An Exact Framework for the Trigger Arc Traveling Salesman Problem*

Miguel Miní[1], Lucas de Oliveira Silva[1], Flávio K. Miyazawa[1]

[1]Institute of Computing – University of Campinas (Unicamp)
Campinas – SP – Brazil

m.alessandro.mini@gmail.com,

{lucas.oliveira.silva,fkm}@ic.unicamp.br

Abstract. *The Trigger Arc Traveling Salesman Problem generalizes the classical TSP by introducing dynamically evolving arc costs that are modified upon traversal of designated "trigger" arcs. Thus, each arc's cost is determined by the most recently traversed trigger arc that affects it, leading to an order-dependent cost. Such a structure makes an explicit modeling of trigger arcs computationally expensive, even for moderately sized instances. We propose an exact framework based on a standard MIP formulation of TSP, using classic rounding heuristics, together with a callback to separate trigger violations in candidate tours. Also, to control model size, we incorporate triggers progressively only as needed. The proposed approach solves several benchmark instances from the literature to optimality and achieves single-digit optimality gaps on a subset of larger instances.*

1. Introduction

The Trigger Arc Traveling Salesman Problem (TA-TSP) was recently introduced by Cerrone et al. to model routing settings in which arc traversal costs depend on the path already taken [Cerrone et al. 2025]. Their motivating application arises in warehouse operations with compact storage systems, though similar mechanisms also arise in other path-dependent routing problems. In the TA-TSP, an arc's cost may be modified by trigger arcs, and the incurred cost is determined by the most recently traversed trigger arc affecting it. Cerrone et al. proposed a mixed-integer programming (MIP) formulation that explicitly models trigger relations; however, it becomes prohibitively large even for medium-sized instances. For the TSP part, our approach uses a standard MIP model and classical tour-construction ideas [Wolsey 1998, Cook et al. 2011, Karp 1979].

We revisit TA-TSP from an exact-algorithm perspective. Our approach combines three components: a compact master formulation, a callback-based separation procedure for violations of trigger ordering, and standard TSP rounding heuristics. The central idea is to handle triggers lazily. The master formulation enforces only lightweight consistency constraints, while ordering constraints on visited trigger arcs are imposed a posteriori by separation. This strategy gives a simple yet effective workflow. We solve a restricted

*This work was partially supported by CNPq under grants 313146/2022-5, 404315/2023-2, 404779/2025-5, and 195856/2025-2, by FAPESP under grant 2022/05803-3, and CAPES under grant 88887.980265/2024-00.

exact model, analyze the resulting tours to find invalid trigger orderings, and add cuts when violations are found. The considered trigger set is also expanded progressively, only when the current restricted set is too coarse. Eventually, the master model becomes an exact MIP formulation once the considered triggers reach the full set. Experimentally, these smaller intermediate models substantially improve computational performance.

2. MIP Formulation for the TA-TSP

Let $G = (V, A)$ be a directed graph. Each arc $b \in A$ has a *base cost* $\bar{c}_b$. As in the classical TSP, the goal is to find a Hamiltonian cycle starting and ending at a depot $s \in V$ while minimizing the total cost of the selected arcs. In the TA-TSP, however, arc costs may depend on previously traversed arcs. For each arc $b \in A$, we are given a set $T_b \subseteq A$ of *trigger arcs*. Let $T = \bigcup_b T_b$ denote the multiset of all triggers. Intuitively, traversing an arc in T_b before b may alter the cost of b. Traversal order is defined with respect to the depot s. More formally, if b belongs to the tour and the last arc of T_b visited before b, starting from s, is a, then a is the *active trigger* for b and b is the *target* of a. Then the cost of b changes, and $c_{a,b}$ denotes the resulting cost of traversing b. If no arc in T_b is visited before b, or $T_b = \emptyset$, no trigger is active, and the cost of b remains its base cost $\bar{c}_b$.

Without loss of generality, and to simplify notation, we introduce a *dummy trigger* variable d_b indicating that no explicit trigger is active for b. Such triggers represent the case where no arc in T_b is visited before b, or $T_b = \emptyset$. Dummy triggers allow us to move the model's cost to trigger variables: one variable set describes the tour, while another encodes the cost assignment.

