FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Diplomarbeitspräsentation

# Implementing Variations of the Traveling Salesperson Problem in a Declarative Dynamic Programming Environment


Marius Liviu Moldovan

Masterstudium:
Computational Intelligence

Technische Universität Wien
Institut für Informationssysteme
Arbeitsbereich: Datenbanken und Artificial Intelligence
Betreuer: Univ.Prof. Dipl.-Ing. Dr.techn. Stefan Woltran
Mitwirkung: Projektass.(FWF) Dipl.-Ing. Michael Abseher


dbai
Database and Artificial Intelligence Group

## Context

- The *Traveling Salesperson Problem without Repetitions* (TSP-NR) is an NP-hard combinatorial optimization problem defined as follows: Given the distances between n cities, return the cheapest tour that visits each of them once.
- The *Traveling Salesperson Problem with Repetitions* (TSP-R) is a generalization of the former. It stipulates that for each city the range between minimum and maximum allowed number of visits can be specified (otherwise the default value is exactly one visit).

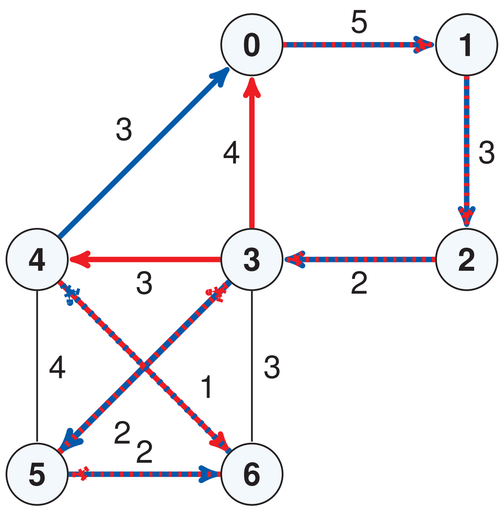


Figure 1: Graph with optimal solution for TSP-NR (blue) and TSP-R, where node **3** must be visited exactly twice and all others at least once (red).

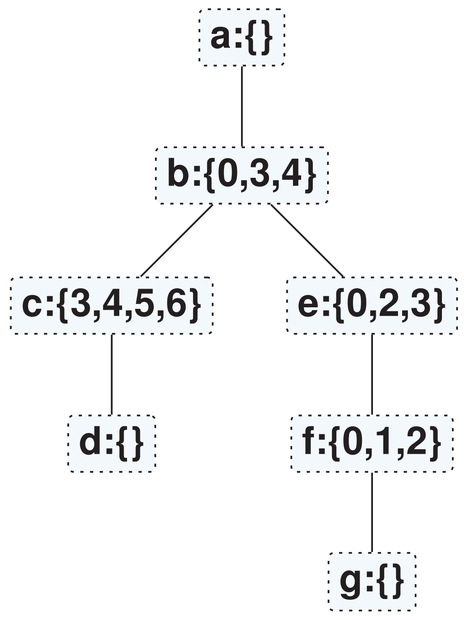


Figure 2: A tree decomposition for the shown graph.

- *Dynamic Programming* (DP) is a mathematical and computer engineering method used mainly for solving discrete optimization problems by first solving subproblems and storing their solutions and then combining the latter into a complete solution.
- *Answer Set Programming* (ASP) is a form of fact-driven declarative programming that is based on the semantics of stable models. ASP is highly maintainable and flexible, however often with the price of a higher running time and memory consumption.
- The motivation for our work derives from the need to easily prototype dynamic programming algorithms for relevant problems, such as the TSP, without losing on efficiency.

## Methodology

- The *D-FLAT* System, developed by the **dbai** Group, both promises flexibility and handles large amounts of data efficiently.
- It decomposes the input graph into a tree, such that each edge and each vertex of the graph must belong to at least one node of the decomposition. Further, if a vertex belongs to two different nodes, it must belong to all nodes on the path between the former.
- At every node, D-FLAT executes an ASP program, which is tailored to the problem to be solved, stores the results for each node in a table and finally recomposes the optimal solutions by looking up all tables.

