Presentation Notes

Slide 1 – Introduction

- Hi everyone, I am Lukas Koller and today I am presenting my IDP project *Formalizing Theory of Approximating the Traveling-Salesman Problem.*

Slide 2 – Traveling-Salesman Problem

- The Traveling-Salesman Problem (TSP) is a very well known combinatorial optimization problem, with application in many different areas, such as operation research and logistics, planning and scheduling, or the design of printed circuit boards.
- The TSP describes the task of finding a shortest Hamiltonian cycle in a given undirected graph with edge weights. A Hamiltonian cycle is a cycle that visits every vertex of the graph exactly once.
- On the left of this slide you can see a graph and an optimal Hamiltonian cycle.
- The decision problem of the TSP is NP-hard. This means that, unless P=NP, there is no polynomial-time algorithm that solves the TSP.

Slide 3 – Approximation Algorithms for TSP

- A typical approach when dealing with NP-hard optimization problems is to use an approximation algorithm.
- An approximation algorithm computes an approximate solution in polynomial time.
- The approximation ratio of an approximation algorithm bounds the distance between the approximate solution and the optimal solution.
- For a minimization problem like TSP, an approximation ratio of 2 means that any approximate solution has at most a cost of double the cost of the optimal solution.
- Unfortunately, there is bad news for the TSP. There is no constant-factor approximation algorithm for the TSP, unless P=NP. This means, the TSP cannot efficiently approximated. Therefore, the TSP is simplified by making assumptions for the given graph. The simplified TSP is called Metric TSP. We assume the given graph is complete and the edge weights satisfy the triangle-inequality. Both these assumptions hold if we want to find the shortest route for an idealized scenario where a mail-carrier delivers mail. The mail-carrier can go from any other address to any other address without stopping. The triangle-inequality holds, because when the mail-carrier makes and intermediate stop, the length of the route does not decrease.
- The decision problem of the Metric TSP is still NP-hard.

Slide 4 – Overview

- For my IDP I have formalized and verified two approximation algorithms for the Metric TSP: the DoubleTree and the Christofides-Serdyukov algorithm. Secondly, I have formalized an L-reduction from the Minimum Vertex-Cover Problem (VCP4) with maximum degree of 4 to the Metric TSP. The L-reduction proves the MaxSNP-hardness of the Metric TSP. A consequence of the MasSNP-hardness is the non-existence of a polynomial-time approximation scheme. A polynomial-time approximation scheme is essentially an approximation algorithm with an additional parameter that influences the approximation ratio.
- For the remainder of my presentation I will explain the main ideas and my formalization approach for the Christofides-Serdyukov algorithm and the L-reduction.

Slide 5 – Approximation Algorithms for Metric TSP

- The following lemma is a central idea for the Christofides-Serdyukov algorithm.
- The lemma states: given a Eulerian graph that spans the vertices of the graph we are given from the Metric TSP, we can in polynomial time compute a Hamiltonian cycle for the Metric TSP. The total weight of the Hamiltonian cycle is at most the total weight of all edges in the Eulerian graph.
- A graph is Eulerian if every vertex has even degree.
- The Hamiltonian cycle is computed by first computing an Eulerian tour and then removing all duplicate vertices. An Eulerian tour is a cycle in a graph that uses every edge exactly one. Because the graph is complete and the edge weights satisfy the triangle-inequality, removing duplicate vertices from the Eulerian tour results in Hamiltonian cycle that has less total weight than the Eulerian tour.
- With this lemma, we only have to construct an Eulerian graph with the smallest total weight to approximate the Metric TSP.

Slide 6 – Christofides-Serdyukov Algorithm

- The Christofides-Serdyukov algorithm is the best known approximation algorithm for the Metric TSP, with an approximation ratio of 1.5.
- The Christofides-Serdyukov algorithm start by computing a minimum spanning-tree (MST).
- The MST is not a Eulerian graph. Therefore, in the second step the Christofides-Serdyukov algorithm computes a minimum perfect-matching between the vertices with odd degree in the MST. The union of the MST and the matching is a Eulerian graph.
- Now, simply the lemma stated on the previous slide is applied: first a Eulerian tour is computed. For example .... Finally, the duplicate vertices in the Eulerian tour are removed to obtain a Hamiltonian cycle.
- The total weight of a MST is at most the total weight of the optimal Hamiltonian cycle. This is because removing any edge from the optimal Hamiltonian cycle results in a spanning-tree. On the other hand, the total weight of the matching is at most half of the total weight of the optimal Hamiltonian cycle. Therefore, approximation ratio of the Christofides-Serdyukov algorithm is 1.5.

Slide 7 – Formalization of the Christofides-Serdyukov Algorithm

- I have formalized the Christofides-Serdyukov algorithm in Isabelle/HOL.
- Therefore, I build on the existing formalization of undirected graphs by Mohammad Abdulaziz.
- The Christofides-Serdyukov algorithm depends on there other algorithms: an algorithm for minimum spanning-tree, minimum perfect-matching, and Eulerian tour. I have proven the feasibility and the approximation ratio of the Christofides-Serdyukov algorithm.
- With Hoare-Logic, I have also refined the definition of the Christofides-Serdyukov algorithm to a WHILE-program.