Our model uses the following decision variables:

$$\begin{aligned}
x_a &\in \{0,1\} && \forall a \in A \quad \text{(1 if arc } a \text{ is selected)},\\
y_{a,b} &\in \{0,1\} && \forall b \in A,\ a \in T_b \quad \text{(1 if } a \text{ is the active trigger for } b\text{)},\\
d_b &\in \{0,1\} && \forall b \in A \quad \text{(1 if } b \text{ uses the dummy trigger)}.
\end{aligned}$$

The objective is

$$\min \sum_{b \in A} \bar{c}_b d_b + \sum_{b \in A} \sum_{a \in T_b} c_{a,b} y_{a,b}. \tag{1}$$

The master model includes standard TSP constraints, namely degree and subtour-elimination constraints, which we omit and instead refer to classical MIP formulations of the TSP [Wolsey 1998, Cook et al. 2011]. Subtour-elimination constraints are separated dynamically: whenever the current solution contains a subtour, a violated constraint is added by callback.

Moreover, we include the following basic trigger constraints in the master model:

$$x_a \geq y_{a,b} \qquad \forall b \in A,\ a \in T_b, \tag{2}$$

$$x_b = \sum_{a \in T_b} y_{a,b} + d_b \qquad \forall b \in A. \tag{3}$$

Constraint (2) ensures that a trigger can be active only if its target arc is in the tour. Con-

straint (3) assigned to each selected arc exactly one trigger, either from input or dummy. The master model may activate an incorrect trigger for a target arc because it does not explicitly encode which trigger is last in the tour order. We handle this problem via a callback that generates separating cuts.

Fix an integer solution to the master model, which gives an invalid trigger assignment $y_{c,d} = 1$. Let b be the correct trigger arc that should've been active. If c is *behind* b in the tour, then let P be the path from c to d. Otherwise, c is *after* b, and let P be the path from d to c instead. In both cases, we add the cut

$$\sum_{a \in P} x_a \leq |P| - y_{c,d}. \tag{4}$$

In both cases, we have the same type of cut. Intuitively, once $y_{c,d}$ is selected, the path that certifies the violation cannot remain entirely within the tour. In the implementation, separation is computed based on an incumbent returned by the solver. For each arc a present in such a solution, we identify the last trigger arc that targets a and compare this trigger with the one assigned by the master model. Whenever they differ, an inequality of type (4) is added. Note that the incumbent arcs constitute a valid tour. Thus, we also compute the set of triggers that should've been active to produce a valid TA-TSP solution.

3. Progressive Trigger Set Expansion

Although the master model accounts for all trigger variables, our implementation does not necessarily include them simultaneously. This choice is practically important, since several instances contain hundreds of thousands of triggers, making the full model prohibitively large from the start. Instead, we include trigger variables only when required.

Namely, for each arc, triggers are sorted in increasing order of cost and considered sequentially. Starting from a small prefix, the number of triggers per arc is doubled at each step until the full set is reached. This geometric expansion keeps the initial model much smaller while exposing cheaper trigger candidates first. The rule also has a simple theoretical justification. For every arc b, let $\tau_b^1, \tau_b^2, \ldots$ be its triggers sorted by nondecreasing cost, and let $T_b^{(k)}$ be the prefix after the k-th doubling step. Then each restricted family $T^{(k)} = \cup_b T_b^{(k)}$ is *cost-closed*: if a trigger for b is present, then every cheaper trigger for the same target is also present.

Proposition. Let T' be a cost-closed trigger set and let P^* be optimal for the TA-TSP restricted to T'. If the trigger assignment used by P^* remains valid under the full trigger set T, then P^* is also optimal for the full problem.

