Constraint Programming Algorithms for Route Planning Exploiting Geometrical Information

Alessandro Bertagnon
Department of Engineering
University of Ferrara
Ferrara, Italy
alessandro.bertagnon@unife.it

Problems affecting the transport of people or goods are plentiful in industry and commerce and they also appear to be at the origin of much more complex problems. In recent years, the logistics and transport sector keeps growing supported by technological progress, i.e. companies to be competitive are resorting to innovative technologies aimed at efficiency and effectiveness. This is why companies are increasingly using technologies such as Artificial Intelligence (AI), Blockchain and Internet of Things (IoT). Artificial intelligence, in particular, is often used to solve optimization problems in order to provide users with the most efficient ways to exploit available resources.

In this work we present an overview of our current research activities concerning the development of new algorithms, based on Constraint Logic Programming (CLP) techniques, for route planning problems exploiting the geometric information intrinsically present in many of them or in some of their variants. The research so far has focused in particular on the Euclidean Traveling Salesperson Problem (Euclidean TSP) with the aim to exploit the results obtained also to other problems of the same category, such as the Euclidean Vehicle Routing Problem (Euclidean VRP), in the future.

1 Introduction

Given a weighted graph G with n vertices the Traveling Salesperson Problem (TSP) requires to compute the shortest cycle that visits each vertex of G exactly once [1]. The name "Traveling Salesman Problem" comes from the problem's most famous formulation: "A salesman has to visit a set of cities, each of which must be visited only once, and he wants to minimize the length of the tour". The problem is NP-hard [19]. Currently, the best solver for the TSP is Concorde [1], that includes several techniques based on Integer Linear Programming (with branch-and-bound and branch-and-cut) and Local Search.

Some significant sub-classes of the general TSP are the *metric TSP*, in which the distance function between cities enjoys the triangle inequality, and the *Euclidean TSP*, in which the nodes of the graph represent points in the plane and the distance function is the Euclidean distance. These are reasonable assumptions in many important instances: many industrial problem and various benchmarks taken from the TSPLIB [26] fall into these classes.

Both the metric and the Euclidean TSP, as the general TSP, are NP-hard [15]; nonetheless, differently from the general TSP, the Euclidean TSP admits a Polynomial Time Approximation Scheme (PTAS) [2, 23], i.e., given a value $\varepsilon > 0$, it is possible to obtain a solution with cost $(1+\varepsilon)L^*$ (where L^* is the length of the optimal tour) in polynomial time with respect to the number of nodes n (note, however, that the time is exponential with respect to $\frac{1}{\varepsilon}$).

It is worth noting that in the Euclidean TSP more information is available than in the general TSP: the coordinates of the points to be visited are known, and geometrical concepts (straight line segments,

[1] A cycle that visit each vertex exactly once is often referred as Hamiltonian cycle or Hamiltonian circuit.

angles, etc.) can be defined in the Euclidean plane. Despite these results are very important theoretically, TSP solvers do not use the additional geometric information which is available in the Euclidean TSP instances. Concorde, which is in practice faster in most applications, discards the information regarding vertices position on the plane and it computes the distance matrix using Euclidean distances as weights.

In the literature another important work that attempted to use geometric information has recently been published: Deudon et al. [7] train a Deep Neural Network with points coordinates to learn efficient heuristics to explore the search space. This work, instead, is the first attempt (to the best of our knowledge) to exploit geometric information to obtain further pruning in Constraint Programming (CP) during the solution of some route planning problems.

2 Related Works

As mentioned in the previous section, the best solver currently available for the TSP is Concorde[1]; but Concorde can address only *pure* TSPs, i.e., no further side constraints are allowed, while in CP many variants can be easily cast, such as the TSP with Time Windows (TSPTW).

Three representations have been devised, in CP literature, for defining variables in the Hamiltonian circuit problem and the TSP: the *permutation* representation, the *successor* representation and the *set variable* representation [4] later extended to the graph representation [8, 9, 10]. In the following of this paper, we will be mainly concerned with the successor representation.

Given a list L of variables $[Next_1, Next_2, ..., Next_n]$, where n is the number of vertices of the graph G, we denote with $Dom(Next_i)$ the initial domain of the variable $Next_i$, where $Dom(Next_i) = \{1, \dots, n\} \setminus \{i\}$. In the successor representation, the value of the $Next_i$ variable denotes the successor of the vertex i in the resulting tour (e.g. if $n = 5$ and $L = [3,5,4,2,1]$ the corresponding tour will be $1,3,4,2,5,1$). Note that in the Euclidean TSP each vertex i always corresponds to one point $P_i = (x_i, y_i)$ in the plane.

