

Applying Lagrangian relaxation to the Traveling Salesman Problem

Matěj Jirka

Thursday 12:45

Open Informatics - Computer engineering
jirkamat@fel.cvut.cz

Abstract—This electronic document describes application of Lagrangian relaxation to the Traveling Salesman Problem. It contains also algorithm details and results of nontrivial data on input.

I. ASSIGNMENT

A. Problem Statement

Traveling Salesman Problem is difficult discrete optimization problem. The problem can be described as follows. A Salesman wants to visit number of cities and return to his home but he is old and lazy so he has to find the shortest path through the cities and visit each city only once. The Salesman has a map and he also knows all distances between cities so he can decide which path is the shortest but as the number of cities increases this becomes difficult.

Lagrangian relaxation is an approximation of difficult problem by relaxation method to simpler problem. In some cases we can get optimal solution of the relaxed problem. General idea of LR is to divide problem constraints into two sets “hard” constraints and “easy” constraints. It is common to get rid of “hard” constraints and put them into objective function (heuristic), assigned with weights (the Lagrangian multiplier). Each weight (penalty) is added to solution that does not satisfy the particular constraint.

B. Problem Categorization

The Traveling Salesman Problem is an NP-hard problem. Instead of solving NP-hard problem we apply Lagrange relaxation and solve relaxation of TSP.

II. RELATED WORKS

Each one of us ever solved Traveling Salesman Problem. Maybe we do not perceive it but it is true. It is solved by many companies, factories and organizations. Solution of this problem helps us to save kilometers, fuel, money, energy and many other resources. But the way to the optimal solution is long and difficult as the number of constraints increases. Therefore compulsion to speed up and simplify solving of this problem is high. It's not easy at all.

The easiest TSP solver implementation goes through all options and selects the optimal one. It is also the most unwanted one. Better is to use a heuristic to solve TSP [3]. By using heuristics we reduce state space. But it is not guaranteed that we obtain the optimal solution. On the other hand it provides solution many times faster.[5]

One way to get good solution is also through genetic algorithms. Genetic algorithms bases on trial-and-error procedure. It tries to crossover and mutate generation by generation and select candidates from each generation to be representative in next generation. Solving TSP by genetic algorithms approximate to one solution.[2]

III. PROBLEM SOLUTION

A. Design

I will solve TSP by Lagrangian relaxation and formulated as an integer linear programme. TSP can be formulated:

Minimize:

$$z = \sum_{e \in E} c(e)x_e$$

Subject to:

1. $\sum_{e \text{ entering } v} x_e = 1 \text{ for all } v \in V$
2. $\sum_{e \text{ leaving } v} x_e = 1 \text{ for all } v \in V$
3. $u_{v_{src}} - u_{v_{dst}} + |V| * x_e \leq |V| - 1 \text{ for all } v_{dst} \neq v_{src} \in V$
4. $x_e \in \{0,1\} \text{ for all } e \in E$

First condition provides solution with only one entering edge to every node. Second condition is similar to the first one. It provides solution with only one leaving edge from every node. So every node has exactly two edges, one for enter and one for leave. But that is not all. We want solution with exactly one cycle. And this ensures the third condition. We can say that u_i and u_j are timestamps. For each pair of adjacent nodes is true that source node has timestamp one less than destination node. Last condition says that x_e is binary indicator if edge e is selected to solution.

Therefore the third condition is hard to preserve we can remove it but we have to add penalization to solution. Reformulated by Lagrangian relaxation to:

Minimize:

$$Z_{LLBP} = \sum_{e \in E} c(e)x_e + \mu(u_{v_{src}} - u_{v_{dst}} + |V| * x_e - (|V| - 1)) \text{ it is equivalent to } cx + \mu(Ax - b)$$

Subject to:

1. $\sum_{e \text{ entering } v} x_e = 1 \text{ for all } v \in V$
2. $\sum_{e \text{ leaving } v} x_e = 1 \text{ for all } v \in V$
3. $x_e \in \{0,1\} \text{ for all } e \in E$

