14

Predicting Parallel TSP Performance: A Computational Approach

Paula Fritzsche, Dolores Rexachs and Emilio Luque
DACSO, University Autonoma of Barcelona
Spain

1. Introduction

Faced with a scientific force and a critical need to solve large-scale and/or time-constrained problems, the industry reports that access to high-performance computing (HPC) capability is required now more than ever. Continued hardware and software advances, such as more powerful and lower-cost processors, have made it easier for scientists and engineers to install and use clusters / multi-cores and complete high-performance computing jobs.

In particular, the Traveling Salesman Problem (TSP) is one of the most famous problems (and the best one perhaps studied) in the field of the combinatorial optimization. In spite of the apparent simplicity of their formulation, the TSP is a complex solving problem and the complexity of its solution has been a continue challenge to the mathematicians for centuries. Not only the study of this problem has attracted people from mathematics but also many researchers of other fields like operations research, physics, biology, or artificial intelligence, and accordingly there is a vast amount of literature on it. On the other hand, not yet an effective polynomial-time algorithm is known for the general case. Many aspects of the problem still need to be considered and questions are still left to be answered satisfactorily. A significant challenge is being able to predict TSP performance order. It is important to bear in mind, that the TSP conclusions drawn could eventually be applied to any TSP family problem. There are important cases of practical problems that can be formulated as TSP problems and many other problems are generalizations of this problem. Therefore, there is a tremendous need for predicting the performance order of TSP algorithms.

Measuring the execution time (performance) of a TSP parallel algorithm for all possible input values would allow answering any question about how the algorithm will respond under any set of conditions. Unfortunately, it is impossible to make all of these measurements. TSP performance depends on the number of cores used, the data size, as well as other parameters. Detecting the main other parameters that affect performance order is the real clue to obtain a good estimation. The issue of measuring performance for the TSP problem in practice and how to relate practical results to the theoretical analysis is addressed in this chapter as a knowledge discovery methodology.

The defined methodology for performance modelling begins by generating a representative sample of the full population of TSP instances and measuring their execution times. An interactive and iterative process explores data in search of patterns and/or relationships detecting the main parameters that affect performance. Knowing the main parameters which characterise time complexity, it becomes possible to suspect new hypotheses to

restart the process and to produce a subsequent improved time complexity model. Finally, the methodology predicts the performance order for new data sets on a particular parallel computer by replacing a numerical identification. The methodology arises out of the need to give an answer to a great number of problems that are normally set aside. Besides, this is a good starting point for understanding some facts related with the non-deterministic algorithms, particularly the data-dependents algorithms. Any minimum contribution in this sense represents a great advance due to the lack of general knowledge.

An Euclidean TSP implementation, called global pruning algorithm (***GP-TSP***), to obtain the exact TSP solution in a parallel machine has been developed and studied. It is used to analyze the influence of indeterminism in performance prediction, and also to show the usefulness of the methodology. It is a branch-and-bound algorithm which recursively searches all possible paths and prunes large parts of the search space by maintaining a global variable containing the length of the shortest path found so far. If the length of a partial path is bigger than the current minimal length, this path is not expanded further and a part of the search space is pruned. The ***GP-TSP*** execution time depends on the number of processors (***P***), the number of cities (***C***), and other parameters. As a result of our investigation, right now the sum of the distances from one city to the other cities (***SD***) and the mean deviation of SDs values (***MDSD***) are the numerical parameters characterizing the different input data beyond the number of cities.

Comparisons of experimental results with predictions have been quite promising. Therefore, the efficacy of the methodology proposed has been demonstrated. In addition to the prediction capability, an interesting and practical issue from this research has been discovered: how to select the *best* starting city. With this important and non-trivial selection, the time spent on evaluation has been dramatically reduced.

This chapter is organized as follows. The next section describes the Traveling Salesman Problem, their computational complexity and their applications in several fields. Besides, it provides detailed coverage of the ***GP-TSP*** parallel algorithm. Section 3 presents the knowledge discovery methodology to the problem of predicting the TSP performance. Section 4 focuses on the discovering process carried out to find the significant input parameters and building the ***GP-TSP*** prediction model. In addition, two outstanding experiments have been studied. Section 5 summarizes and draws the main conclusions of this chapter. Appendix A shows the specification of the parallel machine used along the experimentation stage. Appendix B shows the characteristics of a clustering tool used to discover internal data information.

2. Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is one of the most famous problems (and the best one perhaps studied) in the field of combinatorial optimization. In spite of the apparent simplicity of its formulation, the TSP is a complex data-dependent problem. Not only the complexity of its solution has been a continue challenge to the researchers of several fields but also the prediction of its performance. Predicting TSP performance is vital due to there are many practical problems that can be formulated as TSP problems and others problems are generalizations of this problem.

2.1 Problem statement

The TSP for *C* cities is the problem of finding a tour visiting all the cities exactly once and returning to the starting city such that the sum of the distances between consecutive cities is

minimized (TSP page, 2010). The requirement of returning to the starting city does not change the computational complexity of the problem.

2.2 Computational complexity

The TSP has been shown to be NP-hard (Karp, 1972). More precisely, it is complete for the complexity class (FP^{NP})[1], and the decision problem version is NP-complete. If an efficient algorithm is found for the TSP problem, then efficient algorithms could be found for all other problems in the NP-complete class. Although it has been shown that, theoretically, the Euclidean TSP is equally hard with respect to the general TSP (Garey et al., 1976), it is known that there exists a sub exponential time algorithm for it.

The most direct solution for a TSP problem would be to calculate the number of different tours through *C* cities. Given a starting city, it has *C-1* choices for the second city, *C-2* choices for the third city, etc. Multiplying these together it gets *(C-1)!* for one city and *C!* for the *C* cities. Another solution is to try all the permutations (ordered combinations) and see which one is cheapest. At the end, the order is also factorial of the number of cities. Briefly, the solutions which appear in the literature are quite similar.

The factorial algorithm's complexity motivated the research in two attack lines: exact algorithms or heuristics algorithms. The exact algorithms search for an optimal solution through the use of branch-and-bound, linear programming or branch-and-bound plus cut based on linear programming (Karp, 1972) techniques. Heuristics solutions are approximation algorithms that reach an approximate solution (close to the optimal) in a time fraction of the exact algorithm. TPS heuristics algorithms might be based on genetic and evolutionary algorithms (Tsai et al., 2002), simulated annealing (Pepper et al., 2002), Tabu search, neural networks (Aras et al., 2003), ant systems, among others.

2.3 Practical problems

The TSP often comes up as a subproblem in more complex combinatorial problems. The best known and important one of which is the vehicle routing problem, that is, the problem of determining for a fleet of vehicles which customers should be served by each vehicle and in what order each vehicle should visit the customers assigned to it (Christofides, 1985). Another similar example is the problem of arranging school bus routes to pick up the children in a school district. The TSP naturally arises in many transportation and logistics applications (TSP page, 2010).

Besides problems having the TSP structure occur in the analysis of the structure of crystals (Bland & Shallcross, 1989), in material handling in a warehouse (Ratliff & Rosenthal, 1983), in genome rearrangement (Sankoff & Blanchette, 1997), in phylogenetic tree construction (Korostensky & Gonnet, 2000), and predicting protein functions (Johnson & Liu, 2006), among others. Important practical computer science problems including the TSP structure appear in clustering of data arrays (Lenstra & Kan, 1975), in sequencing of jobs on a single machine (Gilmore & Gomory, 1964), in physical mapping problems (Alizadeh et al., 1993), in drilling of printed circuits boards (Duman, 2004).

[1] The class NP is the set of decision problems that can be solved by a non-deterministic Turing machine in polynomial time. FP means function problems.

