Solving Traveling Salesman Problem by Ranking

Ayhan Demiriz
Sakarya University
Sakarya, Turkey
ademiriz@gmail.com

Abstract

Network problems and related algorithms (i.e. graph based techniques) are known and studied by a diverse scientific and applied community. Recent studies in Machine Learning (ML) enable analyzing graph data for solving certain problems e.g. link analysis, citation analysis, supervised and unsupervised learning. It is a well-known fact that additional information, whether in the form of additional data, new variables or new set of information from a different source (point of view) might improve the underlying algorithm's performance in ML research. We propose a methodology for solving network problems by considering structural information of the network together with readily available problem definition. First, the vertices (nodes) in the graph are ranked and then the network problem is iteratively solved. The additional information on the network structure can enable reductions in the running time of exact algorithms and improvement on the solution quality of heuristics. We show that the approach is applicable, by demonstrating with on the well-known combinatorial optimization problem of TSP, traveling salesman problem. We also outline how the methodology can be adapted for solving the stable matching (marriage) problem. We report the performance of our methodology, on the TSPLIB benchmark datasets in comparison to the exact method of CONCORDE and other heuristics proposed in the literature.

1 Introduction

Certain network problems can be represented only through models that are computationally hard to solve. Traveling Salesman Problem (TSP) is one of the most famous problems studied in various disciplines. TSP is a fundamental Combinatorial Optimization (CO) problem. Given a set of cities and the costs (distances) associated with traversing between pairs of cities, the objective of TSP is to find the tour with the minimum total cost visiting each city exactly once. Many problems encountered in a variety of fields can be directly posed as TSP, or can be transformed into TSP through clever modeling twists. TSP applications include machine scheduling, cellular manufacturing, circuit design, routing problems in communications and supply chain networks, and genome sequencing.

The TSP is known to be NP-complete. Although it is very easy to state the TSP, it is extremely hard to solve it to optimality for large instances. Therefore various approaches have been deployed to overcome difficulties in solving TSP. Some methods solely depend on meta-heuristics based optimization such as using Genetic Algorithms, Ant Colonies and Simulated Annealing. Some other techniques are based on mathematical programming models such as branch-and-cut, Linear Programming (LP) relaxation of Integer Programming (IP) model, Lagrangian relaxation of IP model. In addition there are some widely used approximation algorithms to tackle with TSP. The best published theoretical complexity result is achieved by Held and Karp where their algorithm has O($n^2 2^n$) complexity. However the Concorde algorithm developed by Applegate, Bixby, Chvátal and Cook (Applegate et al., 2007) has achieved significant successes by optimally solving some large problems. It is based on branch-and-cut and uses some heuristics to increase its efficiency.

The TSP has been a well-known example of a hard combinatorial problem, commonly used to test new ideas in problem solving. It is no coincidence that some of the early papers on simulated annealing, DNA computing, and other approaches for the combinatorial problems describe their methods in the context of the TSP (Applegate et al., 2007). So the TSP is an important test bed of the new ideas.

An innovative and original approach is proposed for solving network problems particularly the TSP by borrowing the ideas from Graph Laplacian. The aim is to rank the cities first according to the underlying graph's structure (Demiriz, 2008) and then to develop a family of algorithms based on the rankings to solve the underlying network problems. Once the sequence of the vertices (cities in the TSP case) is given, it is easier to construct a fast algorithm to solve the underlying network problem. The rankings of the vertices simply determine the sequence that the algorithm should optimize the corresponding network problem.

The well-known Google's PageRank algorithm (Page et al., 1998) uses similar ideas for ranking web documents. Essentially, the web documents are ranked based on the incoming links (weights) on the graph. Notice that in the case of the web documents links are directed. The strength of the PageRank comes from two innovations. The first one is that realization of the hyperlinks as a measure of popularity. The second one is that the usage of anchortext in web index in place of using page title alone. In both cases, new information is embedded in learning process. A recent poll (More Data or Better Algorithm?)[1] in KDNUGGETS.COM suggests that more data is preferred to better algorithms among data mining practitioners. Therefore the proposed approach utilizes so-called structural information for solving network problems in general. Potentially such information should improve the existing heuristic approaches. The similar successes are attainable with this approach as in the case of the PageRank algorithm.