Now we have only three simple conditions and we can solve TSP. There is only one problem. How to find vector μ ? Vector μ is $\mu \in \mathbb{R}_{\geq 0}^{mA}$ positive vector. Its components are called *Lagrangian multipliers*. Next step is to find vector μ^* , that produce the best greatest lower bound (Z_{LLBP}) of z . This finding is called *Lagrangian Dual*. To find this vector we can use *Subgradient method*[4]:

1. Set μ, k to 0, $\pi = 2$
2. Compute $Z_{LLBP}(\mu_k)$ and a vector $x_k \in X$
3. Compute subgradients $\delta = b - Ax$
4. Compute step size $\Delta = \frac{\pi(OrigProblemSolution - Z_{LLBP})}{\sum_{1 \leq i \leq mA} \delta_i^2}$
5. Update Lagrangian mult. $\mu_i = \max(0, \mu_i + \Delta * \delta_i) \forall 1 \leq i \leq mA$
6. Remember best lower bound
7. Reduce agility(π)
8. If π is too small (Beasley suggests 0.005) or best lower bound achieved finish else repeat from 2.

B. Implementation

To solve TSP with Lagrangian relaxation I used Matlab with TORSCHS Scheduling Toolbox. I implemented function `TSP_Lagrange(gSize, kIter)` with two parameters. Where `gSize` is number of nodes in automatically generated graph and `kIter` is number of iteration of subgradient method.

1. First step is to calculate Euclidean distances between nodes and preserve them in vector c .
2. Assemble other matrices for ILP of original problem of TSP
 - a. Add conditions 1-3 to matrix A
 - b. Set vector b
 - i. For conditions 1 and 2 set value 1
 - ii. For conditions 3 set value `gSize - 1`
 - c. Set lower bounds and upper bounds
3. Compute original TSP problem
4. Cut off condition 3 from all matrices of original problem
5. Iterate over subgradient method as described earlier
6. Plot graph of path and graph of evolution of optimal solutions

IV. EXPERIMENTAL RESULTS

A. Benchmark Settings

All tests were run on CPU i7-3630QM, 8.00 GB RAM and 64-bit Windows7. Version of Matlab was R2014b. I run `TSP_Lagrange` function with increasing number of graph size and number of iterations and track all results.