2.4 Related problems

An equivalent formulation in terms of graph theory can be described. Given a complete weighted graph find a Hamiltonian cycle with the least weight. The vertices would represent the cities, the edges would represent the roads, and the weights would be the cost or distance of that road (Gutin & Punnen, 2006).

Another related problem consists of finding a Hamiltonian cycle in a weighted graph with the minimal length of the longest edge. This problem, known as the bottleneck traveling salesman problem, is really useful in transportation and logistics areas.

Related variations on the TSP include the resource constrained traveling salesman problem which has applications in scheduling with an aggregate deadline (Miller & Pekny, 1991). The prize collecting TSP (Balas, 1989) and the orienteering problem (Golden et al., 1987) are special cases of the resource constrained TSP. The problem of finding a tour of maximum length is the objective in MAX TSP (Barvinok et al., 2003). The maximum scatter TSP is the problem of computing a path on a set of points in order to maximize the minimum edge length in the path. It is motivated by applications in manufacturing and medical imaging (Arkin et al., 1996).

2.5 *GP-TSP* parallel algorithm

As a representative of the practical problems, a global pruning TSP algorithm (called *GP-TSP*), has been deeply studied. It obtains the exact TSP Euclidean solution in a parallel machine. For simplicity, the algorithm works with cities in R^2 instead of R^3 and uses the Euclidean distance due to it is the most straightforward way of computing distances between cities in a two-dimensional space. Nevertheless, the choice of the distance measure used (Euclidean, Manhattan, Chebychev, …) is irrelevant. More over, it would be the same to work with an equivalent formulation in terms of graph theory. Therefore, the ideas of this article can be generalized.

The *GP-TSP* algorithm is indeed both useful and profitable to analyze the influence of indeterminism in performance prediction. It is a branch-and-bound algorithm which recursively search all possible paths. It follows the Master-Worker programming paradigm (Fritzsche, 2007). Each city is represented by two coordinates in the Euclidean plane. Considering C different cities, the Master defines a certain level L to divide the tasks. Tasks are the possible permutations of $C-1$ cities in L elements. The granularity G of a task is the number of cities that defines the task sub-tree, this is $G = C - L$. At the execution start-up the Master sends the cities coordinates to every Worker.

A diagram of the possible permutations for five cities (**V**ienna, **G**raz, **L**inz, **B**arcelona, **M**adrid), considering the salesman starts and ends his trip at Vienna, can be seen in Figure 1. The Master can divide this problem into 1 task of level 0 or 4 tasks of level 1 or 12 tasks of level 2 for example. The tasks of the first level would be represented by the cities Vienna and Graz for the first task, Vienna and Linz for the second, followed by Vienna and Barcelona, and Vienna and Madrid. The requirement of returning to the starting city is without detracting from the generality. In this closed cycle the salesman may begin and end in the city who wants.

Knowing the latitude and longitude of two cities on the Earth, it is possible to determine the distance between them in kilometres. The table 1 lists the latitude and longitude of the five cities mentioned previously. Figure 2(a) shows a strictly lower triangular distance matrix where each box contains the Euclidean distance in kilometres between two cities.

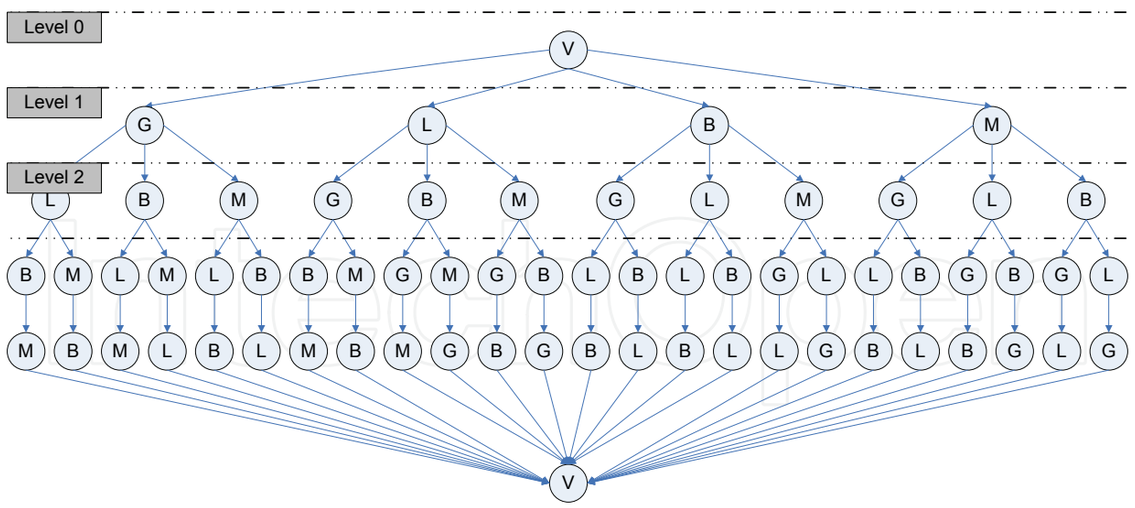


Fig. 1. Possible paths for the salesman considering five cities: **V**ienna, **G**raz, **L**inz, **B**arcelona, **M**adrid

	Latitude	*Longitude*
Barcelona	40° 26' north	3° 42' west
Graz	48° 13' north	16° 22' east
Linz	47° 05' north	15° 22' east
Madrid	48° 19' north	14° 18' east
Vienna	41° 18' north	2° 06' east

Table 1. Latitude and longitude of the five cities

Workers are responsible for calculating the distance of the permutations left in the task and sending to the Master the best path and distance of these permutations. One of the characteristics of the TSP is that once the distance for a path is superior to the already computed minimum distance it is possible to prune this path tree.

Figure 2(b) and Figure 2(c) exhibit the pruning processes for the *GP-TSP* algorithm where each arrow has the distance between the two cities it connects. Analyzing Figure 2(b), the total distance for the first followed path (in the left) is of 3845 km. The distance between Vienna and Barcelona on the second path (in the right) is already of 4737 km. It is then not necessary for the algorithm to keep calculating distances from the city Barcelona on because it is impossible to reach a better distance for this branch. Analyzing the other example, the total distance for the first followed path (in the left of Figure 2(c)) is of 3845 km. Then, the distance between Linz and Barcelona on the second path (in the right of Figure 2(c)) is already of 4839 km. Therefore, it is not necessary for the algorithm to keep calculating distances from the city Barcelona on.

3. TSP knowledge discovery methodology

The scientific experimental knowledge discovery methodology presented here is a first attempt to estimate the performance order of a TSP parallel algorithm. As well as the process of knowledge discovery is certainly not new, it is typical of the experimental

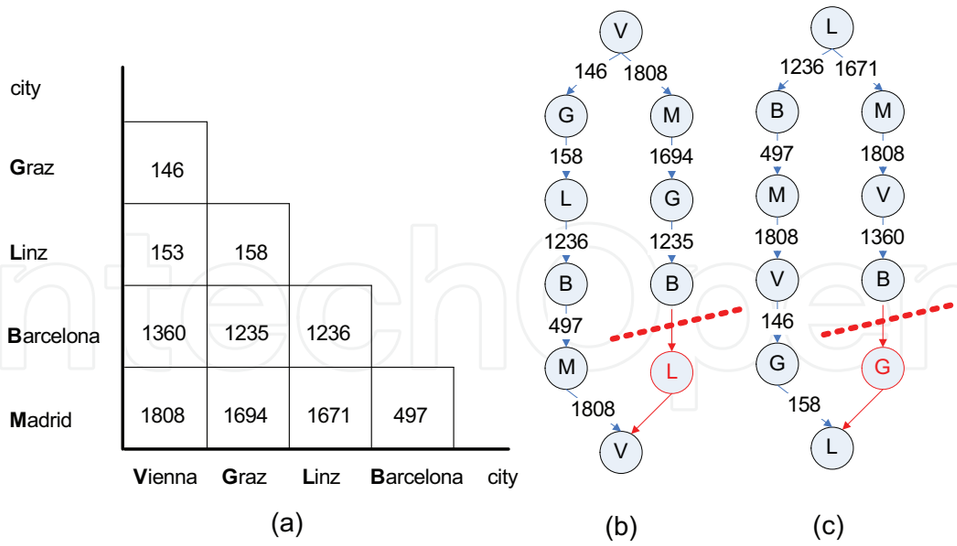


Fig. 2. (a) Matrix of Euclidean distances between cities (in km), (b)-(c) Two pruning processes in the *GP-TSP* algorithm

sciences. An experimental science is based on observation of performing repeated controlled experiments. Before computers were used to automate this process, people involved in math, physics or statistics were using probability techniques to model historical data.
The methodology consists of three main phases. First, the design and composition of experiments to define and improve the TSP asymptotic time complexity. Next, the validation of the built model. Finally, the definition of the TSP asymptotic time complexity.