The applicability of our approach is shown by experimenting on TSPLIB[2] benchmark datasets. We report results from some other heuristics and the optimal tour lengths as well. We essentially present three different versions of our approach depending on the way of controlling the level of the information complexity in the underlying TSP. By the information complexity, we mean that either the full distance matrix or the partial information is present in the TSP. The full-matrix representation is considered as the highest level of information complexity as unnecessary information might exist within such representation.

The remaining of this paper is organized as follows. We introduce our methodology in the next section with a toy example. We also give the description of the the way information complexity is controlled in Section 2. We then give results based on the experiments run on the benchmark datasets from TSPLIB in the subsequent section. We finally conclude our paper in Section 4.

2 Foundation of the Methodology

Ranking the data is an ongoing research area with diverse applications. In this paper the usage of a ranking algorithm (Demiriz, 2008) based on graph Laplacian (Belkin and Niyogi, 2004; Belkin et al., 2006) is used to solve network problems. Ranking problem has recently become a major research area in machine learning. The ranking approach used in this paper resembles the algorithm proposed in (Zhou et al., 2004). The primary objective in that particular paper is to develop an algorithm based on some semi-supervised approach to rank the items for a given query. As in (Zhou et al., 2004), the ranking approach can exploit the intrinsic manifold structure of the data. Formally, ranking is defined as finding a function $f : \mathbb{R}^d \to \mathbb{R}$ that orders the data $X \in \mathbb{R}^d$ correctly. The ranking algorithm is based on graph representation of the data. Thus a graph $G = (V, E)$ can be formed from X by Euclidean neighborhood relations where $x \in X$ is represented by the vertices V and the relationships are represented by the edges $E \subseteq V \times V$ on the graph.

The spectral graph theory (Chung, 1997) is utilized to tackle the ranking problem. Essentially, the spectral properties of the normalized Laplacian are used for this purpose. Normalized Laplacian is defined as $\mathcal{L} = D^{-1/2}LD^{-1/2} = D^{-1/2}(D - W)D^{-1/2} = I - D^{-1/2}WD^{-1/2}$ where W is the adjacency matrix, D is a diagonal matrix formed by row sums of W, L is the traditional Laplacian matrix i.e. $D - W$, and I is the identity matrix (Chung, 1997). One of the most important spectral properties of the normalized Laplacian ($\mathcal{L}$) is that its eigenvalues vary between 0 and 2. If there are multiple eigenvalues which are equal to 0 then the underlying graph is not connected. An eigen-

[1]http://www.kdnuggets.com/news/2008/n08/1i.html
[2]http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

Table 1: Distance Matrix Between Airports (in km) (Adapted from ((Gueret et al., 2000))

	A1	A2	A3	A4	A5	A6	A7
A1	0	786	549	657	331	559	250
A2	786	0	668	979	593	224	905
A3	549	668	0	316	607	472	467
A4	657	979	316	0	890	769	400
A5	331	593	607	890	0	386	559
A6	559	224	472	769	386	0	681
A7	250	905	467	400	559	681	0

value of 2 indicates that the graph is bipartite. On the other hand, it is known from the convergence of the random walk that the stationary distribution, π, of the random walk is equivalent to the eigenvector corresponding to eigenvalue 1 of the underlying transition matrix i.e. $P = D^{-1}W$. In other words, the corresponding eigenvector for this transition matrix, P, can easily be shown that is equal to $\pi = \frac{1D}{\sum_l d_l}$. This particular stationary distribution is achieved, if the graph is connected.