The constraint model includes an `alldifferent`(L) constraint [25] on the list L of all variables, that ensures that each node has exactly one incoming edge, as well a `circuit`(L) [3, 6, 20] constraint (sometimes called `nocycle`) that avoids sub-tours, i.e., cycles of length less than n.

Caseau and Laburthe [6] propose a simple but efficient propagation algorithm for `circuit` constraint (nocycle) combined with new branching strategies based on the combination of first-fail and max-regret. First-fail selects first the variable with smallest domain because it has a higher probability of running out of elements and thus lead to failure. The regret of a variable is defined, instead, as the cost difference between its two best possible assignments, so the strategy is to first select those variables in which the regret is higher in order to avoid a significant increase in the cost of the solution. They also propose to filter values based on the objective function. For this last purpose, they apply the assignment-based and the spanning tree relaxations.

Another filtering rule proposed for the `circuit` constraint is that of Kaya and Hooker [20] based on graph separators theory. Francis and Stuckey [14] compare the effectiveness of different propagation algorithm for `circuit` when adding explanation in the context of a lazy clause generation solver.

Pesant et al. [24] address the TSPTW, in which cities must be visited within given temporal intervals, and exploit the `circuit` constraint together with the minimum spanning tree relaxation. Focacci et al. [12, 13] introduce hybrid approaches that merge CP and Operations Research techniques, including: reduced costs filtering, use of the assignment problem and minimum spanning forest relaxation.

A filtering technique presented more recently by Benchimol et al. [4] for the Weighted Circuit Constraint (WCC) has been able to obtain, on medium dimension instances, results comparable with the solver Concorde. In their work, they use a variety of techniques. In order to obtain an initial upper bound to be used for propagation, they first run the Lin-Kernighan-Helsgaun algorithm [22, 17]. Then the filtering technique uses the Held and Karp [16] scheme together with a Lagrangian relaxation to obtain the reduced costs of the edges and uses it to remove them. The algorithm is also able to identify the mandatory arcs that if removed would increase the current lower bound. In the experiments with asymmetric TSPs, they also use additive bounding [11] to combine both the 1-tree and the assignment problem relaxations.

Fages et al. [10], achieve significant improvement by casting the problem in CP(Graph) and through the introduction of improved search strategies (i.e. Last Conflict heuristic). Fages and Lorca [9] shown how properties of the reduced graphs associated to an Asymmetric TSPs can be used to improve the Minimum Spanning Tree relaxation. The nodes of the reduced graphs are the Strongly Connected Components of the original graph.

Isoart and Régin [18] design a propagator based on the search of k-cutsets. The combination of this constraint with the WCC constraint has resulted in a significant reduction in the computation time.

3 Ongoing research

Several algorithms have already been developed, some of them have recently been published in their first version [5] while others are still being studied.

3.1 The nocrossing constraint

A well-known finding in the literature (e.g., [2]) is that the optimal solution of Euclidean TSP does not have crossing edges (see Figure 1).

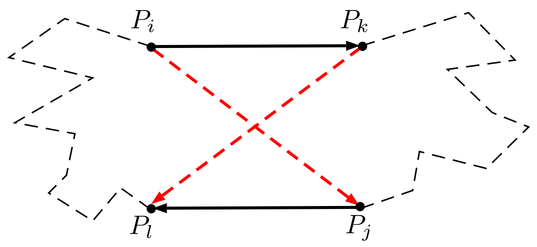

Figure 1: Crossings elimination leads to a shorter route.

Property 1. *There is no crossing in the optimal solution for a Euclidean TSP.*

We can take advantage of Property 1 to reduce the size of the search tree by eliminating branches presenting crosses. We propose the `nocrossing` constraint that eliminates assignments leading to non-optimal solutions. In the successor representation, it is defined as follows.

Definition 1. *The* `nocrossing`$(i, Next_i, j, Next_j)$ *constraint assures that segments* $\overline{iNext_i}$ *and* $\overline{jNext_j}$ *do not cross each other.*

The `nocrossing` constraint is a binary constraint since i and j are ground values at the time when it is imposed. Suppose we have n nodes, to avoid crossings $n(n-1)/2$ constraints are introduced, one constraint for each pair of nodes of the graph. The `nocrossing` constraint can be implemented thorough a pair of propagators: one propagates changes in the domain of the variable $Next_i$ on the domain of the variable $Next_j$, while the other propagator propagates changes on the other way.

