TSP Neighbourhood Reduction with POPMUSIC

Éric D. Taillard[1]

HEIG-VD
Rte de Cheseaux 1, CP 521, CH-1401 Yverdon-les-Bains, Suisse
eric.taillard@heig-vd.ch

Abstract

A key point for implementing a fast and efficient local search is to use a neighbourhood of limited size containing all the pertinent moves. For the travelling salesman problem, the most efficient neighbourhoods are based on Lin-Kernighan moves. In order to speed-up the computation, only a subset of moves are evaluated. This article propose a method with a low algorithmic complexity for generating a limited subset of pertinent edges that must be used in the solution tour. Combined with a state-of-the art local search, this technique is able to produce solutions of very high quality to very large problem instances.

1 Introduction

The travelling salesman problem (TSP) is certainly the most studied NP-hard combinatorial optimisation problem. Now, we are able to exactly solve instances up to several thousands of cities and to find solutions for instances with millions of cities at a fraction of the percent above the optimum [2, 3]. One of the best heuristic source codes for the TSP is due to [4]. It is referred below as LKH.

2 Neighbourhood limitation

For limiting the computational time, it is necessary to limit the number of edges that can be used in solution tours. Various techniques have been proposed for this purpose: In the case of Euclidean problem instances, a Delaunay triangulation can be computed (in $O(n \log n)$) and the moves considered are those only containing the edges of the sub-graph induced by the triangulation. This principle can be extended to general problem instances by using a technique based on 1-trees. Provided that the average vertex degree in the sub-graph is limited (typically to a value from 5 to 10), finding a local optimum relative to Lin-Kernighan (or k-opt) moves can be performed with an empirical algorithmic complexity that is less than quadratic.

However, computing the 1-trees has a quadratic complexity. Therefore, the LKH implementation requires a computational time that grows quadratically for general problem instances, even if the local search is faster. This is illustrated in Figure 1.

2.1 Low complexity sub-graph generation

For limiting the preprocessing time, we propose to build few dozens of TSP tours with a low complexity, a randomised heuristic. The union of the tours defines a sub-graph whose edges are the only ones that can be used by the local search moves. This technique was called *tour merging* by [1].

The randomised procedure must produce TSP tours of adequate quality with a low algorithmic complexity. We propose to use the POPMUSIC [6] template for creating this randomised procedure. For getting an initial solution that is convenient for POPMUSIC, Algorithm 1 is used. Notice that the last part of this algorithm can be seen as a fast POPMUSIC since n/a sub-paths are optimised.

The main steps of this algorithm are illustrated in Figure 2.

The solution produced by Algorithm 1 is improved by a standard POPMUSIC where all sub-path of r consecutive cities are optimised with a Lin-Kernighan neighbourhood. The POPMUSIC optimisation improves the quality of the candidate edges since it removes the inappropriate edges belonging to 2 consecutive sub-paths obtained at step 9.

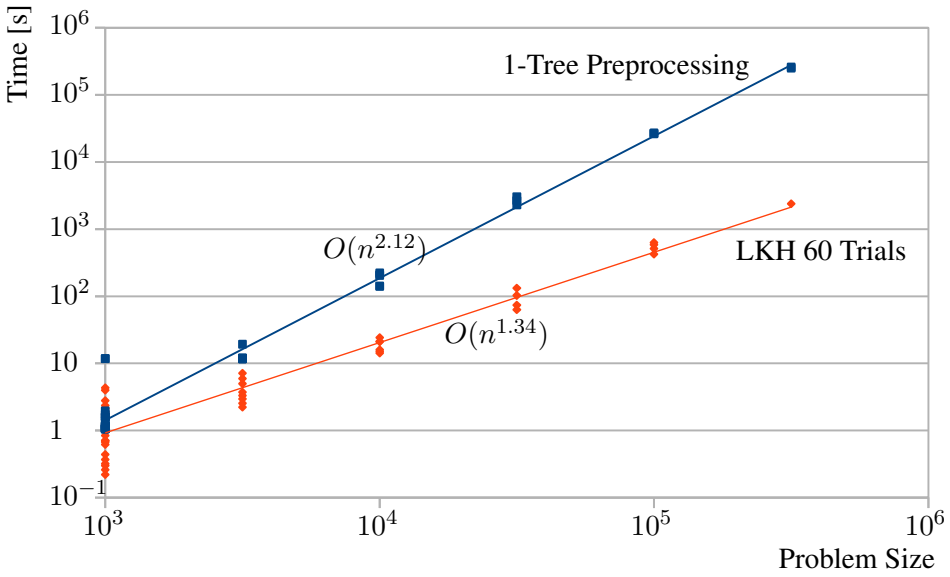

Figure 1: Computational time of LKH program for uniform (E) and clustered (C) DIMACS instances, with candidate edge list found with 1-trees and sub-gradient optimisation, 60 trials and a single run. The lines corresponds to interpolation polynomials.

