Variable Neighborhood Search for Non-deterministic Problems

Marco Antonio Cruz-Chávez[1], Alina Martínez-Oropeza[1], Jesús del Carmen Peralta-Abarca[2], Martín H. Cruz-Rosales[3], Martín Martínez-Rangel[4]

[1]Engineering and Applied Science Research Center, [2]FCQeI, [3]FC,[4] FCAeI. UAEM
Av. Universidad 1001, Col. Chamilpa, 62209, Cuernavaca, Morelos, MÉXICO

mcruz[1],alinam[1]@uaem.mx

Abstract. A comparative analysis of several neighborhood structures is presented, including a variable neighborhood structure, which corresponds to a combination of the neighborhood structures evaluated in this paper. The performance of each neighborhood structure was tested using large random instances generated in this research and well-known benchmarks such as the Classical Symmetric Traveling Salesman Problem and the Unrelated Parallel Machines Problem. Experimental results show differences in the performance of the variable neighborhood search when it is applied to problems with differing complexity. Contrary to reports in literature about variable neighborhood searches, its performance varies according to the complexity of the problem.

Keywords: Population, Diversity, Hamming Distance, Population-based Algorithm, Individual.

1 Introduction

For many decades, heuristic methods have been widely used to undertake a large variety of not only theoretical problems, but practical ones too. These problems are classified by the complexity theory into P (Polynomial time), NP (Non-deterministic Polynomial time) and NP-Complete based on their characteristics and nature. NP-Complete problems are the most difficult problems [1], which become intractable in the worst case for large test problems. Because deterministic methods are not enough to solve them, it is necessary to use non-deterministic methods to bind the problem, in an attempt to get high-quality solutions, without the guarantee of optimality.

One of the most frequently used heuristics is local search, which involves the use of neighborhood structures. Sometimes the use of a sole heuristic is not enough to find good solutions for hard problems because the solution space is very complex. In such cases, the neighborhood structures have shown themselves to be efficient search methods for these problems. Recently, a new type of neighborhood structure, better-known as the Variable Neighborhood Structure (VNS), has been applied to several

optimization problems because of its good performance. Moreover, it has been shown to be an efficient method to use when searching for approximated solutions.

There is some research in the literature about variable neighborhood search, as they are referred to in this paper. In [2], the authors present a two phase hybrid approach; the structure combines a VNS in the first phase with an iterated local search in the second phase, while always accepting the best solutions. The variable neighborhood search involves 13 different neighborhood structures, which are randomly selected during execution. Experimental results show the algorithm is competitive with other approaches in literature. For nine data sets, it obtained one improved and eight equal solutions.

Another approach, proposed by [3], is a VNS which is implemented in a local search algorithm. Some modifications of this approach are presented. VNS and its variants were tested in five problems: Travelling Salesman Problem (TSP), p-median Problem (PM), Multi-source Weber Problem (MW), Minimum Sum-of-squares Clustering Problem (MSSCC), and Bilinear Programming Problem with Bilinear Constraints (BBLP). It showed competitive results, especially for the PM and MW problems. In [2], a hybrid approach is presented that combines a variable neighborhood search in the first phase with an iterated local search in the second phase, which always accepts the best solutions for the Attribute Reduction in Rough Set Theory. The approach was tested in over 13 well-known datasets. Experimental results demonstrate that it produces solutions competitive with the best techniques.

This research was motivated by the continuous need to find high-quality solutions for important combinatorial problems, such as TSP and UPMP, because in Meta heuristics, the local search is the most time-consuming procedure. Therefore, in this research, a hybrid local search is applied to an NP and an NP-Complete problem to observe its performance in different complexity problems, under the same conditions.

Experimental results show that the good performance of a variable neighborhood search depends on the search space complexity of the problem. High quality solutions are obtained for CSTSP, which is classified as an NP-Complete problem, but poor quality solutions are obtained for UPMP, which is a less complex NP problem. The contribution of this research is the finding that the performance of local search depends on the hardness of the problem. Contrary to what one might expect, the performance is better for the NP-complete problem than for the NP problem, both of which are studied in this paper.