Practically, there is no need to use the power method to find the stationary distribution once it is shown that the underlying graph is connected. Otherwise, an algorithm is utilized that it is similar to Google's PageRank (Page et al., 1998) which is not necessarily symmetric (undirected) to find the stationary distribution of the random walk (Demiriz, 2008). Since the problem studied in this paper is the symmetric TSP and the underlying graph is connected, the stationary distribution π can simply be utilized to rank the cities.

2.1 Description of the Algorithm

The proposed approach is summarized in Figure 1. Essentially once the rankings are supplied either a greedy approach can be used or the rankings are used to warm-start any related algorithm to solve the network problems (TSP in this case). In other words, the sequence given by the rankings can be used on any algorithm of choice to solve network problems.

2.2 Toy Example

To demonstrate the approach on a toy example, the sample dataset from (Gueret et al., 2000), (see Table 1) is used. The dataset is given in the full-matrix format to illustrate the algorithm simpler in Table 1. The sample data is originally used

- Calculate the stationary distribution π to rank the vertices
  1. If a connected graph (e.g. a symmetric adjacency matrix) is considered, simply use $\pi = \frac{1D}{\sum_l d_l}$ formula.
  2. Otherwise use the power method such as the PageRank algorithm starting from π.
- Starting from the ranking results use either a greedy approach or warm start any known method to find solution to the network problem.

Figure 1: A Concise Depiction of the Algorithm

to build a mathematical programming model to plan (construct) a flight tour between airports. If π is computed from the distance (adjacency in this case) matrix, the stationary distribution for the airports is calculated to be as (0.1300, 0.1724, 0.1278, 0.1665, 0.1397, 0.1283, 0.1354). Thus the ranking is (A2, A4, A5, A7, A1, A6, A3).

A simple algorithm can be devised once the sequence (ranking) is given as follows. Given that A2 has the highest rank, greedy approach needs to start optimizing from A2 first. It is the best choice to fly to A6 from A2 (224 km) based on the distance matrix. Thus a flight tour must have a section from A2 to A6. At this point column 6 (A6) and row 2 (A2) can be crossed-out from the distance matrix to enable the selection correctly for the remaining steps. We then check the row 4 for the shortest distance available to fly since A4 is the next in the ranking list. From A4 the shortest way leads to A3 which is 316 km in distance. We then cross-out column 3 and row 4 from the distance matrix. The next airport is A5 in the ranking list to be optimized. It is the best choice to fly A1 from A5 (331 km). We cross-out column 1 and row 5 at this step. For airport A7, the cheapest flight will be to A4 since the first column is crossed-out already i.e. 400 km in distance. We then cross-out column 4 and row 7. Similarly from A1, A6, and A3 the best flights will be to A7, A5, and A2 respectively. Thus the best tour according to this greedy approach will include following legs: A2$\rightarrow$A6, A4$\rightarrow$A3, A5$\rightarrow$A1, A7$\rightarrow$A4, A1$\rightarrow$A7, A6$\rightarrow$A5, and A3$\rightarrow$A2 in total a distance of 2575 km. In short a tour starting A2 will be A2$\rightarrow$A6$\rightarrow$A5$\rightarrow$A1$\rightarrow$A7$\rightarrow$A4$\rightarrow$A3$\rightarrow$A2. This is indeed the shortest possible tour. So with the help of ranking results, the greedy algorithm is able to find the optimal tour.

2.3 Expanding the Scope

It is evident from initial studies that search space can significantly be reduced by ranking the vertices in network problems. It is also evident that information overloading might cause some problems at the ranking step. By information overloading it is meant that a fully-connected distance matrix may have excessive information in the first place. Therefore some of the links in the graph might be unnecessary after all. Similar observations are also made in (Sourlas, 1986). In (Sourlas, 1986), it is noticed that the solutions only contain links between very near neighbors. So it is acceptable to remove some of the links. In short, more work is needed to justify the need to trim search space by removing some links of the underlying graphs deliberately. This indeed creates a new family of problem: reducing the search space by locating and removing the excessive information in the adjacency matrix.