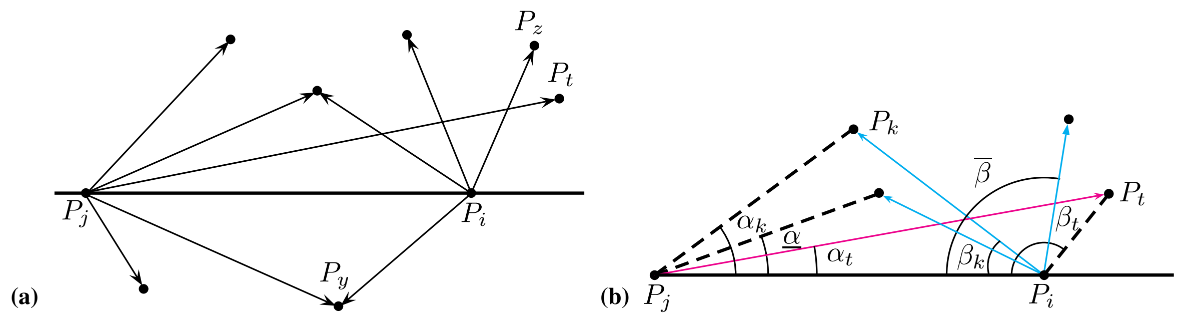

Figure 2: An arrow from P_x to P_y means that $y \in Dom(Next_x)$. Dashed lines are plotted to show the angles.

The nocrossing constraint could be implemented in a naive manner, for example using the table constraint [28] or the propia library [21]. A table constraint consists of a table (usually a list of tuples) of values that the involved variables must, or must not, assume. However this implementation would be inefficient, because the constraint wakes up most of the time without being able to carry out any propagation. Moreover, with the table constraint one should initially compute large tables, containing, for all pairs of edges in the graph, if they cross or not. Propagating such constraint would have the usual cost of arc-consistency propagation for a single constraint of $O(d^2)$ (if d is the size of the domains) in each activation of the constraint.

From the definition of arc-consistency and Definition 1, a value v can be removed from $Dom(Next_j)$ only if the segment $\overline{P_jP_v}$ intersects all possible segments originating from P_i (e.g. segment $\overline{P_jP_t}$ in Figure 2a).

We define a *possible segment* from P_i a segment $\overline{P_iP_j}$ such that $j \in Dom(Next_i)$. We denote with $\overleftrightarrow{P_iP_j}$ the (infinite straight) line passing through points P_i and P_j, and with $\angle P_iP_jP_k$ the counterclockwise angle formed by the segments $\overline{P_iP_j}$ and $\overline{P_jP_k}$ with vertex in P_j from P_i to P_k.

We have identified the following necessary and sufficient conditions for pruning and we have exploited them within our propagation algorithm.

Theorem 1 (Necessary condition)**.** *Let $\overline{P_jP_t}$ a possible segment from P_j. If $\overline{P_jP_t}$ crosses all possible segments from P_i then all segments originating from P_i must lie on the same half-plane with respect to the line $\overleftrightarrow{P_iP_j}$.*

The propagator is suspended waiting that all elements in the domain of $Next_i$ lie on the same half-plane. We select one element in each half-plane and the propagator suspends waiting that one of the two elements is removed from $Dom(Next_i)$.

Theorem 2 (Necessary condition)**.** *Let $\overline{P_jP_t}$ a possible segment from P_j. If $\overline{P_jP_t}$ crosses all possible segments from P_i then $\alpha_t \leq \underline{\alpha}$, where $\underline{\alpha} = \min\{\alpha_q \mid q \in Dom(Next_i)\}$ and $\alpha_k = \angle P_iP_jP_k$ (see Figure 2b).*

Theorem 3 (Sufficient condition)**.** *Suppose there exists a possible segment $\overline{P_jP_t}$ from P_j such that $\alpha_t \leq \underline{\alpha}$. If $\overline{P_jP_t}$ crosses all possible segments from P_i then $\beta_t > \overline{\beta}$, where $\overline{\beta} = \max\{\beta_q \mid q \in Dom(Next_i)\}$ and $\beta_k = \angle P_kP_iP_j$ (see Figure 2b).*

Thanks to these theorems, the nocrossing propagator can be implemented with $O(d)$ complexity (if d is the size of the domains) per each activation (to be compared to the $O(d^2)$ of a naive propagator).

A study we are conducting to evaluate the actual performance of the nocrossing constraint is shown in Figure 3.