This paper is organized as follows. Section two and three present an introduction to the complexity problems undertaken, which are the Classical Symmetric Traveling Salesman Problem and the Unrelated Parallel Machines Problem. Section four describes the neighborhood structures tested in this research. Section five details the proposed VNS. Section six explains the statistical analysis performed on the obtained results and compares the results of each structure. Finally, section seven present conclusions.

2 Classical Symmetric Traveling Salesman Problem

The Classical Symmetric Traveling Salesman Problem (CSTSP) is a discrete optimization problem [1, 4], classified as an NP-complete problem [1] due to its complexity and nature. The aim of the Classical Symmetric Traveling Salesman Problem is to minimize the total travel distance when visiting all the cities exactly once and returning to the home town [5]. A graph G = (V, E) consists of a finite set V of vertices, identifying the cities, and a finite multiset E of edges or distances between cities. The problem involves unordered pairs (i, j) of cities, where the same city must not be visited more than once and the total travel distance is minimized. The tour has a beginning city and an ending one. Therefore, E = {(i, j): i, j ∈ V,} and cij is the cost (distance) associated with the edge (i, j). The mathematical formulation of the integer linear programming model is described in (1 to 4). [6], shows the objective function in (1), where the aim is to minimize the total travel cost and is based on a set of constraints (2 to 4), which must be met to obtain feasible solutions. The set of constraints in (2) specifies that only city i can be reached from city j. The set of constraints in (3) specifies that only city j can be reached from city i. The last set of constraints in (4) ensures that all the cities have been visited.

$$\min f = \sum_{(i,j)\in E}^{m} C_{ij}X_{ij} \tag{1}$$

$Subject\ to$:

$$\sum_{\{C_j:(i,j)\in E\}} X_{ij} = 1 \qquad \forall i \in E \tag{2}$$

$$\sum_{\{C_i:(i,j)\in E\}} X_{ij} = 1 \qquad \forall j \in E \tag{3}$$

$$\sum_{\{(i,j)\in E, i\in S, j\in S\}} X_{ij} \leq |S| - 1 \quad to \quad S \subset V, 2 \leq |S| \leq |V| - 2 \quad \forall i \tag{4}$$

3 Unrelated Parallel Machines Problem

The Unrelated Parallel Machines Problem (UPMP) is a variant of the classical Job Shop Scheduling Problem (JSSP) relaxed to get a mapping of UPMP [7]. The UPMP is classified as NP [8]. The UPMP can be described as the set *J={1,2,...,n}* of *n* independent jobs that have to be scheduled in *K={1, 2, ..., n}* positions corresponding to *I={1,2,...,m}* unrelated parallel machines that process jobs at different rates, meeting certain constraints to obtain feasible solutions according to the objective function. It is done with the goal to minimize the total completion time of processing all the jobs. According to this, any job can be processed in any machine, and any ma-

chine can process any job, but the processing time depends on the machine and position of the assigned job. It takes into account the basic constraints of the problem which are shown in the mathematical formulation (5 to 8) [16], and involves a penalization according to the assigned position. This penalization forces the job *j* to be scheduled in the first position, in an attempt to reduce the processing time.
The features mentioned above show many similarities to the requirements of manufacturing systems currently used in industry. As demand increases, the machinery requirements grow, so enterprises have to acquire new equipment. This is the main reason why machines have different capacities, and it is a central part of the problem. Capacities are considered to try to ensure efficient scheduling while avoiding overspending on equipment and machinery.

$$\min f = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{n} kP_{ij}X_{ikj} \tag{5}$$

$Subject\ to$:

$$\sum_{i=1}^{m}\sum_{k=1}^{n} X_{ikj} = 1 \qquad j = 1, \dots, n \tag{6}$$

$$\sum_{j=1}^{n} X_{ikj} \leq 1 \qquad i = 1, \dots, m \quad k = 1, \dots, n \tag{7}$$

$$X_{ikj} \in \{0, 1\} \qquad i = 1, \dots, m \;\; k = 1, \dots, n, j = 1, \dots, n \tag{8}$$

The mathematical formulation presents the objective function in (5), which minimizes the total completion time of processing all the jobs. The processing time of job j depends on the machine and position where it was assigned, and the processing time kPij directly contributes to the value of the objective function.
The set of constraints represented by (6) ensures that each job will be processed only once. Constraint (7) guarantees that each position k processes at most one job j. The last set of constraints (8) shows that a certain job was scheduled in a certain position in a certain machine. Therefore, xijk can only take binary values, xijk = 1 if the job was assigned, and 0 otherwise.