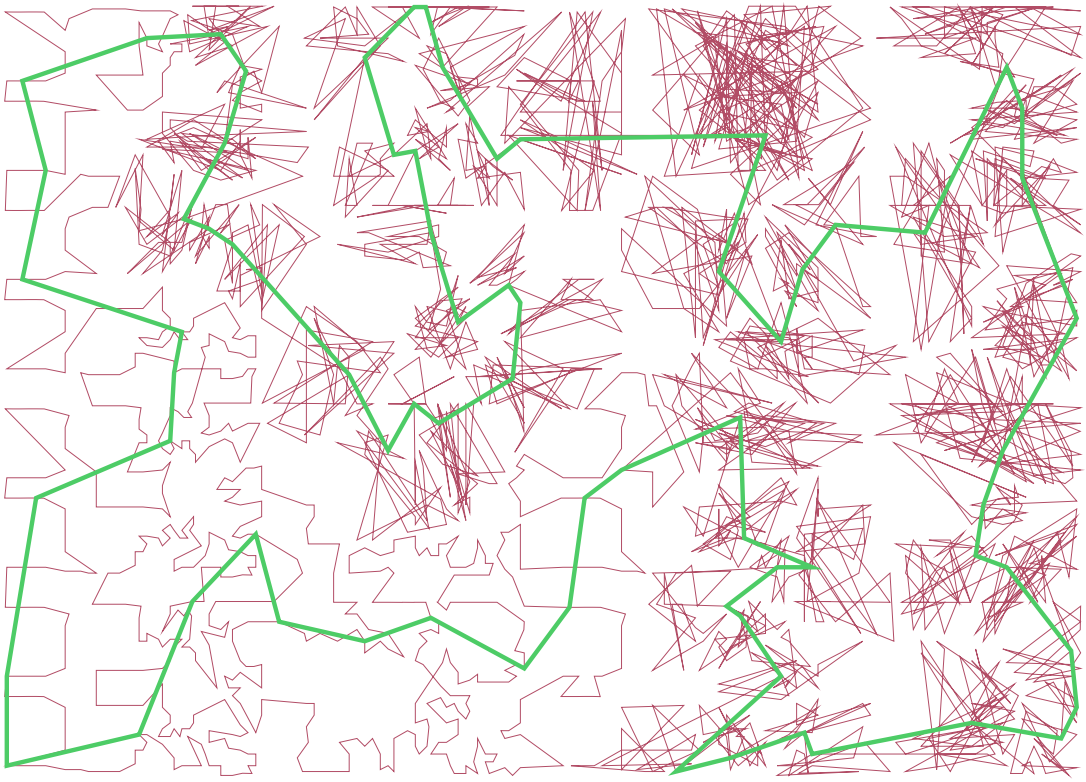

Figure 2: Building an initial solution adapted for POPMUSIC. First, a tour is found on a sample of cities (bold line). Then, the remaining cities are inserted after the closest of the sample, creating clusters of cities. Finally sub-path including the cities assigned to 2 consecutive clusters are optimised. The figure shows the situation when the cluster optimisation is partially performed.

Algorithm 1: Generating a feasible TSP tour adapted for POPMUSIC template.

Data: n cities, distance function $d(i, j)$ between cities i and j, parameter $0 < a < n$
Result: TSP tour T

```
Uniformly select a random sample S of a cities;
Build a LK-optimal tour T_s on S;
T = T_s
for each city c ∉ S (in arbitrary order) do
    c_c = argmin(d(s, c)), (s ∈ S);
    Insert c just after c_c in T;
for each city c ∈ T_s (in the order of appearance in T_s) do
    Let n_c be the number of cities inserted at the previous step after c and after the city next to c in T_s;
    Optimise a sub-path of n_c cities in T, starting from c, with a 2-opt local search
```