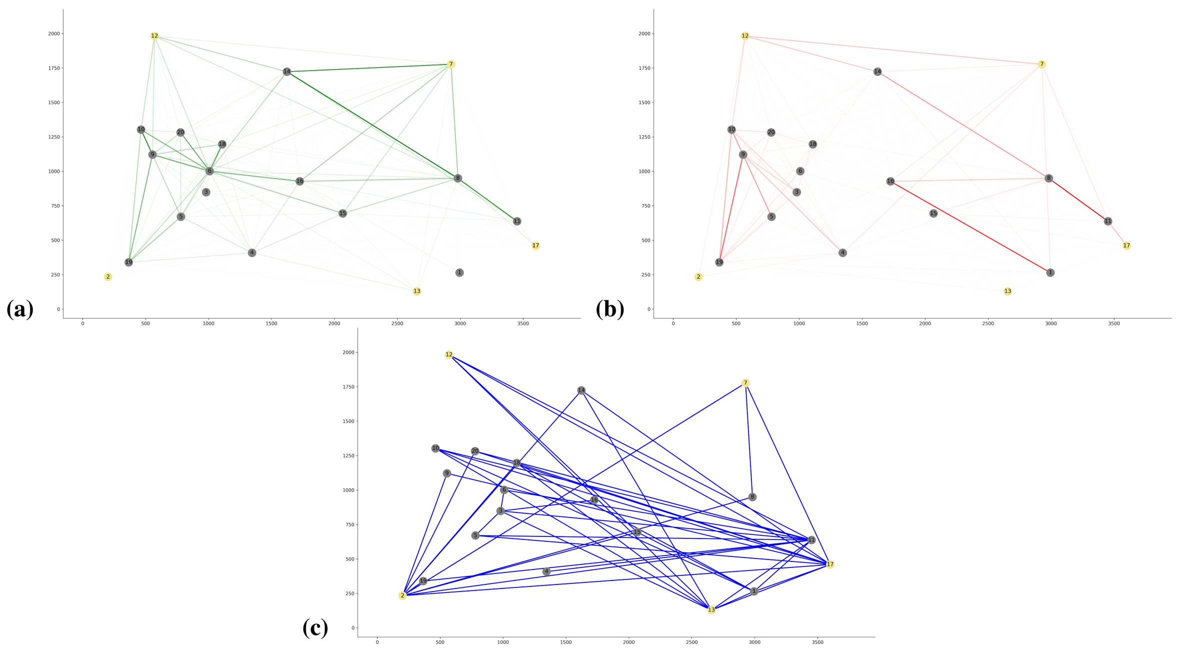

Figure 3: A graphical representation of the `nocrossing` constraint while solving an instance with 20 nodes. The darker the color, the higher is the value for the following indicators: (a) number of value deletion; (b) number of failures; (c) instances of `nocrossing` that have not performed pruning (binary value: true, false).

3.2 The clockwise constraint

Further pruning can be obtained by means of the convex hull of the set of points. We recall the definition of convex hull.

Definition 2. *Let S be a set of points, the convex hull of S is defined as the smallest convex set containing S. The convex hull is therefore the smallest convex polygon that includes all the points of the set S.*

The following property resulting from Property 1 can also be exploited to reduce the space of possible solutions.

Property 2. *Let k of the n points in the Euclidean TSP be vertices on the boundary of the convex hull. Then the order in which these k points appear in the optimum traveling salesman tour must be the same as the order in which these same points appear on the boundary of the convex hull.*

From Property 2 follows that the optimal solution of the TSP is is a simple polygon, and it divides the plane into exactly two areas: an *internal* and an *external* area. Please observe that the points on the border of the convex hull do not necessarily appear consecutively in the solution, an example is reported in Figure 4a.

Let $V = \{1,2,...,n\}$ be the set of vertices, where n is the number of vertices of the graph, and let $H = \{(h_1,h_2,...,h_k)|h_i \in V, i = 1,..,k\}$ be the list of vertices that lies on the boundary of the convex hull ordered clockwise. The following is a summary of three different ways we proposed to exploit the information about the convex hull for constraint propagation.

The simplest way to satisfy the Property 2 is to apply the following relation:

$$\forall i \in H, Dom(Next_i) \cap \{h_j | j = 1,...k, j \neq (i+1)\} = \emptyset \qquad (1)$$

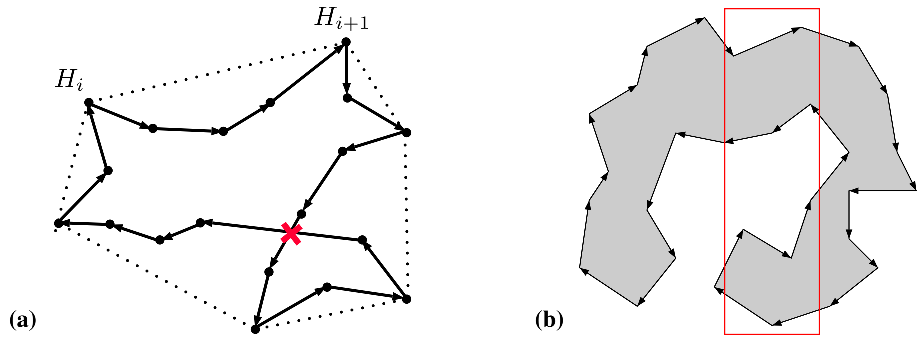

Figure 4: (a) Visiting vertices of the perimeter of the convex hull not in order generates a cross in the solution of the TSP, therefore that solution cannot be optimal. (b) Imagine to cut the optimal TSP with two vertical lines, the convex hull reasoning can be extended to the borders (i.e., the parts of the circuit inside the stripe).