4 Neighborhood Structures

A neighborhood structure is a technique implemented with an algorithm in order to exploit the solution neighborhood, with the intent to improve the quality of that solution, according to the objective function of the undertaken problem. A critical aspect of designing some optimization algorithms is choosing an appropriate neighborhood structure. This allows for better exploitation of the solution space, according to the algorithm and the problem, because the procedure will help search for new and improved solutions.

A neighborhood structure is determined by the criterion of neighbor selection, that is, the movement σ carried out to reach a solution s' from the current solution s. This procedure is performed iteratively while the selection criterion if fulfilled.
In this research, four different neighborhood structures were applied to local search for the CSTSP, and the UPMP. The neighborhood structures used for CSTSP and UPMP are explained as follows:

- **An Adjacent Pair.** An array position *num* is randomly selected by the neighborhood structure. According to the features of the structure, that position is permutated with the position *num + 1*, and a new neighboring solution s' [9, 10, 11] is obtained. The new solution is then evaluated, according to the objective function.

- **A Random Pair.** Similarly, this neighborhood structure performs a single permutation between two different positions of a solution. The difference from the previously described structure is the type of movement implemented to reach a neighboring solution. In this case, two positions of the solution array are randomly selected (num1 and num2) [1, 10, 12]. These positions must be different and not adjacent. If the condition is fulfilled, the permutation is performed.

- **Two Adjacent Pairs.** In this case, the complexity of the neighborhood structure has increased. This technique applies two permutations, unlike the previous two structures. For this structure, it is necessary to generate two random numbers which correspond to positions in the current solution. The selected positions, num and num1, must be different and non-adjacent. The movement is performed with the position adjacent [13] to the selected one; num permutes with num+1, and num1 with num1+1. In this case there were two randomly selected jobs *j*, which were non-adjacent. The jobs are then placed in the next array position, and they are permuted in pairs, resulting in a new neighboring solution.

- **Two Random Pairs.** This neighborhood structure increases in complexity, because it generates four random numbers (num1, num2, num3, and num4) which have to meet certain characteristics before being taken. The constraints include the numbers being different and non-adjacent. Once the positions are selected, permutations take place, performing swaps in pairs [12, 14, 15]. When applying this neighborhood structure, as compared to the previous structures, the difference is greater between the initial solution and the neighboring one. This indicates that the neighborhood size is larger than in the other cases.

5 Variable Neighborhood Search

A variable neighborhood is a technique that incorporates specific characteristics of more than two different neighborhoods. In the literature, some researchers have developed approaches to exploit this concept, due to its improved performance in comparison to basic neighborhood structures.

The development of the variable neighborhood in this paper was based on the research of [3, 7, 2], their obtained results of the neighborhood structures, and the previous explanation of the CSTSP. This paper presents a more straightforward variable neighborhood search, which incorporates basic and low-complexity searches. This change attempts to optimize the time required to perform the local search, because it is very time-consuming to conduct a neighborhood search.

The variable neighborhood search is able to handle variable neighborhood-sizes, due to the random interaction of the structures during the execution time, which improves the exploitation of the solution space. The neighborhood structures presented in this research, including the variable neighborhood structure, were tested for the CSTSP and the UPMP, which are contrasting problems both in concept and in complexity. In both cases the neighborhood structures were applied to local search, although in the case of UPMP, the local search procedure was implemented to improve the metaheuristic Ant Colony. The structure of the variable neighborhood search is shown in Figure 1 [7].

All the neighborhood structures and the Ant Colony algorithm for UPMP were developed using Visual C, 2008 with Windows Vista Home Premium.