To control the level of information complexity we can devise three different approaches. We can essentially change the size of the nearest neighborhood across the cities until the tour constraints are violated and the size of the largest distance to constitute the neighborhood boundaries by starting the largest distance in the distance matrix to remove and then continue removing until tour constraints are violated. These two methods effectively construct constraints for all the cities i.e. they determine the size of the neighborhood globally. In other words, assume there are n cities in the TSP, practically the original TSP has a nearest neighborhood size of $n-1$ i.e. the full distance matrix; then we can start reducing the nearest neighborhood size by removing the largest distance from any city to the others until the tour constraints are violated i.e. a tour can no longer be formed according to greedy approach. In the second approach, by starting from the largest distance in the full distance matrix to remove one by one, we can effectively draw balls (i.e. $\varepsilon - ball$) around the cities to determine minimally acceptable balls that forms a tour. Notice that the size of these balls are same for all the cities in this way. The third way is to determine a neighborhood size for the each city separately. In this way, we can essentially adjust the $\varepsilon - balls$ for the each city in the TSP. Notice once the ranking is determined, all three approaches greedily choose the next best movement at the each step until the tour constraints are violated i.e. no more movement is possible or addition of such a movement creates a short tour which is not feasible. We leave further discussions on the details of our implementation to the next section.

So far our formulation (usage) of Graph Laplacian ranks cities (nodes) based on the total distances. This is equivalent to using the mean value over n cities (nodes) on fully connected network. However it might be problematic to use the summation in our case since we try to reduce the information complexity which effectively assigns zeros to some corresponding values. In other words it disconnects some cities from each other. Therefore one can argue that using the mean (expected) value which excludes the zeros (disconnected cities) from the calculation is a better statistic than using the summation. Not only the mean value but also the variance (or standard deviation) could be used in this case as an alternative statistic determining the ranking values again on nonzero ones i.e. excluding the disconnected cities. Thus we will use both statistics to rank the cities in the next section as well as plain summation over n cities regardless their connectivity.

Our approach can be applied to other network problems. We can easily utilize the idea of ranking vertices on the stable marriage (matching) problem too. The problem is defined as "given n men and n women, where each person has ranked all members of the opposite sex with a unique number between 1 and n in order of preference, marry the men and women off such that there are no two people of opposite sex who would both rather have each other than their current partners If there are no such people, all the marriages are stable" [3] . The Gale-Shapley (G-S) algorithm finds the stable solution with an $O(n^2)$ complexity. Assume that the preferences are given as in the G-S algorithm. A ranking scheme can be constructed by summing incoming votes (preferences). Essentially there are two $n \times n$ preference matrices: men's and women's preferences. Each person lists his or her preferences from the opposing sex. When the preferences are summed column wise, the people in the opposing sex are essentially ranked. Thus everybody gets a ranking point. Notice that G-S algorithm is not neutral in terms of the sex; it weighs

[3]Wikipedia

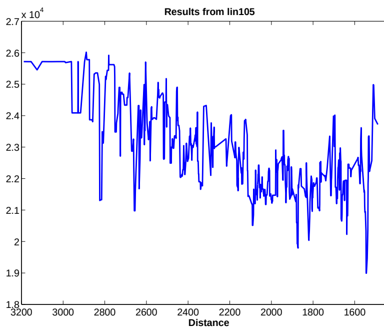

Figure 2: Results from lin105 Using LocDist without Heat Kernel

one of the sexes (man or women depending on the choice). However the ranking scheme is neutral. Then the priority can be given to the highest ranking person to choose his or her mate and continue with the next person in the ranking list to choose his or her mate until every person is matched with another person. Of course when one chooses his or her mate, he or she chooses the first available person on his or her preference list at that particular step. Notice that there is no engagement state in this type of matching and the complexity is $O(n)$. This particular problem is presented here to show that the proposed approach in this paper can be applied to other network problems as well beyond the TSP. However, caution should be taken as this approach may not result in a stable marriage. In the following section we report results from the experiment on TSPLIB benchmark datasets.