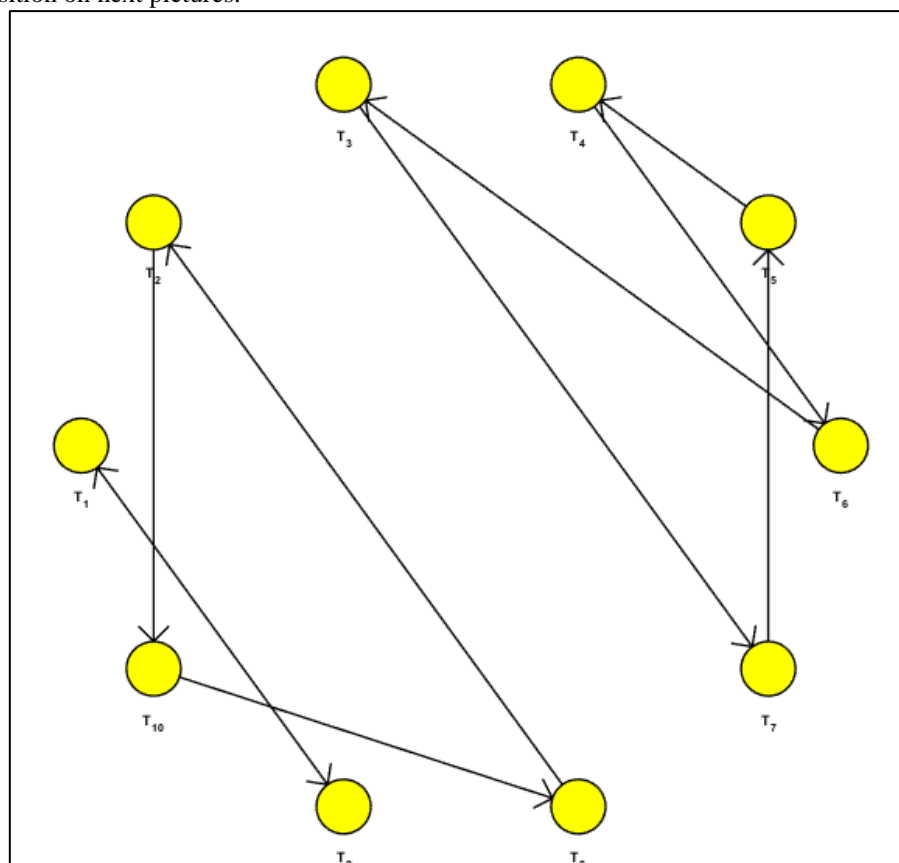
B. Results

Tested on different number of nodes and iterations.

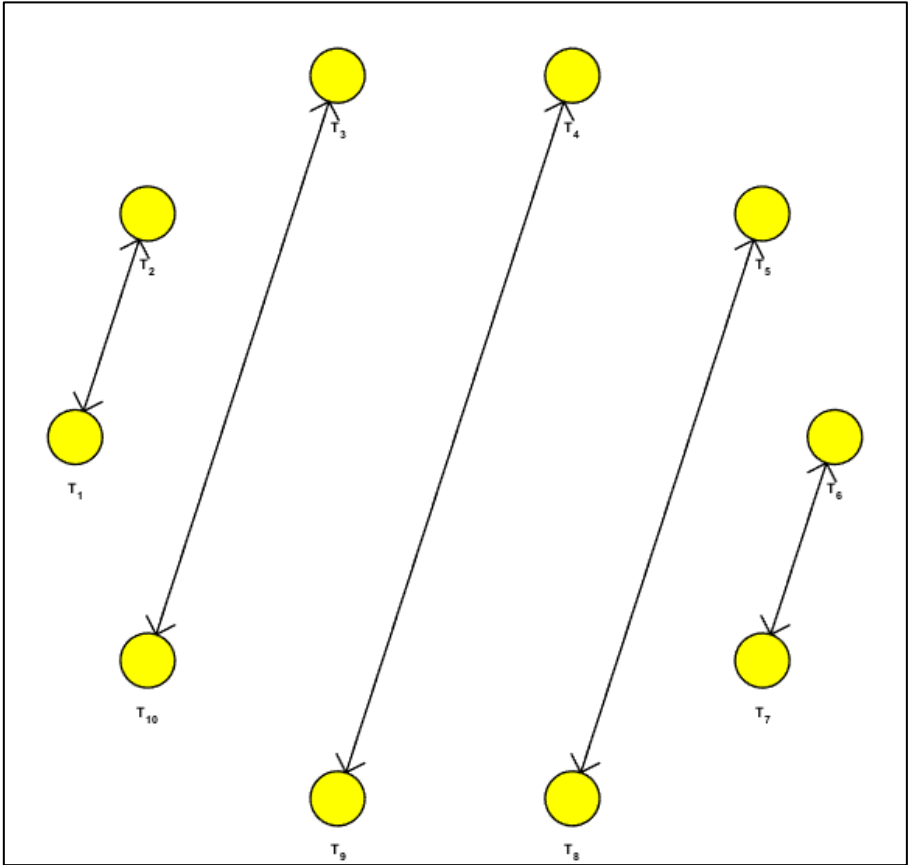
Nodes	iterations	time[s]	Original solution	Lagrange solution	Difference	Poznámky
5	50	3.147837	23.004304	22.108754	0.89555	
5	50	3.389138	18.876999	18.506146	0.370853	
5	100	4.721281	12.755604	12.424479	0.331125	
5	100	4.973687	20.080491	18.577188	1.503303	
5	250	4.317002	14.798264	13.832138	0.966126	
5	250	3.467993	15.61923	15.61923	0	Nalezeno optimální řešení
10	50	6.800718	120.286391	118.931728	1.354663	
10	50	7.528951	101.580177	99.561196	2.018981	
10	100	9.336933	107.970136	104.575488	3.394648	
10	100	10.03316	92.914749	89.585447	3.329302	
10	250	14.13531	102.756157	92.949819	9.806338	
10	250	14.80829	88.760703	80.625188	8.135515	
15	100	16.54413	232.787932	219.987467	12.800465	
15	100	17.7281	262.4769	258.484705	3.992195	
15	250	22.91515	307.6104	301.376743	6.233657	
15	250	24.1367	306.0797	305.343146	0.736554	
15	500	17.20099	303.583283	302.235717	1.347566	
15	500	17.93661	269.975004	264.886606	5.088398	
15	1000	33.55561	304.688059	296.377575	8.310484	
15	1000	33.03645	250.268649	239.071346	11.197303	