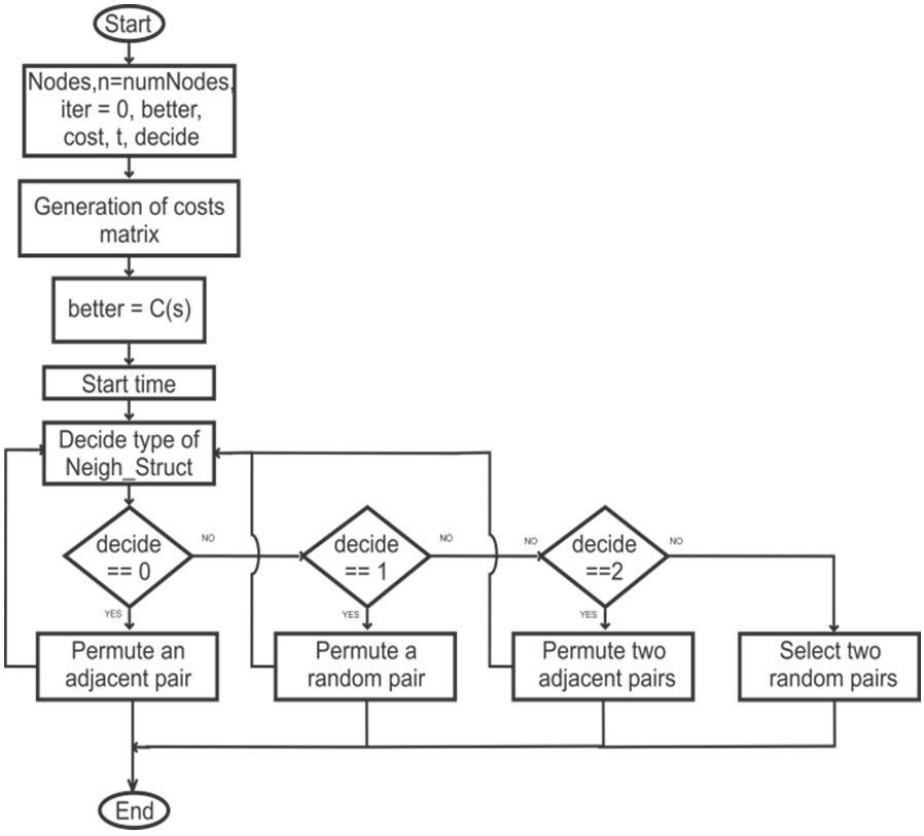

Fig. 1. Variable Neighborhood Structure for the UPMP and the TSP.

6 Experimental Results

Tests were performed on a laptop with the following characteristics: Centrino Core 2Duo processor 2.0 GHz, 3GB RAM memory, Windows Vista Home Premium.

Neighborhood structures and the Ant Colony algorithm were developed using a Visual C 2008 compiler. Test problems were randomly generated for 200, 500, 1000, 2000, 4000, 5000, 6000, 7000, 8000, and 9000 cities for the TSP, respectively. In the case of UPMP, test problems were randomly generated for 200, 250, 270, and 300 jobs to be scheduled on 12 machines, respectively.

This research tested the hypothesis: "*The improvement in solution quality depends not only on the applied variable neighborhood structure, but also on the problem complexity*". This research undertook two problems. Although CSTSP and UPMP are both NP problems, their hardness is different. CSTSP has a more challenging solution space than UPMP [1].

6.1 Experimental Results for CSTSP

All the problems were tested in all the neighborhood structures, including the variable neighborhood one. Each neighborhood structure carried out 30 executions per test problem.

It is noteworthy that problems were randomly generated because existing benchmarks are smaller than the instances proposed in this research. Test problems used for this purpose consist of a symmetric matrix of $n * n$, where n is the number of cities that have to be visited in a tour. Along with the execution, only the best of the elitist solutions were selected, using time as stop criteria. All the test problems were evaluated for 5 minutes in each of the neighborhood structures, obtaining the best found solution, and the total iterations performed during the specified time.

This procedure allows for good performance and direct comparison among the results obtained for the different implemented neighborhood structures, enabling a reliable efficacy and efficiency analysis.