3 Experimental Evaluation

As mentioned in Section 2.3, we run experiments with three different versions of our algorithm. All of them are built on the idea of reducing the information complexity of TSP i.e. full-matrix representation. The first one utilizes the idea of reduced nearest neighborhood in terms of size. The second one on the other hand utilizes a reduced $\varepsilon - ball$ for all the cities in the graph at the same time. The third approach generalizes $\varepsilon - ball$s to the all cities by searching the locally minimum $\varepsilon - ball$s. Thus the third approach can potentially reduce the size of the neighborhood for the each city in the graph.

Before we report the results from our approach, we first present the results from the other well-

Table 2: Optimal and Heuristics Results

Dataset	Greedy	Boruvka	Q-Boruvka	Lin-Ker.	Optimal
a280	3108	**3261**	3151	2579	2579
bayg29	**2066**	2012	2018	1610	1610
bays29	2277	2134	**2653**	2020	2020
berlin52	9951	**10117**	9529	7542	7542
ch130	7167	6861	**7283**	6110	6110
d493	**40836**	40340	39877	35132	35002
d657	**58053**	57365	57212	49002	48913
eil51	521	**541**	480	426	426
eil76	**631**	577	609	538	538
lin105	16766	16469	**17582**	14379	14379
rd400	17359	**18716**	18004	15281	14379
u2319	262757	**271600**	264996	234519	234256

Table 3: Results from Three Different Methods with Simple Summation

Dataset	RedNN (Size)	RedDist (Dist)	LocDist (Dist)
a280	5364 (279)	3971 (225)	3971 (156)
bayg29	2278 (26)	**1795 (25)**	2014 (282)
bays29	2790 (28)	**2415 (358)**	**2415 (358)**
berlin52	12313 (49)	10682 (1092)	10277 (859)
ch130	9934 (129)	8653 (1)	8809 (413)
d493	68903 (492)	68728 (3465)	68728 (3465)
d657	106383 (656)	103950 (3034)	103950 (3024)
eil51	624 (50)	571 (57)	**506 (35)**
eil76	888 (75)	739 (57)	722 (46)
lin105	25715 (102)	20643 (31)	18991 (1489)
rd400	31145 (397)	25936 (908)	23578 (656)
u2319	610482 (2318)	478108 (4438)	478108 (3406)

known heuristics namely Greedy, Boruvka, Quick Boruvka and Lin-Kernighan in addition to the optimal results. The experiments reported in Table 2 are run by using CONCORDE [4]. Clearly the Lin-Kernighan method is the best heuristics. The worst result for the each benchmark problem is formatted in **bold** in Table 2 for the purpose of comparisons later. In the subsequent part of the paper, when we present the results from our approach we highlight better results than the worst cases highlighted in Table 2.

In Table 3, we report results from the three different versions (RedNN, RedDist and LocDist) of the reduced information complexity. Recall that all the three methods rank the cities based on the neighborhood boundaries drawn by different flavors of the information complexity. We report results on 12 different TSPLIB benchmark datasets in various sizes. Reduction boundaries are given in parentheses in the form of either NN size or the size of the $\varepsilon - ball$. Distance based ($\varepsilon - ball$s) reductions give consistently better results compared with nearest neighbor (NN). Notice that NN based reduction cannot reduce the size of the neighborhood as efficient as the distance based methods.

We also extend our approach with the utilization of the heat kernels (Demiriz, 2008) i.e.