3.1 Design and composition of experiments to define and improve the asymptotic time complexity

Foremost it is important understanding the application domain and the relevant prior knowledge, and analyzing their behavior step by step, in a deep way. It is a try-and-error method that requires specialists to manually or automatically identify the relevant parameters that can affect the execution time of the algorithm studied. Discovering the proper set of parameters is the basis to obtain a good capacity of prediction.
Designing a well-built experiment involves articulating a goal, choosing an output that characterizes an aspect of that goal and specifying the data that will be used in the study taking into account the worked hypotheses at that time. The experiments must provide a representative sample (a good training data set) first to measure the quality of the model / hypotheses and then to fit the model. After the necessary training data have been defined the TSP parallel algorithm studied must process each experiment obtaining a tour visiting and the execution time invested as output.
The term knowledge discovery in databases (KDD) refers to the process of analyzing data from different perspectives and summarizing it into useful information. Technically, KDD is the process of finding correlations or patterns among dozens of fields in large relational databases. A KDD process, a bold closed curve in Figure 3, involves data preparation, defining a study, reading the data and building a model, understanding the model, and finally predicting. It is an interactive and iterative process, surrounding numerous steps with many decisions that the end-user carries out (Groth, 1998).

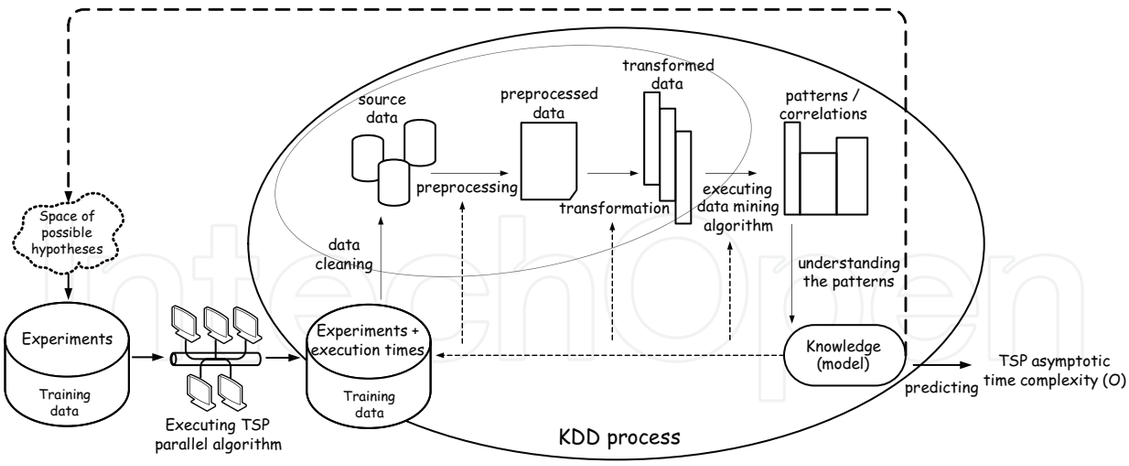


Fig. 3. Knowledge acquisition

Inside the KDD process, a gray closed curve in Figure 3, the stages of data preparation and defining a study surrounds both the decision of choosing between the data mining techniques (classification, regression, clustering, dependency modeling, summarization of data, or change and deviation detection), and also the selection of the data mining algorithm to apply according to the chosen technique.

Regarding the analysis of the problem, a clustering study could be performed to potentially identify groups. Clustering is the process of partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait (often proximity according to some defined distance measure). Therefore, a clustering data-mining tool through k-means algorithm analyzes the measured times and the main parameters values that affect performance in order to summarize these into a useful information. Knowing the main parameters which characterize time complexity, it becomes possible to suspect new hypotheses to restart the process and to produce a subsequent improved time complexity model.

Figure 3 shows the knowledge acquisition process which includes the design of experiments, the execution of the TSP parallel algorithm and the KDD process. There is no doubt that the design of experiments is directly related to the suspected hypotheses. The solid lines in Figure 3 represent the compulsory path to follow in the methodology and the dashed lines represent paths of refinement.

3.2 Validation of the model

A new data set is proposed to be able to validate the created model. Although the validation data set constitutes a hold-out sample, it has not been considered in the building of the model. This enables to estimate the error in the predictions without having the assumption that the execution times follow a particular distribution.

The analytical formulation, together a particular architecture, is used to make predictions for each experiment in the validation data. The quality analysis is a relevant issue in this stage and has to include interest measurements. The prediction for each experiment is then compared to the value of the dependent variable that was actually observed in the validation data obtaining the prediction error. Then the average of the square of these errors enables to compare different models and to assess the accuracy of the model in making predictions.

It is important to bear in mind that every stage in the design of experiments to obtain and improve the asymptotic time complexity is validated. Figure 4 exhibits the entire model validation phase.

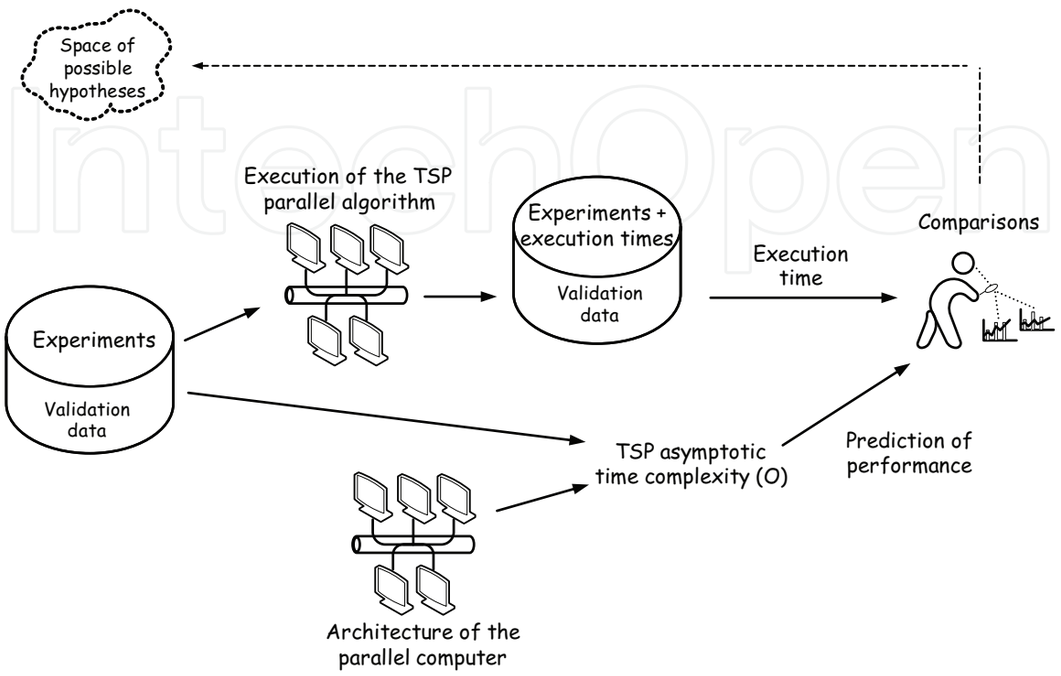


Fig. 4. Model validation phase

3.3 Definition of the asymptotic time complexity

The refined built model allows defining the asymptotic time complexity for the TSP parallel algorithm studied, Figure 5. Then the analytical formulation will be instanced with values coming from a new input data set and a particular parallel computer in order to give a prediction of performance.