As we can see Lagrangian relaxation can also find the best solution. But there are good results in whole spectrum of tests. Average difference between Lagrangian and Original solution is 4.0906513 (5.8%).

There is graph composition on next pictures.

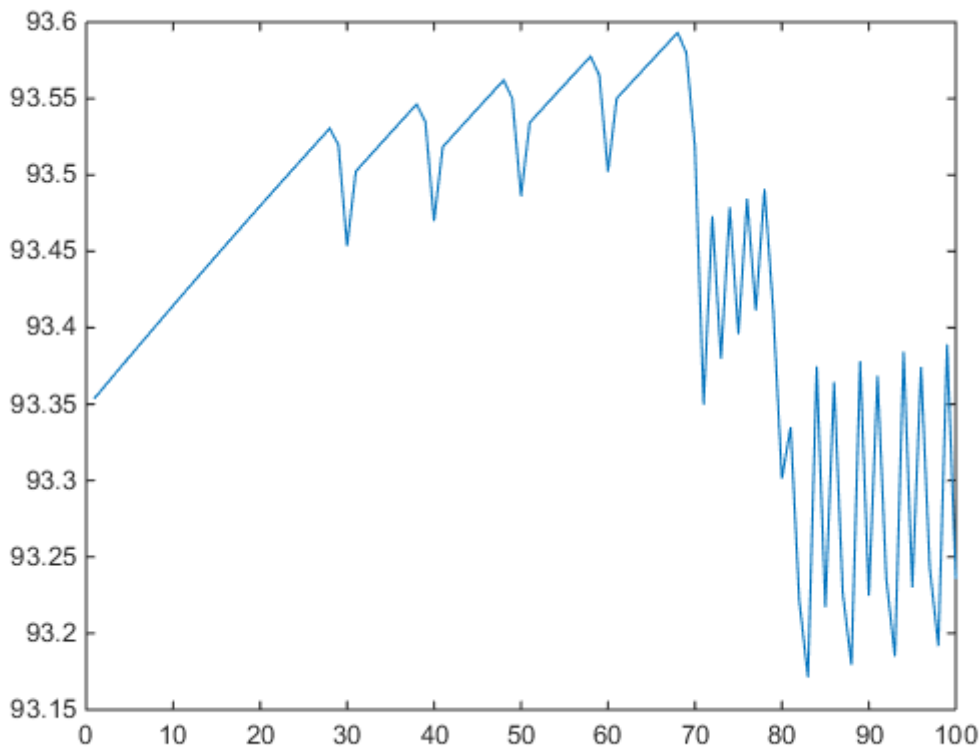


This graph has three strongly connected components. It is normal result because we removed one-cycle-condition. On the next graph it can be seen the worse result we can obtain from lagrangian relaxation with removed one-cycle-condition.

