.794	0.855	0.829	
	W.Avg.	0.785	0.215	0.779	0.777	0.855	0.850	0.779

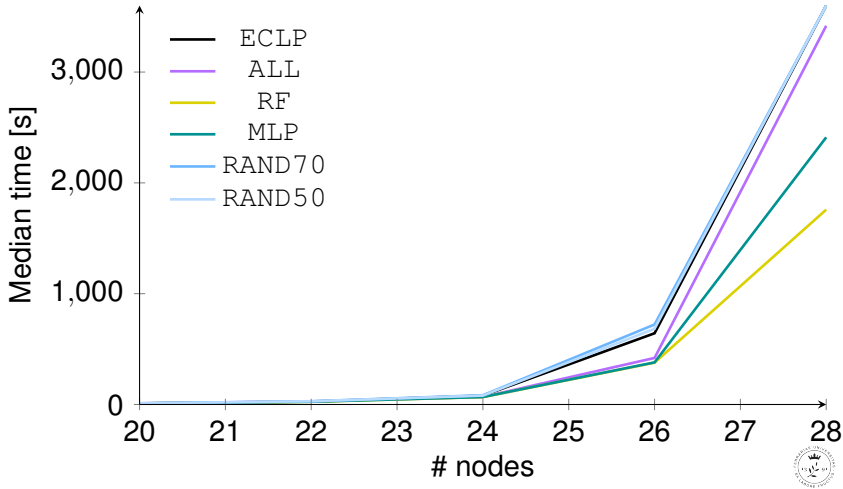
Extended Experiments



All algorithms are implemented in the ECL^{i}PS^e CLP language [SS12]. The time limit for each run was set to 3480s.

Università degli Studi di Ferrara | DE: Department of Engineering Ferrara

Experiments



Università degli Studi di Ferrara | DE: Department of Engineering Ferrara