6.2 Efficacy Testing

To understand the algorithm behavior, an efficacy analysis was conducted. Experimental tests were performed using 10 test problems randomly generated for the CSTSP (200, 500, 1000, 2000, 4000, 5000, 6000, 7000, 8000, and 9000 cities). The problems were evaluated using the neighborhood structures explained in Section 5. Each neighborhood structure conducted 30 executions of 5 minutes each per test problem. Averages of obtained results are shown in Table 1.

According to calculations presented in Table 1, the most effective neighborhood structure in almost all cases is the variable neighborhood structure. The test problem of 500 cities is an exception; the most effective structure was that of two random pairs, although the difference between the best obtained averages of the two structures was minimal. This behavior is caused by one of the main features of the use of heuristics, which is that the results are not constant and do not guarantee optimality.

According to the results presented in Table 1, there is a clear improvement when implementing the variable neighborhood structure for CSTSP versus using the single straightforward neighborhood structures. The improvement was obtained not only for the best found solution, but for the worst one also, independent of the problem size. These results are consistent with those reported in literature.

Table 1. Average of the results obtained in experimental testing for different sized problems.

Prob_ Size	Adjacent Pair	Random Pair	Two Adjacent Pairs	Two Random Pairs	variable neighborhood
200	2012	1994	1974	1945	**1944**
500	5161	5118	5124	**5057**	5098
1000	49661	49375	49202	49166	**49128**
2000	101075	100617	100467	100856	**100232**
4000	198292	198092	197373	197524	**197280**
5000	52345	51885	51762	51831	**51742**
6000	61216	61173	61178	61228	**61168**
7000	72036	72100	72030	72078	**72014**
8000	82539	82617	82284	82482	**82238**
9000	973283	973009	972938	972947	**972930**

6.3 Efficiency Testing

The efficiency is a measure of the time required to find a high quality solution. In this research, efficiency was calculated for 1000 solutions found by each neighborhood structure, and the different sizes of test problems (200, 500, 1000, 2000, 4000, 5000, 6000 cities, respectively) that were studied. Results are shown graphically in Figure 2.

Figure 2 shows that the behavior of all the evaluated neighborhood structures is very similar for small instances (from 200 to 2000 cities). Starting at 4000 cities, some neighborhood structures begin to require more computational effort to find solutions.

Naturally, the interesting cases are the results obtained for large problems (from 4000 to 6000 cities). In these cases, there is a clear difference in efficiency under the same conditions, when only the size of the test problem varies. According to the calculations plotted in Figure 8, the behavior of the variable neighborhood structure is constant for large problems. This demonstrates the variable neighborhood structure's efficiency as competitive, because it shows better efficiency than the neighborhood structure of two adjacent pairs and two random pairs in most of cases.

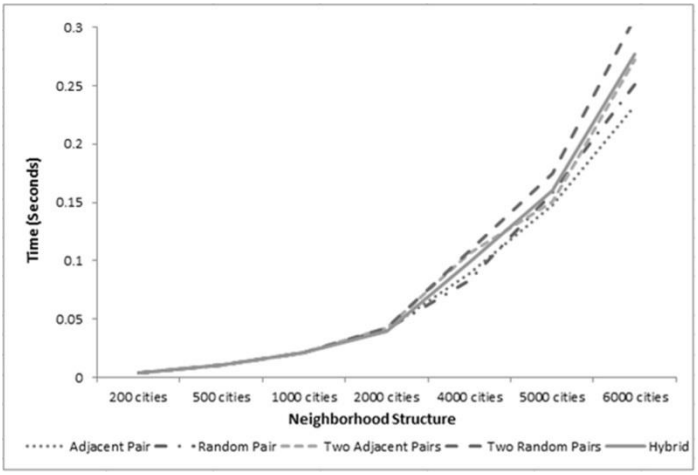

Fig. 2. Time required for five different neighborhood structures to find 1000 solutions for test problems of different sizes (from 200 – 6000 cities).