Equation 1 states that it is possible to eliminate from the domain of each point in H all the other points on the boundary of the convex hull except the next in the sequence (see Figure 5a).

Further propagation can be carried out when the domain associated with a point on the boundary of the convex hull becomes ground. Let H_i be the index of the point on the boundary of the convex hull whose domain has become ground and P its value, i.e., in the current assignment there is the $\overline{H_iP}$ segment. If we choose to go through the points on the boundary of the convex hull in a clockwise direction, in order to satisfy Property 2, no points situated on the left of the $\overline{H_iP}$ segment can have H_i as its successor (see Figure 5b)

The third way is to impose that each path originating from a convex hull vertex cannot reach any convex hull vertex except for the one immediately following it. Implementation is inspired by the `circuit` constraint [6], but performs more powerful pruning (see Figure 5c). It is clear that this third propagation also implies the first one.

The applicability of the three convex hull propagators presented can also be extended to points that lie in the interior of the hull. Once a partial path has been defined, consider the polygon formed by such path plus a segment connecting its extremes. We can apply the propagators to the convex hull of the points inside that polygon (see Figure 4b).

We propose the `clockwise` constraint that implements all the propagation described in this section.

4 Preliminary Result

In order to assess the effectiveness of the proposed algorithms, we devised a series of experiments based on randomly-generated TSPs and taken from structured instances (e.g. TSPLIB [26]).

We implemented all algorithms using the ECL^{i}PS^e CLP language [27]. We started comparing our algorithms with a simple model for the successor representation (that includes `alldifferent` and `circuit` constraints), such a constraint model is named `CLP(FD)` in the following. Then in order to show that the pruning we provide is not subsumed by that of state of the art techniques, we implemented in ECL^{i}PS^e, in the successor representation, also the Held and Karp bound with pruning based on reduced and marginal costs, as proposed by Benchimol et al. [4] (shown with `BvHRRR` in the following). The constraint model, named `Geometric`, includes the `nocrossing` constraint on all pairs of vertices, together with the `clockwise` constraint that implements the propagation described in Section 3.2. The

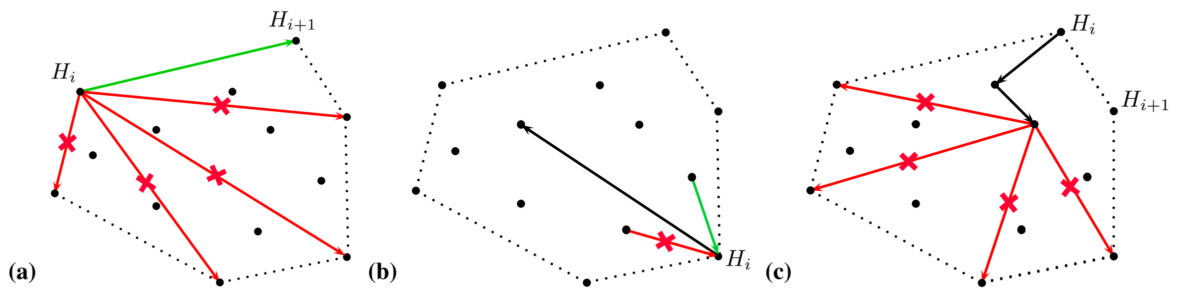

Figure 5: Depiction of three different ways to exploit the information about the convex hull for constraint propagation as implemented in the `clockwise` constraint. The boundary of the convex hull is represented with dotted lines. (a) The successor of a convex hull vertex cannot be another vertices on the boundary of the convex hull except for the one that immediately follows it. (b) In order to visit nodes in a clockwise order, the angle between the incoming edge and the outgoing edge of a convex hull vertex cannot be positive (it must be between $-\pi$ and 0). (c) On each paths originating from a convex hull vertex H_i the first convex hull vertex different from H_i is the one immediately following it, that is H_{i+1}.

`Geometric` model also includes bounds with the Lin-Kernighan-Helsgaun [17]) as included in `BvHRRR`.