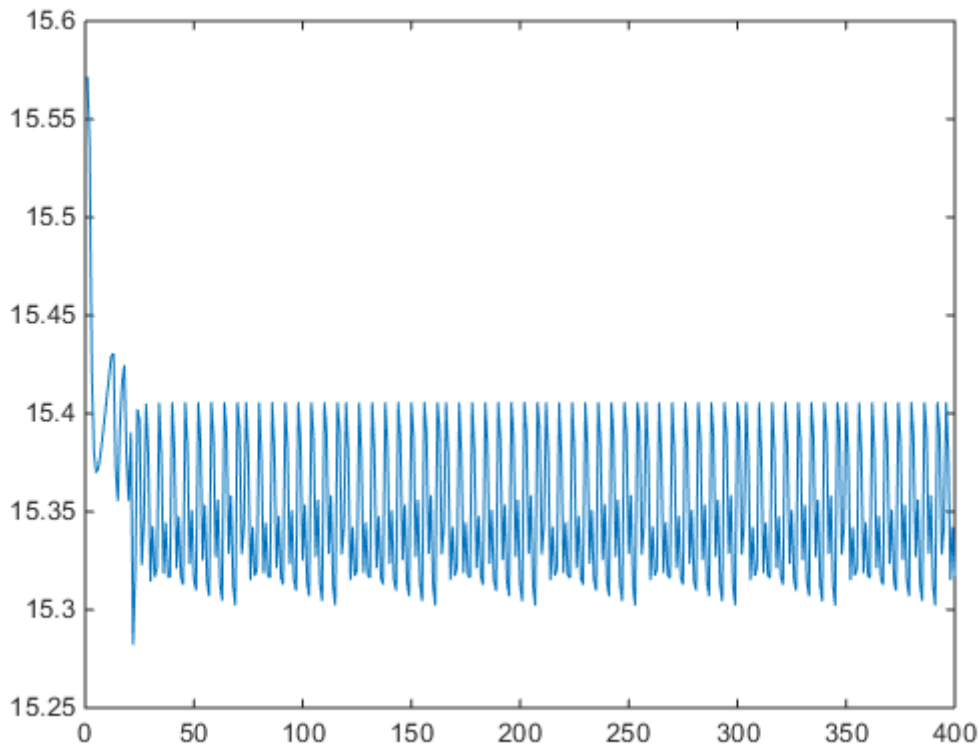


The solution made pairs but all conditions are fulfilled.

There is evolution of optimal solution on next graphs.



PIC EVO 1



PIC EVO 2

C. Discussion

All the results I obtain seem to be valid. I removed one-cycle-condition so the graph is not connected but all nodes have one leaving edge and one entering edge. Minimal cycle in graph has therefore two nodes.

Evolution of Lagrange solutions is also expected. There is no rule that solution has to converge to the best result of original problem. But I think that the oscillation of results is pretty impressive by Subgradient agility.

V. CONCLUSION

I deal with TSP problem and Lagrangian relaxation applied to it. It is well known problem with high time complexity. I find out that we can get feasible solution with breaking one or more conditions of TSP problem. It may not be only TSP it can be almost every formulated ILP problem. But there is also chance that we can get really bad results.

REFERENCES

- [1] Frédéric Giroire, F. H. (2012). *Linear programming and combinatorial optimization*. Ziskáno 15. March 2015, z ENS de Lyon: <http://www.ens-lyon.fr/DI/wp-content/uploads/2012/01/LagrangianRelax.pdf>
- [2] Kylie Bryant, A. B. (December 2000). *Genetic Algorithms and the Traveling Salesman Problem*. Ziskáno 23. March 2015, z Harvey Mudd College Department of Mathematics: <https://www.math.hmc.edu/seniorthesis/archives/2001/kbryant/kbryant-2001-thesis.pdf>
- [3] Nasini, S. (nedatováno). *The Travelling Salesman Problem: Introductory notes and computational analysis*. Ziskáno 20. March 2015, z Departament d'Estadística i Investigació Operativa: http://www-eio.upc.es/~nasini/Blog/TSP_Notes.pdf
- [4] Putz, P. (October 2007). <https://www.ac.tuwien.ac.at/files/pub/putz-07.pdf>. Ziskáno 4. May 2015, z Algorithms and complexity group: <https://www.ac.tuwien.ac.at/files/pub/putz-07.pdf>
- [5] S. Lin, W. K. (15. October 1971). Ziskáno 23. March 2015, z UC MERCED University of California, MERCED: https://eng.ucmerced.edu/people/yzhang/papers/Heuristic/Lin_Kernighan
- [6] Timsjö, S. (1999, JUne 15). *An Application of Lagrangian Relaxation to the Traveling Salesman Problem*. Retrieved March 19, 2015, from CiteSeerX: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.843&rep=rep1&type=pdf>