[4]http://www.tsp.gatech.edu/concorde/index.html

Table 4: Results from the Usage of Heat Kernels with Simple Summation

Dataset	MaxDist	RedNNheat (Size)	RedDistheat (Dist)	LocDistheat (Dist)
a280	302	3973 (199)	4328 (201)	4234 (149)
bayg29	386	**2024 (25)**	**1973 (315)**	**1973 (165)**
bays29	509	3164 (28)	3164 (509)	3164 (437)
berlin52	1716	**9879 (37)**	10394 (1507)	**9547 (774)**
ch130	939	9626 (110)	9930 (704)	8752 (407)
d493	4296	65549 (421)	67374 (3525)	65593 (2479)
d657	4771	89873 (461)	93860 (3139)	93158 (2749)
eil51	86	573 (46)	573 (68)	549 (46)
eil76	85	717 (69)	735 (71)	705 (54)
lin105	3189	22191 (70)	24933 (2180)	21983 (1229)
rd400	1353	27521 (251)	27777 (892)	27777 (659)
u2319	6862	500816 (1512)	543725 (3936)	543725 (3499)

$exp[-d^2(x_i, x_j)/2\sigma^2]$ where $d(x_i, x_j)$ is the distance between point x_i and point x_j. The results are reported in Table 4. We first normalize the benchmark datasets by dividing by corresponding maximum distances reported in Table 4 and then set the parameter σ to be equal to 1. It is noted that the usage of heat kernels enables the algorithm to reduce the information complexity across three approaches i.e. the reduced NN, the reduced distance and the local distance. These results clearly indicate that if we can reduce the information complexity of traveling salesman problems, we might achieve better results. Notice that we do not attempt to search for the optimum parameter (σ) values in these experiments. We prefer using σ equal to 1 for the normalized data to avoid any extra parameter search. However the kernel parameter certainly should be optimized to yield better tours. Our aim with these experiments is to show the applicability of our idea.

To depict the change of the tour lengths by reducing the information complexity in the local distance method, we report results from `lin105` dataset in Figure 2. The maximum distance in `lin105` dataset is 3189 as reported in Table 4. The algorithm starts with the full distance matrix and then reduces the distance at each iteration. Recall that any connection that have a higher distance than the lower limit is omitted from the distance matrix. Thus the information complexity is reduced at each iteration. It should also be noted that the lower limits are different for the each point (city) locally. In other way, the lowest distance limit, i.e. 1489, is not for all the points (cities) but it is valid for some localities. The minimum tour length, 18991, is achieved at a distance of 1545. The distance matrices for the TSPs are usually provided with full connections. However, in reality a city is directly connected with only few other cities. We think that if only the real con-

Table 5: Results from Three Different Methods with Mean Value

Dataset	RedNN (Size)	RedDist (Dist)	LocDist (Dist)
a280	5364 (279)	4094 (189)	3858 (107)
bayg29	2278 (28)	**1596 (25)**	**1956 (220)**
bays29	2790 (28)	**2353 (308)**	**2335 (269)**
berlin52	12313 (49)	10455 (15)	12054 (871)
ch130	9934 (129)	8220 (1)	9224 (393)
d493	68903 (492)	63824 (18)	68402 (3434)
d657	106383 (656)	104475 (2985)	104475 (2985)
eil51	624 (50)	**537 (2)**	590 (42)
eil76	888 (75)	820 (2)	863 (46)
lin105	25715 (102)	19646 (31)	19452 (1263)
rd400	31145 (395)	28840 (754)	23017 (458)
u2319	610482 (2318)	554949 (3538)	457532 (2319)

Table 6: Results from Three Different Methods with Mean Value on Heat Kernel

Dataset	RedNN (Size)	RedDist (Dist)	LocDist (Dist)
a280	4169 (177)	4070 (217)	3983 (120)
bayg29	**2024 (25)**	**2024 (315)**	**1884 (103)**
bays29	3164 (28)	3164 (509)	2303 (89)
berlin52	**9674 (34)**	**10024 (1267)**	**9341 (428)**
ch130	9648 (121)	9908 (678)	7850 (108)
d493	66796 (410)	66606 (3535)	66450 (3308)
d657	94103 (389)	93018 (3211)	93018 (2863)
eil51	573 (44)	560 (64)	**511 (19)**
eil76	712 (66)	735 (70)	680 (24)
lin105	22940 (70)	24941 (2324)	18085 (775)
rd400	28494 (234)	28859 (882)	21230 (135)
u2319	543954 (1238)	535715 (3805)	449709 (2220)

nections are provided, the problem will have less information complexity. Thus we think that the ranking approach might perform remarkably well in this situation.