In Figure 6 we present cactus plots that compare the impact of our filtering algorithms with the constraints already predefined in ECL^iPS^e system. In the x-axis we report the number of optimally solved instances, and in the y-axis the solving time in seconds. Fixed a certain amount of time, Figure 6 clearly illustrates that our filtering algorithms, especially when used simultaneously, significantly increase the number of instances solved.

In Figure 7 we present cactus plot that compare the impact of our filtering algorithm with the pruning proposed by Benchimol et al. [4]. The addition of the filtering on geometric properties improves the runtime.

Admittedly, the instances we were able to solve are not as large as those addressed by state of the art techniques in CP [4, 10, 9]. This could be due to several factors. Our implementation is based on declarative languages that have the advantage of separating problem definition (as much as possible) from implementation details. On the other hand, their performance can be significantly slower than imperative languages, not only due to interpretation vs compilation schemes, but also to the availability of efficient data structures and search techniques.

Although the results achieved so far are a solid starting point, we are still working on improving the performance and making our algorithms scalable to solve larger instances.

5 Conclusion and Future Works

In this paper, we proposed to exploit geometric information while solving route planning problems in order to have additional pruning with respect to the techniques already available in CP.

The pruning of the WCC, proposed by Benchimol et al. [4], is orthogonal with respect to that of the `nocrossing` and `clockwise` constraints. The quality of the propagation carried out by the WCC is proportional to the quality of the current upper bound on the cost of the solution of the TSP, while our propagator can delete values from the domains even if no bound is known yet.

Despite the preliminary results that we have achieved and presented in this paper are good, our approaches are still not competitive with Concorde especially for large instances.

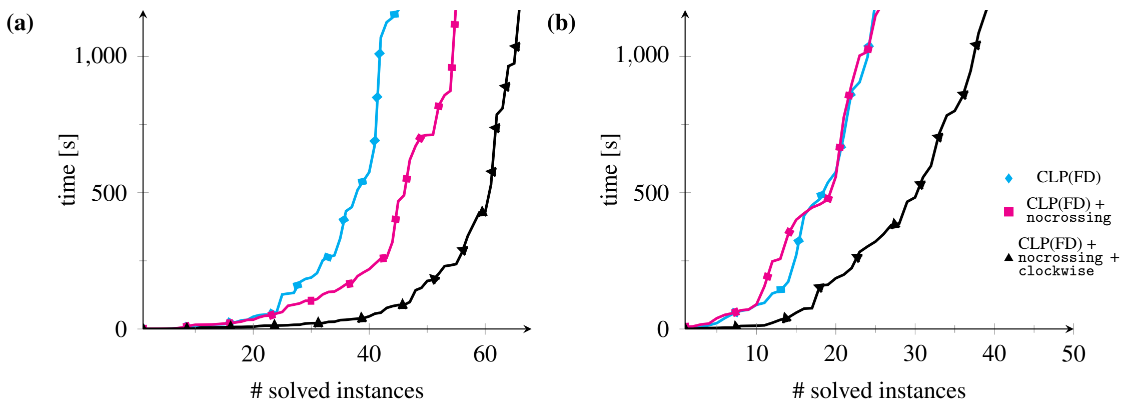

Figure 6: Cactus plot of filtering algorithms which we run over 68 randomly-generated Euclidean TSP instances with 20 nodes (a) and 24 nodes (b). Time limit was set to 1200 seconds.

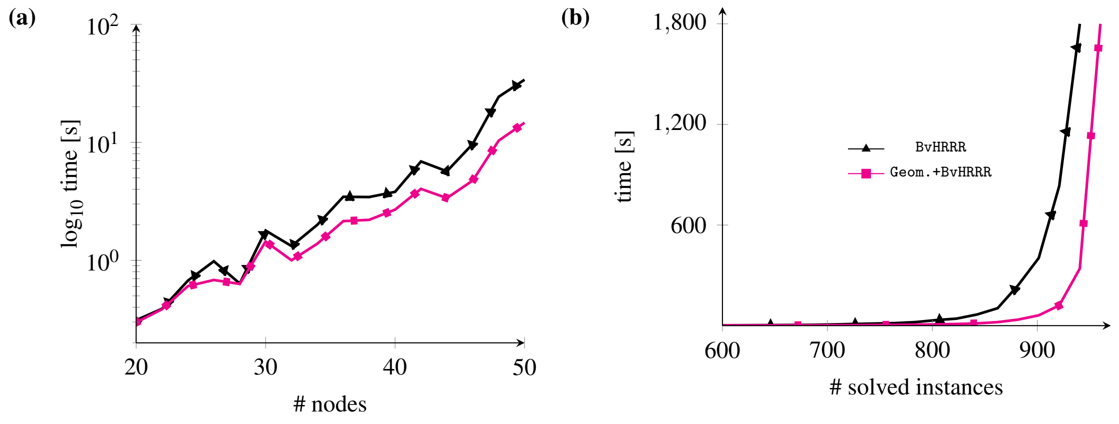

Figure 7: Experimental results on randomly-generated Euclidean TSP instances from 20 to 50 nodes in steps of 2. For each size we generated 60 instances (30 uniform and 30 clustered). Time limit was set to 1800 seconds. (a) Average solving time of filtering algorithms varying the size of the instances. Each point is the average solving time of 60 instances. (b) Cactus plot showing the number of solved instances varying the solving time.