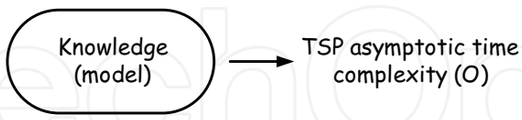


Fig. 5. The final definition of the TSP asymptotic time complexity

The entire TSP knowledge discovery methodology is shown in Figure 6. Every stage in the methodology defined can implicate a backward motion to previous steps in order to obtain extra or more precise information to fit the final model.

4. Analyzing the *GP-TSP* algorithm

Using simple experiments, varying one or two values at a time, it is possible to infer that time required for the parallel *GP-TSP* algorithm depends on certain parameters. Discovering these significant *GP-TSP* input parameters is the main issue of this section. Then, the prediction of *GP-TSP* performance order and two relevant experiments are analyzed.

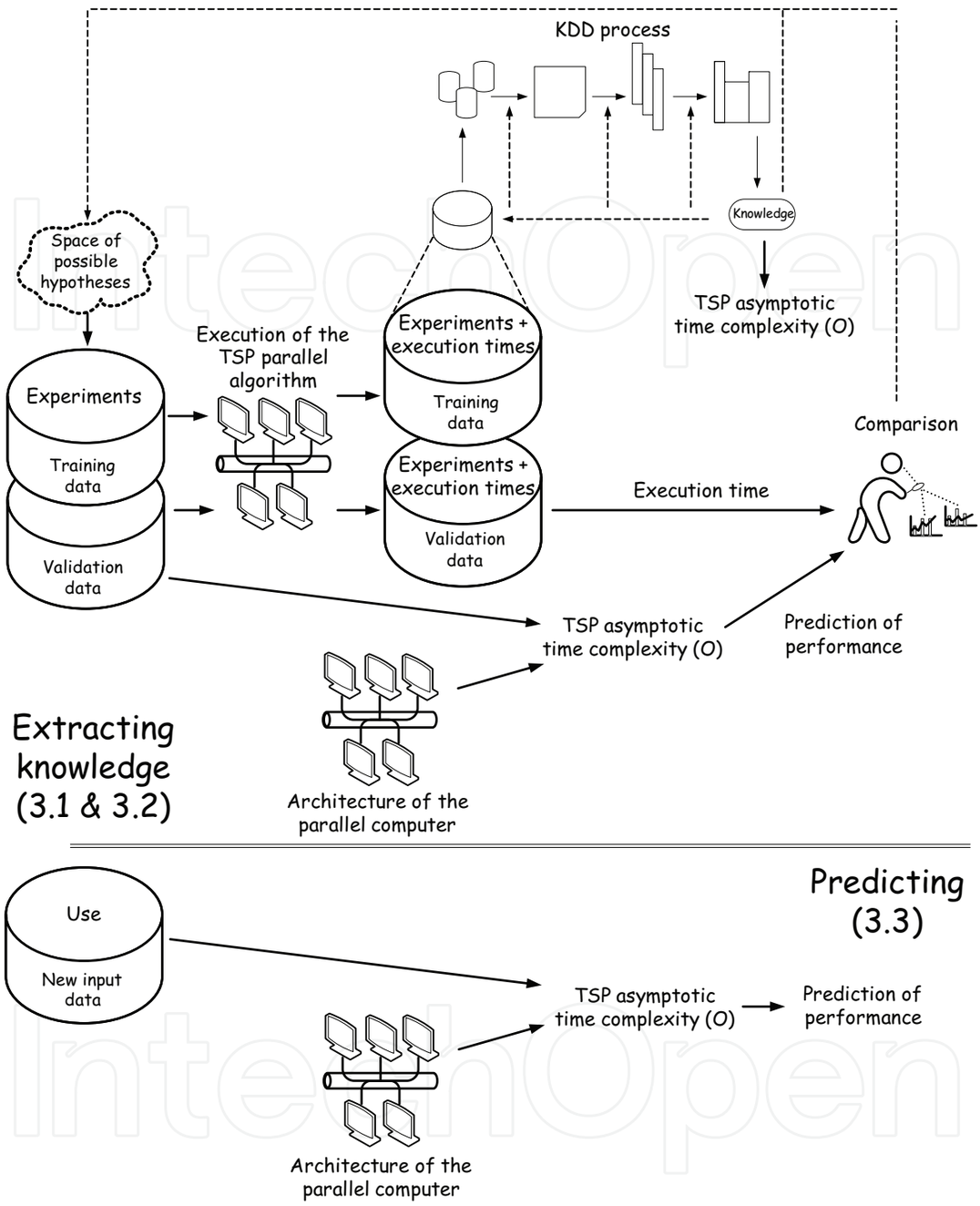


Fig. 6. Performance prediction using the knowledge discovery methodology

4.1 Discovering the significant GP-TSP input parameters

It is clear that the *GP-TSP* execution time depends on the number of processors (P), the number of cities (C), and *other parameters*. Discovering the *other parameters* is the key to obtain a good or an acceptable prediction of performance order. Undoubtedly, the knowledge discovery in databases process (KDD process) has been one of the most profitable stages in the scientific examination. A huge amount of data sets was processed with the only goal of finding

some common properties. First intuitions guided the different tests in order to determine the characteristics, the relationships, and the patterns between the data sets.
As a result of the investigation, right now the sum of the distances from one city to the other cities (*SD*) and the mean deviation of *SDs* values (*MDSD*) are the numerical parameters characterizing the different input data beyond the number of cities (*C*). But how these final parameters have been obtained? Next, it is described the followed way to discover the above mentioned dependencies (*SD* and *MDSD*) and the construction of a model.

4.1.1 First hypothesis → location of the cities (geographical pattern)

Given a number of cities with its pattern of distribution, the initial experiments have provided evidence that times required for the completion of the algorithm are dissimilar. In order to understand the general process, show its progress and results, it has been chosen an example data set to follow along this section. It consists of five different geographical patterns of fifteen cities each one (named $GPat_1$ to $GPat_5$) as it is shown in Figure 7.