6.4 Experimental Results for UPMP

Experimental tests for UPMP were conducted in the same way for the CSTSP. Therefore, all the test problems were tested, and the Ant Colony algorithm was implemented in all of the neighborhood structures, including the variable neighborhood one. Each neighborhood structure carried out 30 executions per test problem.

Problems were randomly generated. Test problems used for this problem consist of a matrix of $n * mn$, where n is the number of jobs that have to be scheduled in m machines with k positions each one ($k = n$). Along with the execution, only the best of the elitist solutions were selected, using time as stop criteria. The Ant Colony algorithm was executed for 3 hours, obtaining the best found solution, and the total number of iterations performed during the specified time.

This procedure allows for good performance and direct comparison among the results obtained for the different implemented neighborhood structures, enabling a reliable efficacy and efficiency analysis.

6.5 Efficacy Testing

Experimental tests were performed using 4 test problems randomly generated for the UPMP of 200, 250, 270, and 300 jobs, respectively, which have to be scheduled on k positions (number of jobs) of 12 machines. The problems were evaluated using the neighborhood structures explained in Section 5, which were implemented into an Ant Colony algorithm, in order to get an improvement in the quality of solution. The enhancement procedure was applied to the neighborhood of the best solution so far. The Ant Colony algorithm conducted 30 executions per test problem on each neighborhood structure. The executing time was almost 3 hours, depending on the input size. Averages of obtained results are shown in Tables 2.

According to calculations presented in Table 2, and contrary to results obtained for CSTSP using the same neighborhood structures, the best solution was found in all cases by the random pair neighborhood structure, leaving the variable structure in second place.

Table 2. Average results obtained in experimental testing for different sized problems.

Problem Size Jobs * Machine	Adjacent Pair	Random Pair	Two Adjacent Pairs	Two Random Pairs	variable neighborhood
200*12	2648	**2317**	2768	2678	2320
250*12	3928	**3460**	3465	3998	3515
270*12	4908	**4279**	4389	4986	4582
300*12	5948	**5139**	5232	5975	5388

This research demonstrates that although CSTSP and UPMP are both NP problems, their hardness is different. CSTSP has a harder solution space than UPMP [1]. Therefore, the variable neighborhood structure has more benefits for hard problems, as opposed to less hard problems where it is easier to find good solutions.

6.6 Efficiency Testing

According to the calculations plotted in Figure 3, the behavior of the variable neighborhood structure's efficiency is competitive, because it shows better efficiency than the neighborhood structure of two adjacent pairs and two random pairs in most of cases.

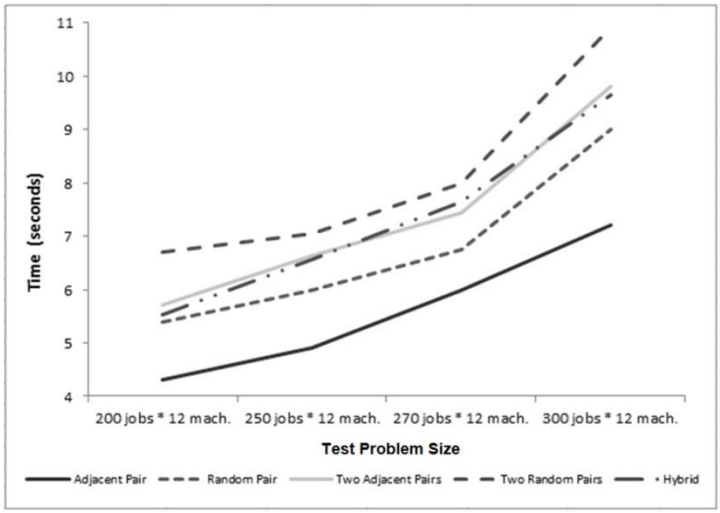

Fig. 3. Time required for each neighborhood structure to find a solution.

7 Conclusions

Many experimental tests were performed for two combinatorial problems with different hardness in their solution space. The problems studied were CSTSP and UPMP. In both cases, test problems were randomly generated, according to the problems' features. Experimental results showed a difference in the variable neighborhood structure performance, because in the case of CSTSP, excellent quality solutions were obtained by the variable neighborhood structure. For UPMP, the same structure did not obtain the best solutions.