We used the *successor* representation, while the *set variable* or the *graph* representations should be experimented extensively, possibly mixing different representations with tunneling constraints.

As future work we also plan to apply extensions of the proposed techniques in the Euclidean VRP and possibly to other similar problems. In a Vehicle Routing Problem (VRP) there is a fleet of vehicles that must reach all the nodes of a graph; in each node the vehicle must load (or unload) goods. Each vehicle has a capacity that must not be exceeded, so it is not possible to use a single vehicle (with only one vehicle the VRP boils down to the TSP). It can be seen that in an optimal solution of Euclidean VRP there are no intersections in the path of each vehicle, even though there may be intersections between the paths of different vehicles.

We are also planning to extend some of the techniques presented here in CLP to Answer Set Programming (ASP).

Acknowledgments

The author wish to thank Prof. Marco Gavanelli for supervising his PhD research activities and Joachim Schimpf for his help in the implementation of some of the constraints. This work was partially supported by GNCS-INdAM.

References

[1] David Applegate, Robert E. Bixby, Vasek Chvátal & William J. Cook (2001): *TSP Cuts Which Do Not Conform to the Template Paradigm*. In Michael Jünger & Denis Naddef, editors: *Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions [based on a Spring School, Schloß Dagstuhl, Germany, 15-19 May 2000], Lecture Notes in Computer Science* 2241, Springer, pp. 261–304, doi:10.1007/3-540-45586-8_7.

[2] Sanjeev Arora (1996): *Polynomial Time Approximation Schemes for Euclidean TSP and Other Geometric Problems*. In: *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, IEEE Computer Society, pp. 2–11, doi:10.1109/SFCS.1996.548458.

[3] N Beldiceanu & E Contejean (1994): *Introducing Global Constraints in CHIP*. *Math. Comput. Model.* 20(12), pp. 97–123, doi:10.1016/0895-7177(94)90127-9.

[4] Pascal Benchimol, Willem Jan van Hoeve, Jean-Charles Régin, Louis-Martin Rousseau & Michel Rueher (2012): *Improved filtering for weighted circuit constraints*. *Constraints An Int. J.* 17(3), pp. 205–233, doi:10.1007/s10601-012-9119-x.

[5] Alessandro Bertagnon & Marco Gavanelli (2020): *Improved Filtering for the Euclidean Traveling Salesperson Problem in CLP(FD)*. In: *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, pp. 1412–1419, doi:10.1609/aaai.v34i02.5498 0.

[6] Yves Caseau & François Laburthe (1997): *Solving Small TSPs with Constraints*. In Lee Naish, editor: *Logic Programming, Proceedings of the Fourteenth International Conference on Logic Programming, Leuven, Belgium, July 8-11, 1997*, MIT Press, pp. 316–330, doi:10.7551/mitpress/4299.003.0028.

[7] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak & Louis-Martin Rousseau (2018): *Learning Heuristics for the TSP by Policy Gradient*. In Willem Jan van Hoeve, editor: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26-29, 2018, Proceedings, Lecture Notes in Computer Science* 10848, Springer, pp. 170–181, doi:10.1007/978-3-319-93031-2_12.

[8] Grégoire Dooms, Yves Deville & Pierre Dupont (2005): *CP(Graph): Introducing a Graph Computation Domain in Constraint Programming*. In Peter van Beek, editor: *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings, Lecture Notes in Computer Science* 3709, Springer, pp. 211–225, doi:10.1007/11564751_18.

[9] Jean-Guillaume Fages & Xavier Lorca (2012): *Improving the Asymmetric TSP by Considering Graph Structure*. *CoRR* abs/1206.3437.

[10] Jean-Guillaume Fages, Xavier Lorca & Louis-Martin Rousseau (2016): *The salesman and the tree: the importance of search in CP*. *Constraints* 21(2), pp. 145–162, doi:10.1007/s10601-014-9178-2.