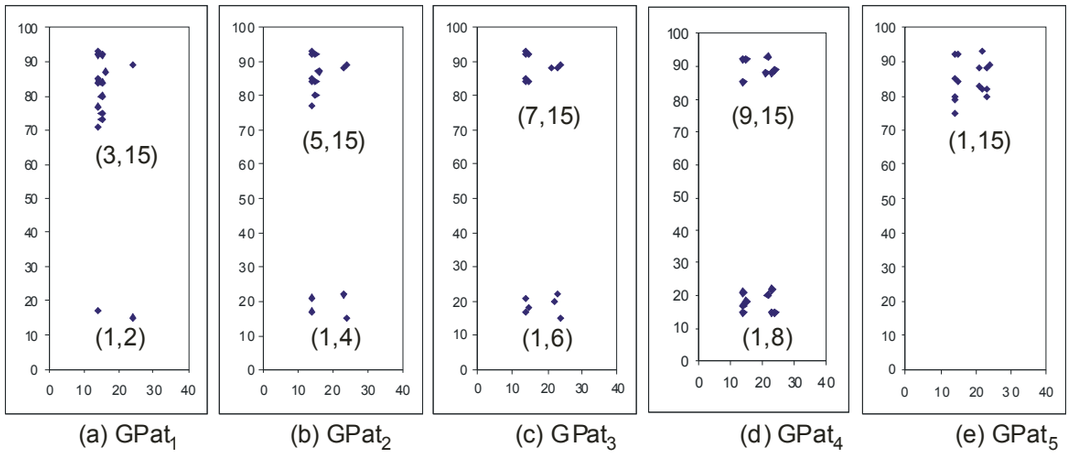


Fig. 7. Five patterns defined for fifteen cities

The *GP-TSP* implementation receives the number of cities (*C*) and their coordinates ((x_1, y_1), …, (x_i, y_i), …, (x_C, y_C)), the level (*L*), and the number of processors (*P*) as input parameters. It behaves recursively searching all possible paths and applying the global pruning strategy whenever it is feasible and, finally, generating the minimal path and the time spent.
Table 2 shows the *GP-TSP* execution times (in sec.) by pattern (columns $GPat_1$ to $GPat_5$) and starting city (1...15) using only 8 nodes of the parallel machine described in Appendix A. It is important to observe the dispersion of times while maintaining constant the number of processors (*P*) and the number of cities (*C*).
Hence before continuing, there are two important concepts to refresh. The main goal of data mining is finding useful patterns and knowledge in data. Besides, clustering is one of the major data mining techniques, grouping objects together into clusters that exhibit internal cohesion (similar execution time) and external isolation. Therefore, in this work, clustering has been applied to discover the internal information and then to decrease the data-dependence. This general action has been done using the well-known k-means clustering algorithm (MacQueen, 1967) included in the Cluster-Frame tool; see Appendix B for extra information about the tool. With the idea of obtaining quite similar groups with respect to the groups (patterns) used at the beginning, *k* was fixed in five (*k* is the number of clusters).

The initial centroids (one for each cluster) were randomly selected by the clustering tool. Figure 8 shows the experiments by cluster in the Cluster-Frame environment.

	Geographical pattern (GPat)									
	1		***2***		***3***		***4***		***5***	
Starting city	***Time spent***	***Assigned cluster***	***Time spent***	***Assigned cluster***	***Time spent***	***Assigned cluster***	***Time spent***	***Assigned cluster***	***Time spent***	***Assigned cluster***
1	216.17	1	36.50	3	15.34	2	10.51	4	8.03	5
2	214.44	1	36.82	3	15.19	2	10.49	4	7.82	5
3	77.25	1	38.09	3	15.57	2	10.02	4	7.71	5
4	72.64	1	37.29	3	15.02	2	10.30	4	7.91	5
5	70.94	1	18.54	2	15.84	2	10.41	4	7.83	5
6	74.21	1	17.83	2	15.24	2	10.24	4	7.71	5
7	75.59	1	18.16	2	10.31	4	10.36	4	7.93	5
8	73.72	1	18.03	2	10.34	4	10.26	4	7.87	5
9	69.47	1	17.79	2	10.27	4	9.98	4	8.14	5
10	74.96	1	17.48	2	10.23	4	9.88	4	8.22	5
11	75.89	1	17.07	2	10.24	4	9.85	4	8.04	5
12	70.17	1	17.39	2	10.28	4	9.87	4	8.12	5
13	73.73	1	18.10	2	10.36	4	9.88	4	7.98	5
14	70.87	1	17.37	2	10.17	4	9.95	4	8.02	5
15	73.30	1	18.00	2	10.32	4	9.97	4	7.78	5
Mean	92.23		22.97		12.32		10.14		7.94	

Table 2. *GP-TSP* execution times (in sec.) and assigned cluster by k-means algorithm

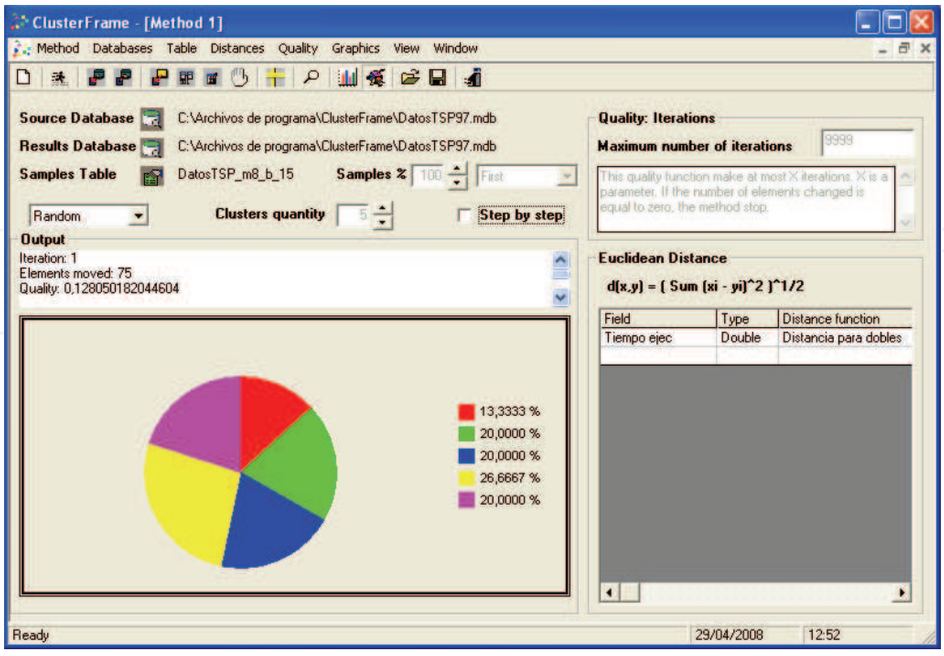


Fig. 8. Cluster-Frame environment

The k-means algorithm aims at minimizing a squared error function. In Equation (1), it is presented the widely used objective function with n data points and k disjoint subsets

$$\sum_{j=1}^{k}\sum_{i=1}^{n} | x_i^{(j)} - c_j |^2 \quad (1)$$

where $|x_i^{(j)}-c_j|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centroid c_j. The entire function is an indicator of the distance of the n data points from their respective cluster centroids.

Table 2 show the assigned cluster for each experiment after executing k-means algorithm. For the clusters 1 to 5, the centroids values were 92.23 sec., 16.94 sec., 37.17 sec., 10.19 sec., and 7.94 sec., respectively.

The quality evaluation involves the validation of the above mentioned hypothesis. For each experiment, the assigned cluster was confronted with the defined graphic pattern previously. The percentage of hits expresses the capacity of prediction. A simple observation is that the execution times were clustered in a similar way to patterns fixed at starting, see Figure 7. In this example, the capacity of prediction was near of 75% (56 hits on 75 possibilities). There was a close relationship between the patterns and the execution times.

Conclusions: The initial hypothesis for the *GP-TSP* has been corroborated; the capacity of prediction has been greater than 75% for the full range of experiments worked. The remaining percentage has given evidence of the existence of other significant parameters. Therefore, a deep analysis of results revealed an open issue remained for discussion and resolution, the singular execution times by pattern. Another major hypothesis was formulated. At this stage, the asymptotic time complexity was defined as *O(P, C, pattern)*.