Slide 8 – L-Reduction (Linear Reduction)

- For the second part of my IDP, I have formalized an L-reduction from the VCP4 to the Metric TSP. An L-reduction is a special type of reduction that is used to prove the MaxSNP-hardness of optimization problems.
- An L-reduction from the problem A to the problem B consists of two functions f and g and two constants alpha and beta. The function f and g are computable in polynomial time.
- The function f maps an instance of the problem A to and instance of the problem B.
- Whereas, the function g maps a feasible solution of the constructed instance to a feasible solution to the original instance of the problem A.
- The costs of the feasible solutions and the optimal solutions for the constructed instance have to satisfy two linear inequalities. I won't go into further detail about the inequalities, but one thing to note is that given an optimal solution to the constructed instance f(xA), the function g has to compute an optimal solution for the original instance xA.

Slide 9 – L-Reduction for Metric TSP I

- The VCP4 is L-reduced to the Metric TSP.
- For the VCP4, we are given an undirected graph where every vertex has a degree of at most 4. The bound on the degree is just needed to prove the linear inequalities. The task for the VCP4 is to find a vertex cover with minimum cardinality. A vertex cover is a subset of vertices such that every edge of the graph is incident to at least one vertex in the vertex cover.
- For the Metric TSP on the other hand, we have to construct a complete graph with edge weight such that the edge weights satisfy the triangle inequality.
- The function f maps an instance of the VCP4 to an instance of the Metric TSP. I illustrate this construction with an example. Assume we are given the graph that is depicted on this slide. The complete graph that is constructed by the function f is the complete graph over individual subgraphs, one for every edge.
- Every corner of a subgraph corresponds to a vertex of the original graph. The subgraphs have the special property that for there is no Hamiltonian path for a subgraph that starts and ends at corners that correspond to different vertices. Therefore, the start and end corner of a Hamiltonian path for a subgraph are used identify a covering vertex for the corresponding edge.
- Corners of different subgraphs that correspond to the same vertex are connected by cheap edges, with a weight of 4.

- All the edge weights in a subgraph are 1. The weight between two vertices in a subgraph that are not connected by an edge is the minimum distance between the vertices in the subgraph.
- All remaining edges are expensive edges with a weight of 5.
- In the example, we connect the corners that correspond to the vertex v with cheap edges. To obtain a Hamiltonian cycle we have to add to expensive edges.
- The number of expensive edges exactly corresponds to the cardinality of the corresponding vertex cover.
- This is the construction for the function f.

Slide 10 – L-Reduction for Metric TSP II

- The function g now maps a feasible solution for the Metric TSP instance to a feasible solution of VCP4 instance.
- Assume we are given an arbitrary Hamiltonian cycle (feasible solution) for the for the Metric TSP instance. We construct a vertex cover from the Hamiltonian cycle.
- As illustrated on this slide, the Hamiltonian cycle might not contain a Hamiltonian path for every subgraph. We transform the Hamiltonian cycle such that it contains a Hamiltonian path for every subgraph. For every subgraph we replace the vertices in the Hamiltonian cycle a Hamiltonian path for the subgraph. Here we have to make sure to keep outgoing edges to corners of other subgraph. This is necessary to ensure that the total weight of the transformed Hamiltonian cycle is at most the total weight of the original Hamiltonian cycle. This is needed to prove the linear inequalities.
- From the transformed Hamiltonian cycle we can identify the covering vertices an construct a vertex cover.

Slide 11 – Formalization of L-Reduction for Metric TSP

- For the formalization of the L-reduction I used a locale that provides an abstract executable graph representation. The graph representation is based on an adjacency map.
- The locale assumes fold-function that are used to computations and iterate over the graph.
- I have implemented the functions f and g with the fold-functions. Within the locale I prove the feasibility and the linear inequalities.
- I finally instantiate the locale with a concrete implementation of an adjacency map to obtain concrete implementations for the functions f and g.
- During my formalization I discovered two issues with the proof that is presented by Korte and Vygen. The first issue is that some properties that are used for the proof of the linear inequalities don't hold if the optimal vertex cover has a cardinality of 1. The L-reduction still works, but an explicit proof for this case is necessary. I have circumvent this problem by simply excluding this case with an additional assumption.
- The other issue is that definition of the function g by Korte and Vygen does not cover all possible cases for Hamiltonian paths of the subgraphs. There are Hamiltonian paths that are not considered in the definition. I have fixed this issue by extending their definition with the necessary case distinction.

Slide 12 – Conclusion and Future Work

- For my IDP I have formalized and verified two approximation algorithms for the Metric TSP.
- I have formalized an L-reduction from VCP4 to Metric TSP, which proves the MaxSNP-hardness of the Metric TSP.
- Future work includes the a formalization of the algorithms required by the Christofides-Serdyukov algorithm. I have started and almost finished an adaptor to an existing formalization of Prim's algorithm for minimum spanning-trees. For minimum perfect-matching and Eulerian tour Korte and Vygen provide algorithms which can be formalized.
- The Christofides-Serdyukov algorithm used a Eulerian multigraph. A formalization of an encoding of multigraphs with simple graphs would be nice.
- Finally, I have not proven the polynomial running time of the reduction function f and g. An approach would be to refine their implementations to IMP- and prove the running times of the refined definitions.