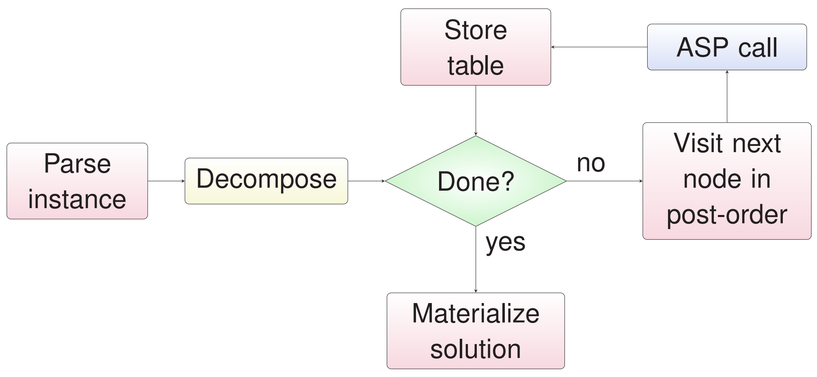


Figure 3: Control flow in D-FLAT, adapted from Figure 4 in [1].

- The goals were to:
  - Develop a dynamic programming concept for the TSP.
  - Prove the flexibility and efficiency of D-FLAT by encoding the TSP-NR and TSP-R based on the developed concept.
  - Show that they are more performant than state-of-the-art ASP encodings on instances with small treewidth, the latter denoting a limited cyclicity of a graph.


References: [1] M. Abseher, B. Bliem, G. Charwat, F. Dusberger, M. Hecher, and S. Woltran. D-FLAT: Progress report. Technical Report DBAI-TR-2014-86, Vienna University of Technology, 2014.


## Solution

- In order to comply to the principle of dynamic programming we always work only on vertices that are in the current node.
- When traversing the tree, we do the following for every tree node using ASP:
  - For every newly introduced vertex (such as vertex **3** in node **e** of the tree decomposition in Figure 2) we guess whether to select the edges that lead to an adjacent current vertex (**0** and **2**) to be part of the tour.
  - If in a join node (here node **b**) the edge selection on the left branch does not correspond to the one on the right, we discard the solution candidate.
  - We keep a counter for the selected adjacent edges of each current vertex.
  - When a vertex was removed (e.g. vertex **1** in node **e**), we check the counter. If it is different from 2 for the TSP-NR, or not in the specified range times 2 for the TSP-R, we eliminate the solution candidate.
  - We memorize for each pair of current vertices (e.g. **0**,**2**,**3** in node **e**) if it is connected by a path.
  - If at removal, a vertex is not connected to any other current vertex we cannot obtain a connected tour and the candidate is discarded.