Proof. Consider any solution P. When enlarging the trigger set from T' to T, additional triggers may become available for each arc of P. Since T' is cost-closed, every newly revealed trigger has a cost at least as high as that of an already available trigger. Thus $\mathrm{cost}(P \mid T') \leq \mathrm{cost}(P \mid T)$. By optimality of P^* in the restricted problem, $\mathrm{cost}(P^* \mid T')$ $\leq \mathrm{cost}(P \mid T') \leq \mathrm{cost}(P \mid T)$ for all P. If the trigger assignment of P^* is unchanged under the full set T, then $\mathrm{cost}(P^* \mid T') = \mathrm{cost}(P^* \mid T)$, and hence P^* is optimal for the input instance. □

After each expansion step, the callback cuts continue to correct invalid trigger assignments in the enlarged model. Thus, the overall method alternates among solving a compactified master model, expanding the trigger set, and separating violated trigger assignments. Each iteration defines a restricted TA-TSP problem. After expansion, the next model is warm-started from the best incumbent and valid cuts already found, so this previously accumulated information can still guide the search.

4. Rounding Heuristic

The assignment-and-patching method for the Asymmetric TSP (ATSP) inspires our rounding heuristic. Starting from the fractional arc values of the current LP solution, we build a directed cycle-cover-like structure on the support graph and then patch the resulting subtours into a single Hamiltonian tour in the spirit of Karp's classical patching method [Karp 1979]. The resulting tour is further improved by a lightweight local search based on swap and relocate moves. Every candidate is evaluated with the exact trigger-aware objective of the TA-TSP.

5. Computational Results

We evaluated our solver and Cerrone's on the TA-TSP benchmarks `grf1-21` and `grf101-134` [Cerrone et al. 2025] using Gurobi 12.0. The tests were executed on an Apple M2 Pro with 10 threads and a 1-hour time limit. Due to space constraints, we report representative instances presented in Table 1. To ensure the Cerrone formulation finds an incumbent, we warm-start it with a surrogate ATSP tour (using minimum trigger costs) and refine it via swap and relocate local search, considering the full objective. For `grf1-21`, we solved 9/21 instances to optimality and 10 within a 5% gap (7.10% average final gap). For `grf101-134`, we optimally solved `grf101` and `grf103`, with 31/34 instances finishing within a 10% gap (4.59% average final gap).

6. Conclusion

We presented an exact solution framework for the TA-TSP based on a compact trigger formulation, with callback-based separation for subtour elimination and trigger consistency, and a progressive trigger expansion scheme. This approach enables handling instances with very large trigger sets while keeping the master problem manageable. Preliminary results show that the method can prove optimality on smaller instances and remains competitive on larger ones. Future work includes a systematic comparison with existing formulations and improvements to trigger-violation separation.

Table 1. Selected preliminary results.

				Our			Cerrone et al.		
Instance	$\|V\|$	$\|A\|$	$\|T\|$	LB	UB	Gap (%)	LB	UB	Gap (%)
`grf2`	20	300	5.7k	**44.53**	**44.53**	**0.00**	**44.53**	**44.53**	**0.00**
`grf12`	40	800	15.7k	**62.36**	**62.36**	**0.00**	60.75	62.72	3.14
`grf15`	60	2700	4.53M	**234.38**	**294.46**	**20.40**	-	*	-
`grf18`	60	1800	53.5k	**77.43**	**78.65**	**1.55**	72.40	91.16	20.58
`grf101`	18	90	1.1k	**16.96**	**16.96**	**0.00**	**16.96**	**16.96**	**0.00**
`grf103`	22	110	1.8k	**20.54**	**20.54**	**0.00**	**20.54**	**20.54**	**0.00**
`grf113`	58	464	23.6k	**52.45**	**54.81**	**4.31**	52.35	*	4.49
`grf128`	122	1342	151.9k	**109.95**	**117.08**	**6.09**	**109.95**	*	**6.09**
`grf134`	142	1562	208.0k	**127.95**	**137.08**	**6.66**	**127.95**	*	**6.66**

Asterisk indicates no improvement over the warm start.

References

Cerrone, C., Truvolo, M., Dragone, R., and Battaglia, R. (2025). The Trigger Arc TSP: Optimise the Picking Process in Warehouses with Compactable Storage Systems. In *Decision Sciences*. Springer.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., and Schrijver, A. (2011). *Combinatorial Optimization*. Wiley.

Karp, R. M. (1979). A Patching Algorithm for the Nonsymmetric Traveling-Salesman Problem. *SIAM Journal on Computing*, 8(4):561–573.

Wolsey, L. A. (1998). *Integer Programming*. Wiley-Interscience, New York.