According to experimental analysis, the hypothesis presented in this paper is confirmed. The contribution of this research is the experimental proof that not all the variable neighborhood structures work well in all discrete optimization problems when compared to other straightforward structures. This is seen clearly in the case of UPMP, where a straightforward structure gets better quality solutions than the variable neighborhood one.

This research demonstrates that although CSTSP and UPMP are both NP problems, CSTSP has a harder solution space than UPMP. The variable neighborhood structure has more benefits for harder problems, as opposed to less hard problems where it is easier to find good solutions.

8 References

1. Papadimitriou C.H. & Steiglitz K., Combinatorial Optimization, Algorithms and Complexity. Inc. Mineola, New York. USA. Dover Publications, (1998).
2. Arajy Yahya Z. & Abdullah S., Hybrid Variable Neighborhood Algorithm for Attribute Reduction in Rough Set Theory. Cairo, Egypt. 10th. International Conference on Intelligent Systems Design and Applications, ISDA. IEEE, (2010).
3. Hansen P. & Mladenović N., Variable Neighborhood Search: Principles and Applications. pp. 449 – 467. European Journal of Operational Research, (1999).
4. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A. & Protasi, M., Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties. Springer-Verlag, (1999).
5. Liu S. B., Ng K. M. & Ong. H. L., A New Heuristic Algorithm for the Classical Symmetric Traveling Salesman Problem. pp. 267-271. World Academy of Science, Engineering and Technology, (2007).
6. Fischetti M., Salazar-González J. J. & Toth P., The Symmetric Generalized Traveling Salesman Polytope. CCC 0028-3045/95/020113-11. pp. 113-123. Vol. 26. Issue. 2. Journal NETWORKS, (1995).
7. Cruz-Chávez M. A., Martínez-Oropeza A. & Serna-Barquera S. A., Neighborhood Hybrid Structure for Discrete Optimization Problems. ISBN-13: 978-0-7695-4204-1. pp. 108 – 113. Proceedings IEEE Electronics, Robotics and Automotive Mechanics Conference. CERMA, (2010).
8. Garey M. R., Johnson, D.S. & Shethi R., The Complexity of Flow Shop and Job Shop Scheduling. Vol. 1, No.2. pp. 117-129. Mathematics of Operation Research, (1976).

9. Lin S. & Kernighan W., An Effective Heuristic for the Traveling Salesman Problem. Vol. 21, No. 2. DOI: 10.1287/opre.21.2.498. Operations Research, (1973).
10. González-Velázquez R. & Bandala-Garcés M. A., Hybrid Algorithm: Scaling Hill and Simulated Annealing to Solve the Quadratic Allowance Problem. Guerrero, México. 3th. Latin-Iberoamerican Workshop of Operation Research, (2009).
11. Pacheco J. & Delgado, C., Different Experiences Results with Local Search Applied to Path Problem. pp. 54- 81. Vol. 2, No. 1. ISSN: 1575-605X. Electronic Journal of Electronics of Comunications and Works ASEPUMA, (2000).
12. Kenneth D. B., Cost Versus Distance in the Traveling Salesman Problem. Dept. Los Angeles. CA 90024 1596. Citeseer. USA. UCLA Computer Science, (1995).
13. Lourenço H. R. & Martin O. C., Iterated Local Search. Vol. 57. pp. 320 -353. Handbook of Metaheustics. International Series in Operations Research & Management Science. SpringerLink, (2003).
14. Martin O., Otto S. W. & Felten E. W., Large Step Markov Chains for the Traveling Salesman Problem. pp 299-326. Complex Systems, (1991).
15. Martin O., Otto S. W. & Felten E. W., Large Step Markov Chains for the TSP Incorporating Local Search Heuristics. pp 219-224. Operations Reasearch, (1992).
16. Pinedo M. L., Scheduling Theory, Algorithms, and Systems. Third Edition. New York University. ISBN: 978-0-387-78934-7, e-ISBN: 978-0-387-78935-4. Ed. Prentice Hall, (2008).