4.1.2 Second hypothesis → location of the cities and starting city

The example data set is the same used previously. Comparing each chart of Figure 7 with its corresponding column in Table 2 it is easy to infer some important facts. The two far cities (1, 2) in Figure 7(a) correspond with the two higher time values of starting city 1 and 2 in Table 2($GPat_1$). The four far cities (1, 4) in Figure 7(b) correspond with the four higher execution time values of starting city 1 to 4 in Table 2($GPat_2$). The six far cities in Figure 7(c) correspond with the six higher time values of Table 2($GPat_3$). The cities in Figure 7(d) are distributed among two zones; therefore, the times turn out to be similar enough, see Table 2($GPat_4$). Finally, the cities in Figure 7(e) are closed enough; in consequence, the times are quite similar, see Table 2($GPat_5$).

An additional important observation is that the mean of execution times by geographical pattern decreases as the cities approach, see again Table 2.

Conclusions: Without doubt, the location of the cities and the starting city (C_1) play an important role in execution times; the hypothesis has been corroborated. However, an open issue remained for discussion and resolution: how to relate a pattern (in general) with a numerical value which means execution time. This relationship would be able to establish a numerical characterization of patterns. On this basis, an original hypothesis was formulated. At this point, the *GP-TSP* asymptotic time complexity was redefined as $O(C, P, pattern, C_1)$.

4.1.3 Third hypothesis → sum of distances and mean deviation of sum of distances

What parameters could be used to quantitatively characterize different geographical patterns in the distribution of cities? In graph theory, the distance of a vertex p, $d(p)$, of such

a connected graph G is defined by $d(p) = \Sigma\, d(p, q)$ where $d(p, q)$ is the distance between p and q and the summation extends over all vertices q of G. This measure is an inverse measure of centrality. Therefore, following the ideas previously mentioned, the sum of the distances from one city to the other cities (SD_j, as it is shown in Equation 2), and the mean deviation of SDs values ($MDSD$) are the worked inputs right now. As greater is the sum of the distances, the lower is the centrality.

$$\forall j : 1 \le j \le C \quad SD_j = \sum_{i=1}^{C} \sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2} \tag{2}$$

The SD value is an index time. If a j particular city is very remote of the others, its SD_j will be considerably greater to the rest and consequently its execution time will also grow. This can be observed in Table 3.

Why is it needed to consider $MDSD$ in addition to SD as a significant parameter? Quite similar SD values from the same geographical pattern (same column) of Table 3 imply similar execution times. The SD_4 and SD_{10} values for the geographical pattern 1 are 230.11 and 234.84, respectively. Then, their execution times are similar 72.64 sec. and 74.96 sec. (labelled with the symbol ◊). Instead, this relation is not true considering similar SD values coming from different geographical patterns (different columns). The SD_3 value for geographical pattern 1 and the SD_{10} value for geographical pattern 2 are similar (315.51 and

	Geographical pattern (GPat)									
	1		***2***		***3***		***4***		***5***	
Starting city	***Time spent***	***SD***	***Time spent***	***SD***	***Time spent***	***SD***	***Time spent***	***SD***	***Time spent***	***SD***
1	216.17	853.94	36.50	746.10	15.34	664.60	10.51	643.75	8.03	148.74
2	214.44	887.44	36.82	740.49	15.19	649.14	10.49	635.54	7.82	104.16
3	* **77.25**	* **315.51**	38.09	820.63	15.57	707.70	10.02	555.70	7.71	141.15
4	◊ **72.64**	◊ **230.11**	37.29	789.80	15.02	678.07	10.30	599.99	7.91	103.35
5	70.94	226.88	18.54	345.83	15.84	643.65	10.41	611.45	7.83	111.79
6	74.21	244.56	17.83	330.76	15.24	638.04	10.24	595.58	7.71	102.81
7	75.59	276.09	18.16	369.56	10.31	467.99	10.36	592.68	7.93	111.28
8	73.72	294.62	18.03	383.38	10.34	490.55	10.26	639.61	7.87	147.14
9	69.47	233.53	17.79	370.10	10.27	491.52	9.98	574.23	8.14	123.19
10	◊ **74.96**	◊ **234.84**	* **17.48**	* **323.12**	10.23	446.48	9.88	578.78	8.22	172.52
11	75.89	259.19	17.07	332.87	10.24	477.42	9.85	544.61	8.04	124.64
12	70.17	234.22	17.39	325.19	10.28	449.03	9.87	534.91	8.12	131.68
13	73.73	306.99	18.10	383.11	10.36	504.79	9.88	530.72	7.98	109.78
14	70.87	239.19	17.37	327.02	10.17	451.21	9.95	574.97	8.02	124.96
15	73.30	295.27	18.00	372.00	10.32	494.09	9.97	534.36	7.78	96.29
MDSD		140.94		165.47		90.60		31.56		16.78

Table 3. *GP-TSP* execution times (in sec.) and sum of the distances from each starting city

323.12, respectively) but the execution times are completely dissimilar 77.25 sec. and 17.48 sec. (labelled with the symbol *). The reason is due to the different between the *MDSD* values of geographical pattern 1 and 2.

Conclusions: It is important to emphasize that the *GP-TSP* algorithm obtains good results of prediction. The asymptotic time complexity for the *GP-TSP* algorithm should be defined as *O(P, C, SD, MDSD)*. Another important fact has been reached beyond was originally sought. Choosing the *j* city which has minimum SD_j associated value, it is possible to obtain the exact TSP solution investing less amount of time. Much better results it would be reached if the algorithm begins considering the closer *L* cities to *j* city.

4.2 Predicting GP-TSP performance order

The *GP-TSP* has a time complexity of *O(P, C, SD, MDSD)*. The analytical formulation allows making predictions for a new data set on a particular parallel computer. Figure 9 shows the prediction framework.

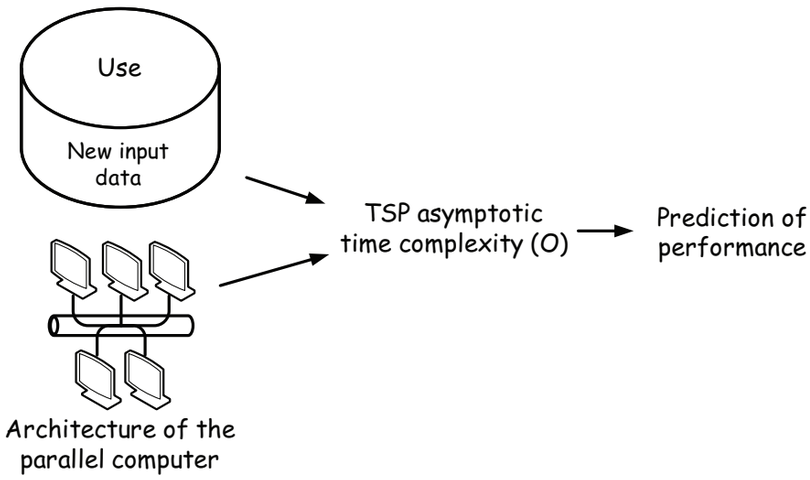


Fig. 9. The prediction of performance framework

4.3 Two relevant GP-TSP experiments

Additional TSP experiments have been performed to verify certain hypotheses. Some of them have shown how important is the geographical pattern of the cities instead of knowing their coordinates. Other experiments which follow a specific pattern have helped to confirm the strong compliance of our hypotheses. Due to the significance, these two groups of experiments were chosen to be developed in this section.

4.3.1 Importance of the geographical pattern

Making geometric transformations (shifting, scaling, and rotation) to well-known patterns is without no doubt a trivial test. This is an excellent case study for understanding the importance of geographical pattern. Applying each one of the transformations to a set of cities, similar execution times are expected executing the same algorithm. This leading to conclude, the time required to reach the solution of the *GP-TSP* algorithm is invariant to certain transformations into the geographical patterns.