[11] Matteo Fischetti & Paolo Toth (1992): *An additive bounding procedure for the asymmetric travelling salesman problem*. *Math. Program.* 53, pp. 173–197, doi:10.1007/BF01585701.

[12] Filippo Focacci, Andrea Lodi & Michela Milano (2002): *Embedding Relaxations in Global Constraints for Solving TSP and TSPTW*. *Ann. Math. Artif. Intell.* 34(4), pp. 291–311, doi:10.1023/A:1014492408220.

[13] Filippo Focacci, Andrea Lodi & Michela Milano (2002): *A Hybrid Exact Algorithm for the TSPTW*. *INFORMS Journal on Computing* 14(4), pp. 403–417, doi:10.1287/ijoc.14.4.403.2827.

[14] Kathryn Glenn Francis & Peter J. Stuckey (2014): *Explaining circuit propagation*. *Constraints* 19(1), pp. 1–29, doi:10.1007/s10601-013-9148-0.

[15] M. R. Garey, R. L. Graham & D. S. Johnson (1976): *Some NP-complete Geometric Problems*. In: *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, STOC '76, ACM, New York, NY, USA, pp. 10–22, doi:10.1145/800113.803626.

[16] Michael Held & Richard M. Karp (1970): *The Traveling-Salesman Problem and Minimum Spanning Trees*. *Operations Research* 18(6), pp. 1138–1162, doi:10.1287/opre.18.6.1138.

[17] Keld Helsgaun (2000): *An effective implementation of the Lin-Kernighan traveling salesman heuristic*. *European Journal of Operational Research* 126(1), pp. 106–130, doi:10.1016/S0377-2217(99)00284-2.

[18] Nicolas Isoart & Jean-Charles Régin (2019): *Integration of Structural Constraints into TSP Models*. In Thomas Schiex & Simon de Givry, editors: *Principles and Practice of Constraint Programming - 25th International Conference, CP 2019, Stamford, CT, USA, September 30 - October 4, 2019, Proceedings*, *Lecture Notes in Computer Science* 11802, Springer, pp. 284–299, doi:10.1007/978-3-030-30048-7_17.

[19] Richard M. Karp (1972): *Reducibility Among Combinatorial Problems*. In Raymond E. Miller & James W. Thatcher, editors: *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, Plenum Press, New York, pp. 85–103, doi:10.1007/978-1-4684-2001-2_9.

[20] Latife Genç Kaya & John N. Hooker (2006): *A Filter for the Circuit Constraint*. In Frédéric Benhamou, editor: *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings*, *Lecture Notes in Computer Science* 4204, Springer, pp. 706–710, doi:10.1007/11889205_55.

[21] Thierry Le Provost & Mark Wallace (1993): *Generalized Constraint Propagation over the CLP Scheme*. *J. Log. Program.* 16(3), pp. 319–359, doi:10.1016/0743-1066(93)90047-K.

[22] S. Lin & Brian W. Kernighan (1973): *An Effective Heuristic Algorithm for the Traveling-Salesman Problem*. *Operations Research* 21(2), pp. 498–516, doi:10.1287/opre.21.2.498.

[23] Joseph S. B. Mitchell (1999): *Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems*. *SIAM J. Comput.* 28(4), pp. 1298–1309, doi:10.1137/S0097539796309764.

[24] Gilles Pesant, Michel Gendreau, Jean-Yves Potvin & Jean-Marc Rousseau (1998): *An Exact Constraint Logic Programming Algorithm for the Traveling Salesman Problem with Time Windows*. *Transportation Science* 32(1), pp. 12–29, doi:10.1287/trsc.32.1.12.

[25] Jean-Charles Régin (1994): *A Filtering Algorithm for Constraints of Difference in CSPs*. In Barbara Hayes-Roth & Richard E. Korf, editors: *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1.*, AAAI Press / The MIT Press, pp. 362–367.

[26] Gerhard Reinelt (1991): *TSPLIB - A Traveling Salesman Problem Library*. *INFORMS Journal on Computing* 3(4), pp. 376–384, doi:10.1287/ijoc.3.4.376.

[27] Joachim Schimpf & Kish Shen (2012): *ECL^iPS^e - From LP to CLP*. *Theory Pract. Log. Program.* 12(1-2), pp. 127–156, doi:10.1017/S1471068411000469.

[28] Neng-Fa Zhou (2009): *Encoding Table Constraints in CLP(FD) Based on Pair-Wise AC*. In Patricia M. Hill & David Scott Warren, editors: *Logic Programming, 25th International Conference, ICLP 2009, Pasadena, CA, USA, July 14-17, 2009. Proceedings*, *Lecture Notes in Computer Science* 5649, Springer, pp. 402–416, doi:10.1007/978-3-642-02846-5_33.