3.1 Extending the Results to Other Statistics

In this section we report results of the three methods used above with different statistics namely mean and variance of the corresponding vertices (cities) of the distance (or kernel) matrix. This could be perceived a bit departure from the idea of Graph Laplacian. As discussed in Section 2.3, we effectively disconnect some vertices by reducing the information complexity. Therefore to prevent the bias of the disconnected vertices, we propose the usage of the mean value and the variance as

Table 7: Results from Three Different Methods with Variance

Dataset	RedNN (Size)	RedDist (Dist)	LocDist (Dist)
a280	5042 (279)	3990 (0)	3877 (92)
bayg29	2257 (27)	**1948 (244)**	**1808 (204)**
bays29	2870 (28)	**2073 (0)**	**2294 (55)**
berlin52	11000 (50)	10675 (1387)	**8811 (747)**
ch130	9960 (128)	**7049 (1)**	7980 (426)
d493	70659 (492)	68323 (3378)	68323 (3378)
d657	98964 (656)	95904 (3112)	93142 (3037)
eil51	616 (50)	547 (47)	547 (45)
eil76	862 (74)	734 (51)	701 (49)
lin105	26225 (104)	22899 (2142)	19671 (978)
rd400	30890 (398)	23240 (728)	22808 (538)
u2319	536742 (2318)	470039 (100)	471603 (2961)

Table 8: Results from Three Different Methods with Variance on Heat Kernel

Dataset	RedNN (Size)	RedDist (Dist)	LocDist (Dist)
a280	5102 (279)	3625 (167)	3625 (108)
bayg29	2247 (28)	1760 (226)	1760 (163)
bays29	2593 (28)	2575 (300)	2311 (234)
berlin52	11358 (49)	11168 (1163)	9450 (1041)
ch130	10315 (129)	8517 (484)	7710 (296)
d493	72457 (492)	69116 (3620)	69116 (3558)
d657	106777 (656)	101600 (3120)	99620 (3046)
eil51	624 (50)	567 (45)	567 (45)
eil76	857 (75)	773 (51)	741 (44)
lin105	23772 (104)	21537 (1832)	19671 (1195)
rd400	31389 (398)	23932 (703)	22731 (543)
u2319	583728 (2318)	490255 (3130)	456026 (2631)

alternative statistics. We use similar experimental setup as above on non-zero values of distance (heat kernel) matrix. The results are reported in Tables 5, 6, 7 and 8. We present results on both the plain distance matrix and the heat kernel matrix.

The results indicate some improvements over simple summation. Notably we now have more highlighted results indicating that the results are better than the some worst cases of the known heuristics reported in Table 2. Notice that our approach clearly fails for higher dimensional problems i.e. larger TSPs. This certainly indicates that the search procedure is not as powerful as needed and it is highly prone to local minima. However the scope of our study is simply to show the applicability of our approach.

3.2 Experimenting with the Heat Kernel Parameter

In this section we report results from the three methods studied in the paper with the usage of the summation, the mean value and the variance of each row of the kernel matrix. In these experiments for the normalized data we change the heat kernel parameter σ between 0.4 and 1.6. The results are depicted in Figures 3, 4 and 5. Clearly the method `LocDist` outperforms the others in-