The coordinates of a city shifted by Δx in the x-dimension and Δy in the y-dimension are given by

$$x' = x + \Delta x \qquad y' = y + \Delta y \tag{3}$$

where x and y are the original and x' and y' are the new coordinates.
The coordinates of a city scaled by a factor S_x in the x-direction and y-direction (the city is enlarged in size when S_x is greater than 1 and reduced in size when S_x is between 0 and 1) are given by

$$x' = xS_x \qquad y' = yS_y \tag{4}$$

The coordinates of a city rotated through an angle θ about the origin of the coordinate system are given by

$$x' = x\cos\theta + y\sin\theta \qquad y' = -x\sin\theta + y\cos\theta \tag{5}$$

An example set consisting of fifteen cities is chosen from the historical database. The execution times were obtained using 32 nodes of the parallel machine described in Appendix A. The shifting and rotation transformations are obtained interchanging x-coordinate by y-coordinate, and the scaling transformation dividing by 2 both coordinates. All these patterns are shown in Figure 10.

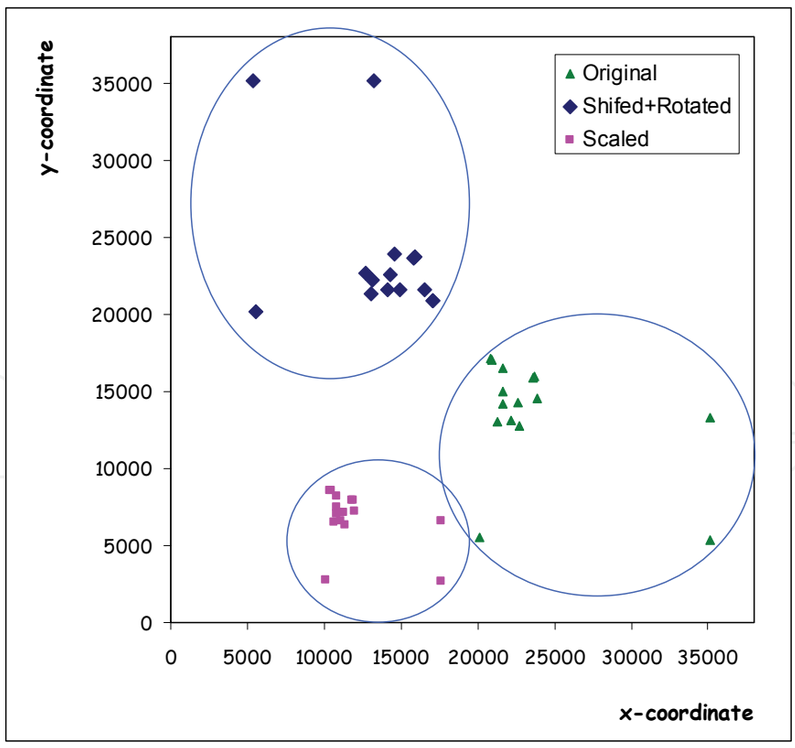


Fig. 10. A historical pattern consisting of fifteen cities. Besides, the same pattern shifted and rotated, and then the pattern scaled

Table 4 exhibits the execution times for the example set starting by each one of the cities. Analyzing the values by row, the historical execution times and the execution times of the geometric transformations for an experiment (row) are quite similar as it was to be expected. For all the experiments, the mean deviation was smaller than 2%.

Starting city	*Pattern*			*Mean deviation*
	Historical	*Shigted+Rotated*	*Scaled*	
1	46.25	48.52	47.30	0.78
2	100.30	105.60	102.77	1.81
3	73.48	76.34	74.52	1.04
4	32.92	34.52	33.75	0.54
5	30.83	31.96	31.35	0.39
6	30.49	31.92	31.22	0.48
7	31.77	33.00	32.21	0.45
8	30.10	31.06	30.43	0.35
9	31.08	32.13	31.92	0.42
10	30.98	32.24	31.60	0.42
11	29.94	31.09	30.36	0.42
12	30.33	31.53	30.85	0.42
13	31.45	32.82	32.14	0.46
14	32.67	33.44	32.53	0.37
15	32.49	33.49	32.89	0.36

Table 4. Comparison of execution times (in sec.) using 32 nodes for the three patterns plotted in Figure 9

4.3.2 Limit case

A singular case is to have the cities uniformly distributed in a circumference, see an example in Fig. 11. As the *MDSD* value will be near to 0, similar execution times are expected. The idea is considering a limit case in order to confirm the hypothesis with respect to the *MDSD* value and the geographical pattern.

Table 5 exhibits a comparative study of *GP-TSP* behaviour; the means and means deviations of execution times of different number of cities uniformly distributed in each circumference pattern are shown. The number of cities is between 15 and 25. As it can be appreciated in Table 5, there is a progressive increase in the mean times. For every circumference, the execution times were quite similar starting by each one of the cities. The mean deviations were smaller than 4%.

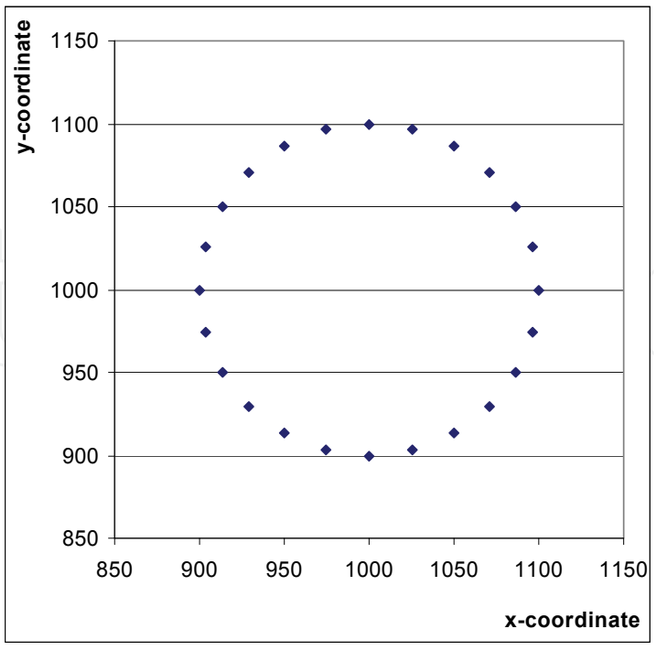


Fig. 11. A circumference pattern composed of 24 uniformly distributed cities

#Cities	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*
Mean	12.71	17.47	23.42	32.93	42.95	54.94	68.67	129.53	367.29	1085.57	2957.15
Mean deviation	0.03	0.04	0.08	0.08	0.07	0.10	0.10	0.11	0.30	2.12	3.03

Table 5. Mean and mean deviation of execution times (in sec.) using 32 nodes by the number of cities that are present in each circumference pattern

5. Conclusions

This chapter introduces a knowledge discovery methodology to estimate the performance order of a hard data-dependent parallel algorithm that solves the traveling salesman problem. It is important to understand that the parallel performance achieved depends on several factors, including the application, the parallel computer, the data distribution, and also the methods used for partitioning the application and mapping its components onto the architecture.

Briefly, the general knowledge discovery methodology begins by designing a considerable number of experiments and measuring their execution times. A well-built experiment guides the experimenters in choosing what experiments actually need to be performed in order to provide a representative sample. A data-mining tool then explores these collected data in search of patterns and/or relationships detecting the main parameters that affect performance. Knowing the main parameters which characterise performance, it becomes possible to suspect new hypotheses to restart the process and to produce a subsequent improved time complexity model. Finally, the methodology predicts the performance order for new data sets on a particular parallel computer by replacing a numerical identification.