## Results

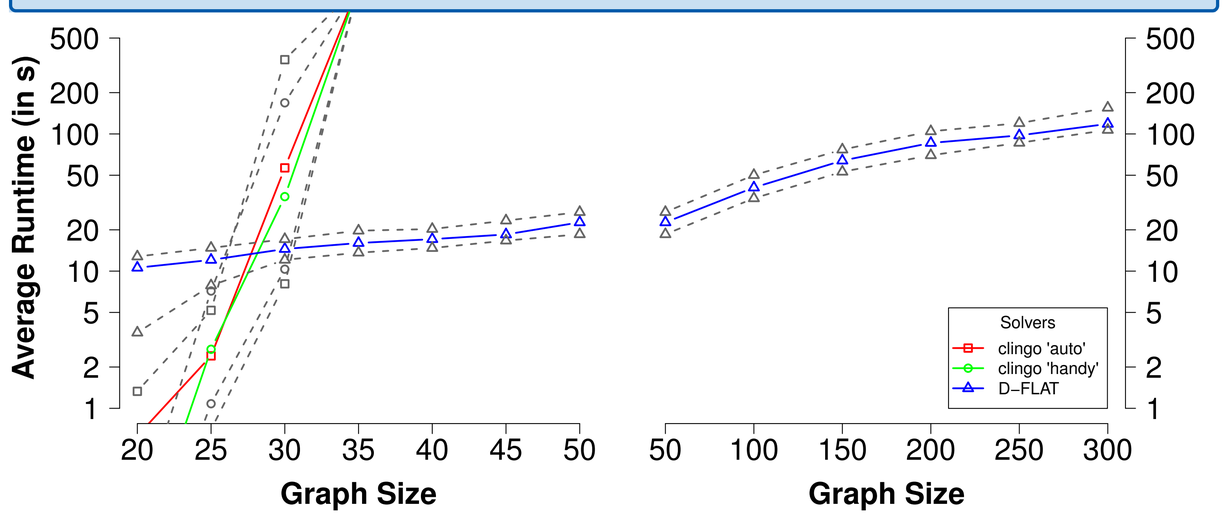


Figure 4: Minimum, average and maximum runtime of TSP-NR encoding for D-FLAT vs. standard ASP TSP-NR encoding with clingo (two configurations) on 8-connected full grids of treewidth 4.

- For treewidths 2, 3 and 4, our D-FLAT encoding of the TSP-NR can solve all our generated full grid instances in less than 10 minutes up to a graph size of 6000, 2100 and 900 vertices, whereas the ASP encoding only terminates up to a graph size of 450, 45 and 30 in the same time span, respectively.
- The memory consumption is also lower with D-FLAT for treewidths 2 & 3.

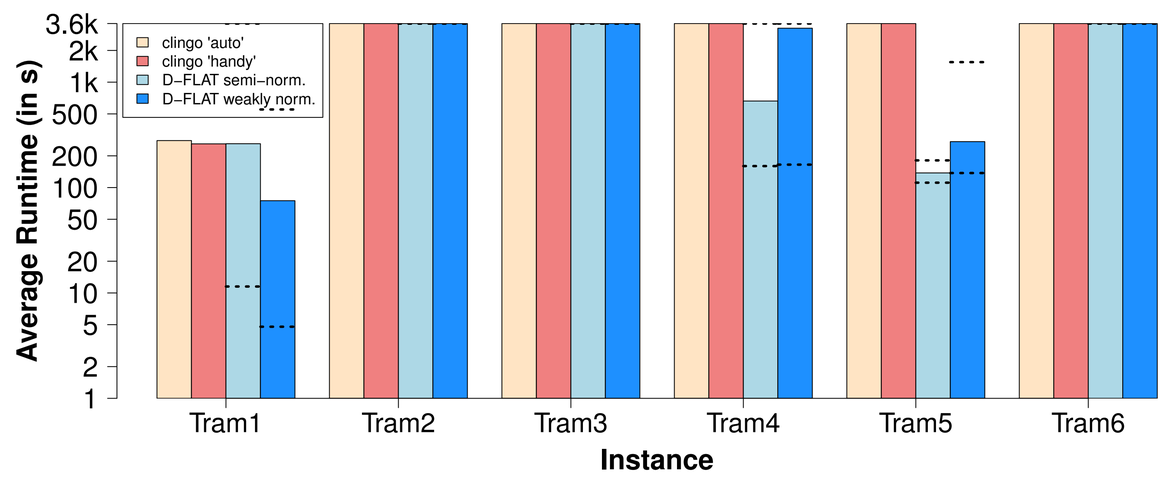


Figure 5: Average runtime of TSP-R encoding for D-FLAT vs. standard ASP TSP-NR encoding with clingo (two configurations each) on real world instances based on Vienna's tramway system (402 stations, treewidth 6). Dashed lines illustrate the minimum and maximum values.

| Inst. | Sat. | To Visit | `maxVisits` |
|---|---|---|---|
| Tram1 | | 4 | 10 |
| Tram2 | ✓ | 4 | 10 |
| Tram3 | ✓ | 4 | 1 |
| Tram4 | ✓ | 15 | 1 |
| Tram5 | ✓ | 24 | 1 |
| Tram6 | ✓ | 24 | 10 |

Figure 6: Instance name, satisfiability, number of vertices that must be visited once and maximum number of visits for the remaining vertices (the minimum number being 0 for all), for the Viennese tramway system instances.

- The following holds for instances based on the same graph: The more vertices must be visited, the faster our encoding becomes.
- One condition must be met for success: The number of maximum visits for the remaining vertices must be restricted.


Contact: marius.moldovan@lasting.ro