Problem size	Best known	Computational Time			%Excess	
		Sub-graph	Total	LKH	Proposed	LKH
1000	23101545.4	2.981	3.921	1.701	0.036	0.016
3162	40519926	9.406	17.54	16.4	0.062	0.022
10000	71865826	29.7	92.2	226	0.11	0.05
10000	72031630	29.9	82.3	225	0.08	0.03
10000	71822483	29.6	87.0	237	0.12	0.03
31623	127282138	95.3	427	2396	0.19	0.06
31623	126647285	95.2	470	2427	0.83	0.73
100000	224330692	325	1850	26860	0.88	0.76
100000	225654639	326	1805	27446	0.24	0.11
316228	401301206	1234	7821	262090	0.26	0.15
1000000	713187688	5250	31061	—	0.27	—
3162278	1267318198	27770	129866	—	0.28	—
10000000	2253088000	208195	611402	—	0.28	—

Table 1: Results for uniform (E) DIMACS instances.

Using a very basic Lin-Kernighan local search procedure with an empirical complexity of $O(n^{2.78})$, a 2-opt local search with empirical complexity $O(n^{2.29})$ and choosing $a = 1.5 \cdot n^{0.56}$, the empirical complexity of Algorithm 1 is about $O(n^{1.6})$. Applying POPMUSIC with $r = 50$ to the solution so obtained takes a time proportional to the number of cities.

3 Numerical results

On Euclidean DIMACS problem instances, POPMUSIC applied to an initial solution obtained with Algorithm 1 produces solutions that are 5.7% to 12.8% above best solution values known [2]. Although POPMUSIC produces not tremendously good solutions, the union of 20 solutions obtained by this technique generates an candidate set of edges of excellent quality for a local search. Table 1 provides the computational time (seconds on Intel i7 930 2.8GHz) and the solution quality of LKH on DIMACS uniform (E) instances. This table provides the computational time for producing 20 solutions with POPMUSIC and the total time of the method when LKH works with the candidate edges obtained with the union of 20 tours. Then, LKH is run with standard parameters (not exploiting the Euclidean property of the instances, 1 run with 60 trials). Finally, Table 1 gives the quality of solutions produced by our method and by standard LKH. For problems of size 1000 and 3162, the results are averaged for 10 (respectively:

Problem size	Best known	Computational Time			%Excess	
		Sub-graph	Total	LKH	Proposed	LKH
1000	11174460.5	3.24	4.18	4.13	0.11	0.10
3162	19147233.6	10.5	20.3	15.1	0.68	1.50
10000	33001034	33.1	85.4	68	0.72	3.31
10000	33186248	33.4	76.8	73.6	0.86	1.42
10000	33155424	33.2	71.1	73.9	0.25	0.42
31623	59545390	108	230	469	0.49	3.57
31623	59293266	107	264	455	0.79	2.29
100000	104617752	368	975	3457	1.16	4.56
100000	105390777	368	947	3467	0.85	7.47
316228	186870839	1372	3495	32018	0.98	9.44

Table 2: Results for clustered (C) DIMACS instances.

5) instances.

Table 2 provides the same information for clustered (C) DIMACS instances. In this table, LKH was run with additional parameter INITIAL_PERIOD = 1000 for speeding-up the preprocessing time. We see in these tables that our method is much faster for large instances and produces better solutions for clustered instances.

For uniformly generated problem instances in hypercubes of dimension 2, 3 and 4 with toroidal distances, whose size ranges from half a million to more than 2 million cities, we have obtained average solution lengths 0.25%, 0.11% and 0.06% above the predicted optimum [5].

References

[1] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, USA, 2007.

[2] Keld Helsgaun. Best known solutions to Dimacs TSP instances. Last updated: October 6, 2014.

[3] Keld Helsgaun. General k-opt submoves for the lin-kernighan tsp heuristic. *Mathematical Programming Computation*, 1, 2009.

[4] Keld Helsgaun. Helsgaun's implementation of Lin-Kernighan, 2016. Version LKH-2.0.7.

[5] David S. Johnson, Lyle A. McGeoch, and E. E. Rothberg. Asymptotic experimental analysis for the Held-Karp traveling salesman bound. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 341–350, 1996.

[6] Éric D. Taillard and Stefan Voss. POPMUSIC: Partial optimization metaheuristic under special intensification conditions. In Celso Ribeiro and Pierre Hansen, editors, *Essays and surveys in metaheuristics*, pages 613–629. Kluwer Academic Publishers, 2001.