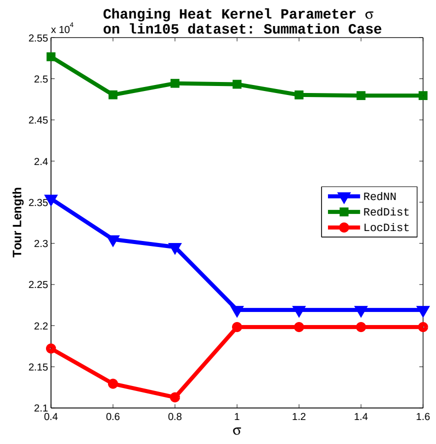

Figure 3: Changing Heat Kernel Parameter with Summation

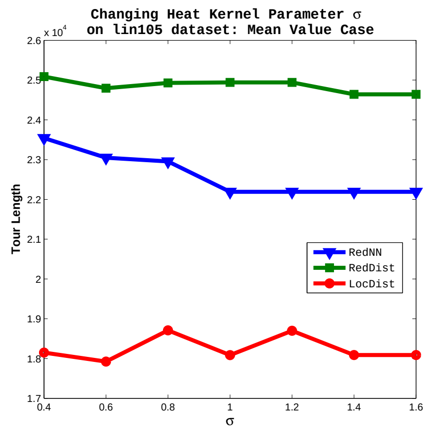

Figure 4: Changing Heat Kernel Parameter with Mean Value

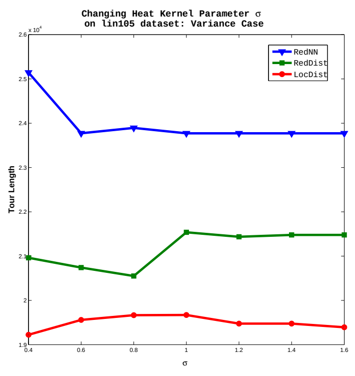

Figure 5: Changing Heat Kernel Parameter with Variance

dicating that reducing the information complexity locally pays off with constantly better tours.

We still need to do further research to extend the idea of reducing the information complexity to other heuristics methods and the usage of ranking on traveling salesman problems. The aim of this set of the experiments is to show how changing the kernel parameter σ affects the ranking results as well as TSP solutions.

4 Conclusions

We use the idea of Graph Laplacian to rank the data (cities) in order to solve underlying network problems specifically TSP. The idea is promising and requires further research to reach a conclusive outcome. We show that reducing the information complexity in TSP will result in better tours. This can be extended to any known methods in general. Different approaches to reduce the information complexity can be proposed for this purpose. We implement three methods in this regard which are named as reduced NN, reduced distance and local distance. We show the applicability of our idea by experimental evaluations on TSPLIB datasets. We think that ranking will result in better tours in the case of networks (graphs) with limited connections provided that the tour constraints are satisfied. In general, the idea of ranking nodes of any network to solve the underlying network problem can be efficiently implemented. We show this specifically on TSP and discuss the applicability on matching problems particularly stable marriage problem.

In our experiments, we reduce the information complexity starting from the fully connected distance matrix. However the direction of the search can be changed to start with an empty distance matrix and then add connections to it to find best possible partial connection matrix that satisfy the tour constraints. We also think that recent advances in constraint programming can be potentially utilized with the idea of ranking presented in this paper to solve TSPs and other network problems in general.

Acknowledgments

This study is partially supported by the Turkish Scientific Research Council through the grant TUBITAK 107M257. Author also would like to thank to John Shawe-Taylor of University College of London for introducing the topic of Graph Laplacian.

References

David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. 2007. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

Mikhail Belkin and Partha Niyogi. 2004. Semi-supervised learning on riemannian manifolds. *Journal of Machine Learning*, 56:209–239.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434.

Fan R.K. Chung. 1997. *Spectral Graph Theory*. American Mathematical Society.

Ayhan Demiriz. 2008. Ranking data with graph laplacian. Kayseri, Turkey. International Conference on Multivariate Statistical Modelling & High Dimensional Data Mining.

Christelle Gueret, Christian Prins, and Marc Sevaux. 2000. *Application of Optimization with Xpress-MP*. Dash Optimization.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web.

N. Sourlas. 1986. Statistical mechanics and travelling salesman problem. *Europhysics Letters*, 2(12):919–923.

Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. 2004. Ranking on data manifolds. In *Advances in Neural Information Processing Systems 16*. MIT Press.