A TSP parallel implementation (called *GP-TSP*) has been deeply studied. The *GP-TSP* algorithm analyzes the influence of indeterminism in performance prediction, and also shows the usefulness and the profits of the methodology. Their execution time depends on the number of cities (*C*), the number of processors (*P*), and other parameters. As a result of the investigation, right now the sum of the distances from one city to the other cities (*SD*) and the mean deviation of *SDs* values (*MDSD*) are the numerical parameters characterizing the different input data beyond the number of cities. The followed way to discover this proper set of parameters has been exhaustively described.
The defined methodology for performance modelling is applicable to other related problems such as the knapsack problem, the graph partition, the bin packing, the motion planning, among others.

Appendix

A. Specification of the parallel machine

The execution has been reached with a 32 node homogeneous PC (Cluster Pentium IV 3.0GHz., 1Gb DDR-DSRAM 400Mhz., Gigabit Ethernet) at the Computer Architecture and Operating Systems Department, University Autonoma of Barcelona. All the communications have been accomplished using a switched network with a mean distance between two communication end-points of two hops. The switches enable dynamic routes in order to overlap communication.

B. Characteristics of Cluster-Frame environment

Cluster-Frame is a dynamic and open environment of clustering (Fritzsche, 2007). It permits the evaluation of clustering methods such as K-Means, K-Prototypes, K-Modes, K-Medoid, K-Means^{+}, K-Means^{++} for the same data set. Using Cluster-Frame, the results reached applying different methods and using several parameters can be analyzed and compared.

6. References

Alizadeh, F.; Karp, R.; Newberg, L. & Weisser, D. (1993). Physical mapping of chromosomes: A combinatorial problem in molecular biology. *Symposium on Discrete Algorithms,* pp. 371-381, ACM Press.

Aras, N.; Altinel, I. & Oommen, J. 2003. A kohonen-like decomposition method for the euclidean traveling salesman problem-knies/spl i.bar/decompose. *IEEE Transactions on Neural Networks*, Vol. 14, No.4, pp. 869–890.

Arkin, E.; Chiang, Y. ; Mitchell, J.; Skiena, S. & Yang, T. (1996). On the Maximum Scatter TSP, *In Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 97)*, pp. 211-220, ACM New York.

Balas, E. (1989). The Prize Collecting Traveling Salesman Problem. *Networks,* Vol.19, pp. 621-636.

Barvinok, A.; Tamir, A.; Fekete, S.; Woeginger, G; Johnson, D. & Woodroofe, R. (2003). The Geometric Maximum Traveling Salesman Problem. *Journal of the ACM,* Vol.50, No.5, pp. 641-664.

Bland, R. & Shallcross, D. (1989). Large Traveling Salesman Problems Arising from Experiments in X-ray Crystallography: a Preliminary Report on Computation. Operations Research Letters, Vol.8, pp. 125-128.

Christofides, N. (1985). Vehicle Routing. *N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, Combinatorial Optimization,* pp. 315-338, Wiley, Chichester, UK.

Duman, E. & Or, I. (2004). Precedence constrained TSP arising in printed circuit board assembly. *International Journal of Production Research,* Vol.42, No.1, pp. 67-78, 1 January 2004, Taylor and Francis Ltd.

Fritzsche, P. (2007). ¿Podemos Predecir en Algoritmos Paralelos No-Deterministas?, *PhD Thesis, University Autonoma of Barcelona, Computer Architecture and Operating Systems Department*, Spain. http://caos.uab.es/

Garey, M.; Graham, R. & Johnson, D. (1976). Some NP-complete geometric problems, STOC '76: Proceedings of the eighth annual ACM symposium on Theory of computing, pp. 10-22, Hershey, Pennsylvania, United States, ACM, New York, NY, USA.

Gilmore, P. & Gomory, R. (1964). Sequencing a One-State-Variable Machine: A Solvable Case of the Traveling Salesman Problem. Operations Research, Vol.12, No.5, pp. 655-679.

Golden, B.; Levy, L. & Vohra, R. (1987). The Orienteering Problem. *Naval Research Logistics,* Vol.34, pp. 307-318.

Groth, R. (1998) Data mining: a hands-on approach for business professionals, Prentice Hall PTR.

Gutin, G. & Punnen, P. (2006). *The Traveling Salesman Problem and Its Variations,* Springer, 0-387-44459-9, New York.

Johnson, O. & Liu, J. (2006). *A Traveling Salesman Approach for predicting protein functions,* Source Code for Biology and Medicine, Vol.1, pp. 1-7.

Karp, R. (1972). Reducibility among combinatorial problems: In Complexity of Computer Computations. *Plenum Press,* pp. 85-103. New York.

Korostensky, C. & Gonnet, G. (2000). Using traveling salesman problem algorithms for evolutionary tree construction. *BIOINF: Bioinformatics,* Vol.16, No.7, pp. 619-627.

Lenstra, J. & Kan, A. (1975). Some simple applications of the Travelling Salesman Problem. Operations Research Quarterly, Vol.26, No.4, pp. 717-732.

Lilja, D. (2000). *Measuring computer performance: a practitioner's guide,* Cambridge University Press, ISBN: 0-521-64105-5, New York, NY, USA.

MacQueen, J. (1967). Some Methods for Classification and Analysis of MultiVariate Observations, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol.1, pp. 281-297, L. M. Le Cam and J. Neyman, University of California Press.

Miller, D. & Pekny, J. (1991). Exact Solution of Large Asymmetric Traveling Salesman Problems. *Science,* Vol.251, pp. 754-761.

Miller, R. & Boxer, L. (2005). *Algorithms Sequential and Parallel: A Unified Approach,* Charles River Media. Computer Engineering Series, 1-58450-412-9.

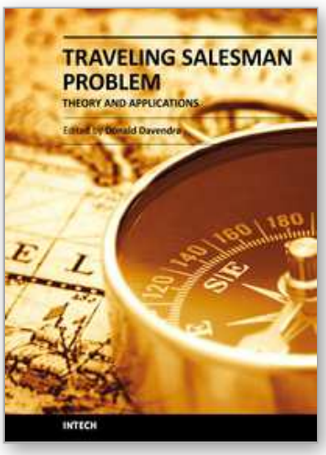
Pepper, J.; Golden, B. & Wasil, E. (2002). Solving the travelling salesman problem with annealing-based heuristics: a computational study. *IEEE Transactions on Man and Cybernetics Systems*, Part A, Vol. 32, No.1, pp. 72–77.

Ratliff, H. & Rosenthal, A. (1983). Order-Picking in a Rectangular Warehouse: A Solvable Case for the Traveling Salesman Problem. Operations Research, Vol.31, No.3, pp. 507-521.

Sankoff, D. & Blanchette, M. (1997). The median problem for breakpoints in comparative genomics, *Proceedings of the 3rd Annual International Conference on Computing and Combinatorics (COCOON'97)*, Vol.1276, pp. 251-264, New York.

Tsai, H.; Yang, J. & Kao, C. (2002). Solving traveling salesman problems by combining global and local search mechanisms, *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, Vol.2, pp. 1290–1295.

TSP page (2010). http://www.tsp.gatech.edu/.



Traveling Salesman Problem, Theory and Applications
Edited by Prof. Donald Davendra

ISBN 978-953-307-426-9
Hard cover, 298 pages
Publisher InTech
Published online 30, November, 2010
Published in print edition November, 2010

This book is a collection of current research in the application of evolutionary algorithms and other optimal algorithms to solving the TSP problem. It brings together researchers with applications in Artificial Immune Systems, Genetic Algorithms, Neural Networks and Differential Evolution Algorithm. Hybrid systems, like Fuzzy Maps, Chaotic Maps and Parallelized TSP are also presented. Most importantly, this book presents both theoretical as well as practical applications of TSP, which will be a vital tool for researchers and graduate entry students in the field of applied Mathematics, Computing Science and Engineering.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen