JUREMA, A NEW BRANCH & BOUND ANYTIME ALGORITHM FOR THE ASYMMETRIC TRAVELLING SALESMAN PROBLEM

Tiago Carneiro Pessoa
Universidade Estadual do Ceará
Mestrado Acadêmico em Ciências da Computação – MACC
Av. Paranjana, 1700 – Serrinha, CEP: 60740-000
E-mail: carneiro@larces.uece.com

Marcos José Negreiros Gomes
Universidade Estadual do Ceará
Mestrado Profissional em Computação Aplicada – MPCOMP/UECE-IFCE
Av. Paranjana, 1700 – Serrinha, CEP: 60740-000
E-mail: negreiro@graphvs.com.br

ABSTRACT

In many different ways to solve combinatorial optimization problems, the Branch-and-Bound (BnB) method is one that performs implicit and complete enumeration of the solution space, and is one of the most popular methods because of its inherent optimality proof when in its termination. We show a new way to do Branch-and-Bound, evaluated to the Asymmetrical Traveling Salesman Problem (ATSP), called JUREMA. Jurema is an abundant tree from the Brazilian semi-arid, which topology in dry seasons looks like the BnB tree of the related method. The method Jurema showed to be of high performance for the very difficult ATSP instances, comparatively superior to traditional BnB-DFS commonly used for this propose. The method can be extended for other NP-Hard problems, once it can be combined with heuristic and metaheuristics with great success. Jurema is an Anytime BnB. We present the details of the method and the results for the difficult instances form the literature.

KEYWORDS. ATSP. Branch & Bound. Anytime algorithms. Main area: Combinatorial Optimization.

1. Introduction

The Travelling Salesman Problem (TSP), is the most well known and studied Combinatorial Optimization problem (GUTTIN; PRUNNER, 2002; ZHANG, 2004; HELSGAUN, 2006; LAPORTE, 2006). Due to its fame and hardness, the TSP plays a special role in testing and development of new optimization techniques (VAUDORIS; TSANG, 1999).

The TSP is symmetric if the cost matrix is symmetric ($c_{ij}=c_{ji}$, $\forall i,j$), asymmetric otherwise ($\exists\ c_{ij}\neq c_{ji}$, $\forall i,j$). The asymmetric case is a more general way of representation and its instances are frequently harder to solve than the symmetric case instances (ZHANG; KORF, 1996; ZHANG, 2004).

Due to the Combinatorial Optimization problems' relevance, a large number of algorithms were created to solve them, algorithms that can be divided into complete and approximate (BLUM; ROLI, 2003).

Complete algorithms are those that guarantee to find an optimal solution for a valid instance of the problem, in a certain period of time. Among the complete algorithms, a method called Branch-and-Bound (BnB) proves the optimality of solutions of any combinatorial problem, since the instance is properly adjusted. The BnB consists in a set of algorithms that have common characteristics and it is the most widely used method to solve hard combinatorial problems (CRAINIC; ROUCAIROL, 2006).

According to Zhang (1993; 2000), BnB algorithms evaluates the search space in a gradual way and as the time passes, the algorithm can find better solutions till the optimality is proved. Thus BnB algorithms can be easily adapted as Anytime Algorithms, i.e. an algorithm that anytime during its execution can provide a solution to an instance of the problem (GRASS, 1996).

Generally, as a search strategy, Branch-and-Bound methods uses Depth-First Search techniques, where the most recently generated sub problem is explored first or Best-First Search (Breadth-First plus evaluation (Van LE, 1993)), where the most promising sub problem is explored first (ZHANG, 1996; PAPADIMITRIOU; STEIGLITZ, 1998).

It is presented here a novel way of doing BnB that uses the best of the most widely used search strategies. To this composition of strategies, it was given the name JUREMA, since the Jurema Search's resulting tree looks like an important and abundant tree found in the Brazilian semi-arid region, called Jurema. In this methodology, it is observed that new solutions are more often found then the traditional BnB-DFS, in most cases evaluated.

The remainder of this paper is structured as follows: in section 2 it is presented the Jurema Method, in section 3 is presented the computational evaluation and the results of the comparison between DFS and the Jurema Method; in the conclusions it's considered the results achieved and future work.

2. The Jurema Method

The Jurema Method has three main characteristics:

1. Is a hybrid method and starts the search from a known solution, called Guiding Solution;
2. Is a different search algorithm. Its search is not performed from root to leaf, like traditional Branch-and-Bound algorithms. The Jurema Search is performed from leaf to root;
3. As a search strategy, the Jurema method applies a Depth-First Search and Breadth-First Search combination.

2.1 Origin of the chosen name

The Jurema Method preempts that the Guiding Solution is a good solution (Upper Bound), near the global optimum. The Guiding Solution can be quickly obtained using traditional heuristics or metaheuristcs. From this starting solution, new upper bounds (solutions) can be quickly found by searching the starting solution's neighborhood, at levels closer to the leaf.

Once quickly found, these new upper bounds would promote a premature pruning of subtrees present in levels closer to the Branch-and-Bound tree's root, resulting in a tree with scattered branches and few leafs.

When a sketch of what a Jurema Method's resulting Branch-and-Bound tree would looks like was drawn, was noticed that this resulting tree really looked like with the Brazilian's semi-arid region trees. These trees, called *xerofitas* or *xerofilas*, have a twisted and covered by thorns trunk, scatter branches and, during the dry season, as a way to save water, they loses its leafs.

The Jurema tree, *Mimosa tenuiflora* (BAKKE *et. al*, 2006), is *xerofita* whom belongs to the acacia's family and it is one of the most abundant species of Brazilian semi-arid's flora, Figure 2-1.

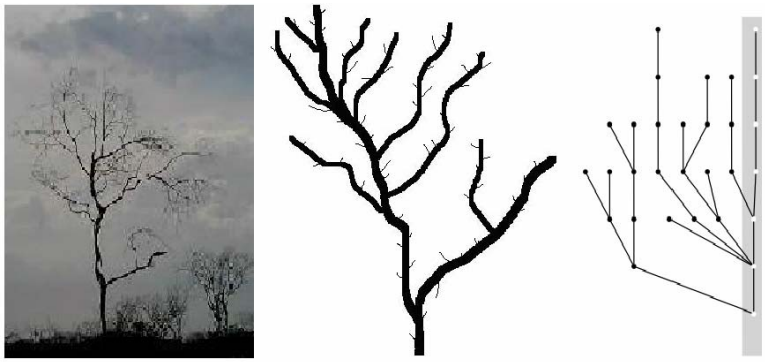


Figure 2-1: Three "Juremas": Black Jurema (real one), a sketch of what a Jurema Method's resulting Branch-and-Bound tree would looks like and, finally, a Jurema Method's Branch-and-Bound tree.

The Jurema is, for some cultures, a holy tree (Figure 2-1). For some indigenous cultures, the Black Jurema is the "Tree of Life" (KANABOGY, 2009; VOLPATTO, 2009). In Figure 2-1, it is shown, from left to right, the Black Jurema found in Brazilian's semi-arid region, a sketch of what a Jurema Method's resulting Branch-and-Bound tree would looks like and finally, a Jurema Method's Branch-and-Bound tree.

2.2 Jurema Method

The Jurema Method is divided in three major steps (Algorithm 2-1): upper bound calculation; lower bound calculation and Jurema Search (Algorithm 2-2).

To initial upper bound calculation and Guiding Solution construction, any heuristic or metaheuristic for the ATSP can be used. The only requirement is to store the Hamilton Cycle in a vector of size n+1 (in Algorithm 2-2 this vector is called *guidingSolution*). To construct the *Guiding Solution* and upper bound calculation, it was applied a combination of Farthest insertion and the 2-Opt (Croes, 1954) well known heuristic. A powerful heuristic like Helsgaun (2006) could also be used, but once Helsgaun's usually finds the optimal solution or an upper bound really close to the optimum, the initial purpose of Jurema Method: to find new solutions quickly would not be observed since it would not be possible to observe how often new upper bounds would be found.

In step two, any method to calculate a lower bound to the ATSP can be applied. Was used, in the current Jurema Method implementation, the lower bound calculation present in (LITTLE; MURTY; SWEENEY; KAROL, 1963), based on the Hungarian Method for solving the Assignment Problem, where a lower bound for the ATSP is the sum of the greatest constant

present in each row and column of the original cost matrix C_{nxn}.

The third major step, Jurema Search, is described further.

2.2.1 Formal Description

After the initial upper bound (UB) calculation, the search step (Jurema Search) can be started. A pseudocode for the third step (Jurema Search) can be seen bellow (Algorithm 2-2).

As the search begins at the leaf node (node *n+1*), it is necessary, before starting the search's most external loop, to subtract, from the initial upper bound total cost, called *cost*, the edges *guidingSolution[n]* → *guidingSolution[n+1]* and *guidingSolution[n-1]* → *guidingSolution[n]* weights, since the search step cannot follow another path from nodes *guidingSolution[n-1]*, *guidingSolution[n]* and *guidingSolution[n+1]* other than what has been taken in the construction of the Guiding Solution. These two initial steps are shown in Figure 2-2.

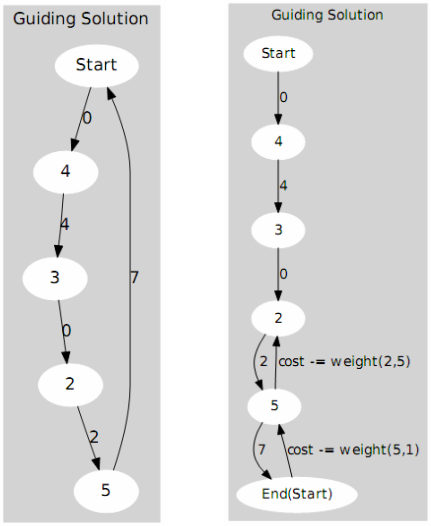


Figure 2-2: Illustrations that represents the initial Guiding Solution (left) and the two necessaries steps before the Jurema Search's most external loop (right).

In algorithm 2-2, the BFS stage's objective is to find, reachable from *actualNode*, all non visited nodes belonging to level *actualLevel + 1*. These nodes are enqueued and this resulting queue is ordered using the lower bound of each enqueued node as a parameter. The objective of this ordination is to explore the most promising regions of the search space first (Jurema Search's Best-First Search element) and avoid DFS to get stuck in branches that could not lead to the optimal solution.

```
Procedure JUREMA_METHOD begin
    upperBoundCalculation(); {UB calculation and Guiding Solution obtaining}
    lowerBoundCalculation(); {LB calculation}
    JUREMA_SEARCH();         {Search step }

end

Procedure DFS_BNB() begin
    upperBoundCalculation(); {UB calculation}
    lowerBoundCalculation(); {LB calculation}
    DFS(Start);
end
```

Algorithm 2-1: Pseudocodes of Jurema Method (up) and DFS-BnB (down). Upper and lower bounds were the same for Jurema and DFS-BnB.

```
Procedure JUREMA_SEARCH Begin
    cost = UB;
    cost = cost - weight(guidingSolutin[n], guidingSolution[n+1]);
    cost = cost - weight(guidingSolutin[n-1], guidingSolution[n]);
    for level:= (n-2) downto 1 do begin
        cost = cost - weight(guidingSolution[level],guidingSolution[next(level)]);
        actualNode = guidingSolution[level];
        generate all actualNode's next level children(CHi) and their respective
        lower bounds(LBi); {BFS stage}
            if(LBi > UB) then kill CHi;
            else
                queue.push(CHi);
        queue.sort(); {Using lower bounds as a sorting parameter, BeFS component}
        activeSet.push(guidingSolution[next(level)]);
        while(!queue.empty()) begin
            branchingNode = queue.pop();
            branch(branchingNode); {DFS stage}
        end
    end
end
```

Algorithm 2-2: Algorithm that represents the Jurema Search.

As the search is performed in a leaf-root way, the *actualNode's* children *guidingSolution[actualLevel+1]* is only inserted into the active set after the BFS step, so it's not discovered and branched again unnecessarily.

Each node discovered through BFS will be a Depth-First Search (DFS) root (Figure 2-3). After this set of steps mentioned, the Jurema Search still is into the most external loop, but now, the search is performing the second iteration.

In Figure 2-3 are shown details of the most external loop's first and second iterations. It's important to underline that nodes belonging to the *Guiding Solution* vector were not discovered by the Jurema Search (gray rectangle nodes), they were discovered by the upper bound calculation and they were used only to guide the searching step. Nodes discovered by the Jurema Search have (*x*)_*y* labels, where *x* represents the order which the node was discovered and *y* represents the node's index (the city number to be visited by the travelling salesman).

A set of illustrations representing the first (left) and second (right) most internal loop's iterations are shown in Figure 2-3. Figure 2-4 shows the Jurema Search's Branch-and-Bound resulting tree.

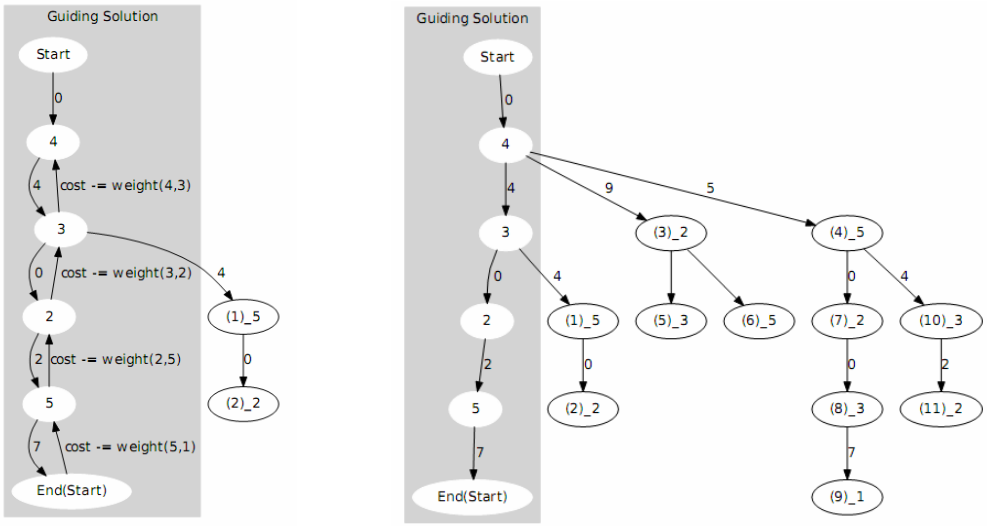


Figure 2-3: Set of illustrations representing the first iteration (left) and the second iteration (right) of the Jurema Search's most external loop.

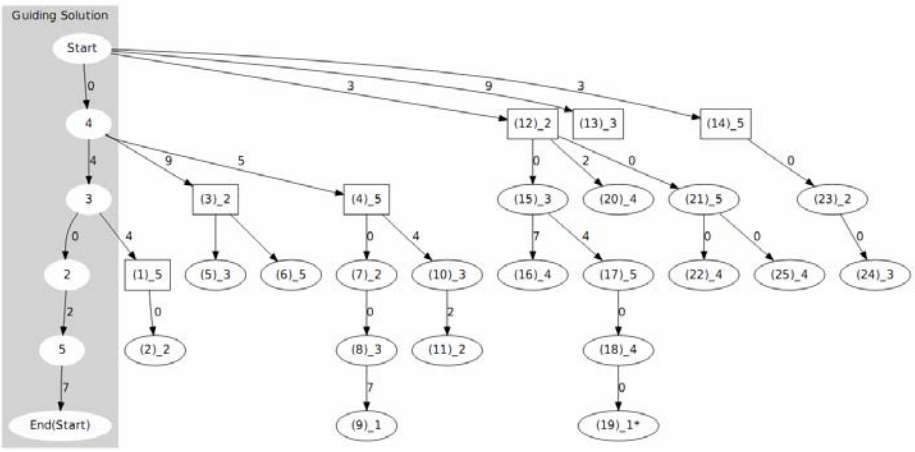


Figure 2-4: Jurema Search's Branch-and-Bound resulting tree[1].

3. Computational evaluation

During the preliminary research, a C++ application was conceived. This application contained an implementation of a Depth-First Search Branch-and-Bound (DFS-BnB) algorithm and an implementation of Jurema Method.

To generate the resulting Branch-and-Bound tree for the two methods, the application writes a metadata. This metadata is interpreted by the software *Graphvz 2.4*, which generates the visualization of the resulting BnB tree.

The conceived application is able to calculate the elapsed time necessary to the Jurema Method and to DFS-BnB perform its searches.

3.1 Test Methodology

The testing step was divided into two stages: in the first stage both algorithms (Jurema and DFS-BnB) performed searches over random asymmetric instances. In the second stage, both algorithms performed searches over asymmetric instances belonging to the TSP-LIB[2] and they were executed as Anytime Algorithms (ZHANG 1993; ZHANG, 2000) during five hours.

In the first stage, each search strategy (Jurema and DFS) received an ATSP instance. These instances had four to forty cities and they were generated by the pseudorandom numbers generator *rand( )*, provided by *GNU C Library (glibc)*[3]. These instances are C_{nxn} and also represent complete graphs where each element c_{ij} can vary from zero to one thousand, in a closed interval, when i and j are different values. When i and j are the same value, c_{ij} receives the value equal to infinity.

The first stage was divided in two steps. In the first step, the lower bound was not considered and only the instances of four to twenty-five cities were utilized, because DFS-BnB cost-based only took a long time to solve instances greater than twenty-five cities. In the second step, the lower bound for the ATSP presented in (LITTLE et al., 1963), based on the Hungarian Method for the Assignment Problem, was applied and all the random asymmetric instances of four to forty cities were considered.

To perform the comparison between Jurema and DFS performances, the tree size and the elapsed time, in minutes, to perform a complete search, were used as a parameter.

[1] Instance of five cities created for didactic purpose and solved by Jurema not using LB (only based on cost), so a lot of branches can be seen even in this small instance.
[2] Available at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp
[3] The GNU C Library Project: http://www.gnu.org/software/libc/

The well known TSP-LIB asymmetric instances were utilized in the second stage. These instances have seventeen to four hundred forty-three cities and they represent complete weighted asymmetrical graphs.

Data collected in the second stage were: elapsed time to find the first solution, elapsed time to find the last solution, elapsed time for searching the entire solution space and the number of solutions found.

3.1.1 Testing environment

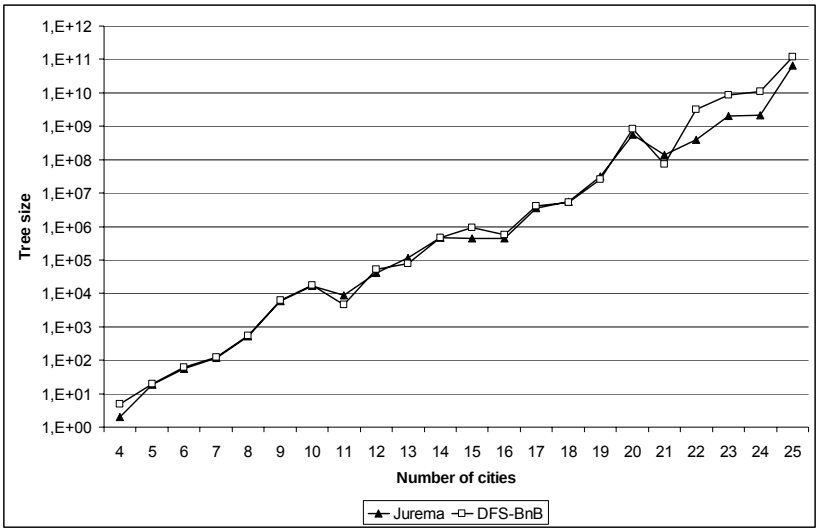
The experiments were done on an Intel Core 2 Duo E6600 @ 2.4GHz, 2 GB DDR2 800 MHz RAM workstation.

The operative system was Arch Linux (kernel 2.6.27) and codes were compiled with Gnu Compiler Collection 4.3.4 using flag –O2. CPU times were measured by *time( )* function, provided by *glibc*.

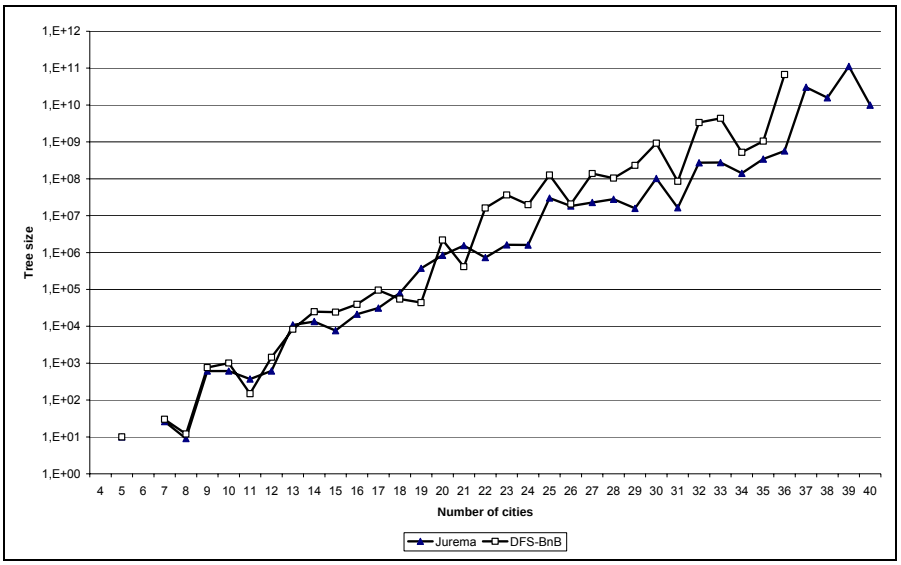
3.1.2 Results

3.1.2.1 Results of stage one

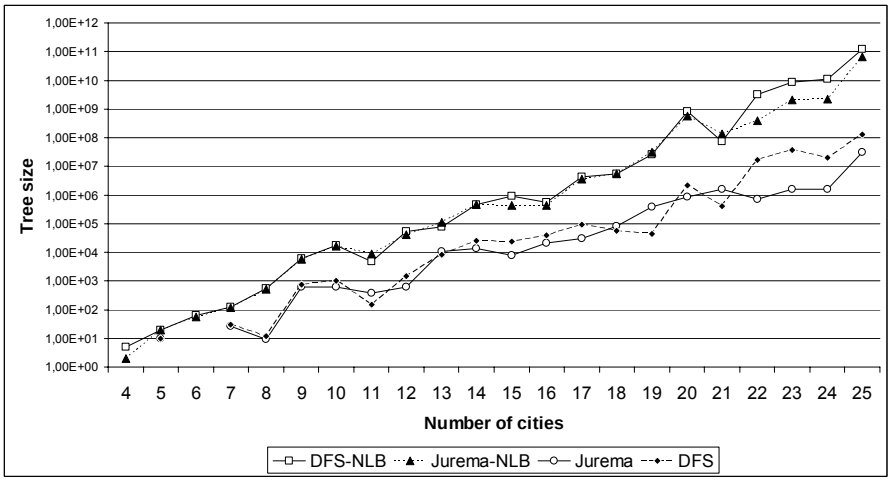
In Graphic 3-1, where the size of the Branch-and-Bound tree is represented on a logarithmic scale, can be concluded that in the first step, where lower bounds were not considered (cost-only based BnB) and only instances of four to twenty-five cities were used, the Jurema Method is slightly better than the DFS-BnB. Otherwise, when the lower bound were considered and all the random asymmetric instances were utilized, the Jurema Method's performance was much better than the traditional DFS-BnB (Graphic 3-2). In Graphic 3-2 the *Y* axis point for instance of sizes four and six (both methods) and size five (Jurema method) were not plotted because there were no branches. The DFS-BnB points for instances of size 37 to 40, in Graphic 3-2, were not plotted because DFS-BnB took a really huge amount of time, if compared to the time required by the Jurema to proof the optimality, without finding any solution.



Graphic 3-1 (step one) – This Graphic presents, on logarithmical notation, a comparison between Jurema and DFS-BnB, both of them are not using lower bound, for random asymmetric instances of four to twenty-five cities.



Graphic 3-2 (step 2): This Graphic presents, on logarithmical notation, a comparison between Jurema and DFS-BnB, both for random asymmetric instances of four to forty cities.



Graphic 3-3: This Graphic presents, on logarithmical notation, a comparison between the tests using lower bound, and not using lower bound (NLB), for random asymmetric instances of four to twenty-five cities.

3.1.2.2 Results of stage two

Details of the instances used in this stage of tests are shown bellow (Table 3-1).

Instance name	Number of cities	Initial LB	LB Gap(%)	Initial UB	UB Gap(%)	Optimal Solution
br17	17	0	-X-	41	5.13%	39
ftv33	34	1099	-14.54%	1563	21.54%	1286
ftv35	36	1248	-15.27%	1662	12.83%	1473
ftv38	39	1321	-13.66%	1719	12.35%	1530
ftv44	45	1392	-13.70%	1946	20.64%	1613
kro124p	100	32649	-9.88%	42161	16.63%	36230
rbg323	323	630	-52.49%	1866	40.72%	1326
rbg358	358	358	-69.22%	1775	52.62%	1163
rbg403	403	304	-87.67%	2848	15.54%	2465
rbg443	443	384	-85.88%	3179	16.88%	2720
p43	43	141	-97.49%	5631	0.18%	5620
Average gap:			-50.89%		20.5%	

Table 3-1: Details of instances used in this stage of tests solved using FITSP + 2Opt. Lower bound presented in (LITTLE et *al., 1963)* were considered.

Table 3-2 and 3-3 shows the results of searches performed by Jurema Method and DFS-BnB through instances presented in Table 3-1, during the period of three hundred minutes (5h).

DFS-BnB						
Name	**Time of first solution found (min)** [4]	**Time of last solution found (min)**	**Cost of last solution**	**Gap(%)**	**Amount of solutions found**	**Optimality proved?**
br17	0.017	0.217	39	0.00%	2	Yes
ftv33	0.983(4)	135.800	1286	0.00%	36	Yes
ftv35	0.567	253.850	1489	1.00%	31	No
ftv38	24.4	84.75	1644	7.45%	16	No
ftv44	24.033	27.833(2)	1894	17.42%	5	No
kro124p	-	-	-	13.63%	0	No
rbg323	-	-	-	40.72%	0	No
rbg358	-	-	-	52.62%	0	No
rbg403	-	-	-	15.54%	0	No
rbg443	-	-	-	16.51%	0	No
p43	-	-	-	0.18%	0	No
***Average gap*: 15.01%**						

Table 3-2: Results of searches performed by DFS-BnB through the TSP-LIB instances.

[4] "0.000(11)" means that eleven solutions were found at 0.000.

JUREMA						
Name	**Time of first solution found (min)**	**Time of last solution found (min)**	**Cost of last solution.**	**Gap(%)**	**Amount of solutions found**	**Optimality proved?**
br17	0.000	0.000	39	0.00%	1	Yes
ftv33	0.000(11)	276.117	1395	8.00%	16	No
ftv35	0.000(4)	208.967	1473	0.00%	26	Yes
ftv38	0.000(6)	146.15	1536	0.30%	19	No
ftv44	0.000(3)	67.517(2)	1862	15.43%	10	No
kro124p	0.000	258.63	40976	13.00%	34	No
rbg323	0.000	0.13	1862	40.42%	2	No
rbg358	0.033	0.033	1769	52.11%	1	No
rbg403	13.683(2)	14.150(2)	2839	15.17%	4	No
rbg443	0.000	0.500	3168	16.47%	3	No
p43	0.000	0.000	5629	0.14%	1	No
***Average gap*: 14.64%**						

Table 3-3: Results of searches performed by Jurema through the TSP-LIB instances.

3.2 Analysis of the results

A first consideration to be made is about the quality of the bounds used. For the random instances (Step One), the strategy to calculate the initial upper bound had a poor performance, finding, for some instances, solution two times bigger than the optimal solution (Graphic 3-1, instances 15, 21, 23 and 24, for example). It is not good for the Jurema Method, since Jurema's performance is affected by the quality of the Guiding Solution. Not the numeric quality, but structural quality. If the upper bound value is a number far from the optimal, but the first vertexes of the optimal Hamilton Cicle are in the Guiding Solution, Jurema Search will not be affected, will find the optimal solution quicky and will terminate its search soon. This is the spirit of the Jurema Method.

For the TSPLIB instances, the strategy applied to LB calculation, had results as poor as the FITSP + 2-Opt presented for the upper bound calculation step. One reason to justify this is that the strategy presented in Little et al. (1963) always removes the great constant present in each row and column of the initial cost matrix. The more columns and rows containing zeros exists in the instance, lower is the lower bound.

Another point to be considered is the following: if the original cost matrix's elements are into a closed interval [0, X], where this X is a very big value, the quality of the lower bound can be affected, once the strategy used in the tests removes only the greatest constant, but this constant can be very small if compared to the others elements of the row or column, compromising the quality of the lower bound.

The good overall results obtained in the first stage of the tests (Graphic 3-3), in the tests that do not consider the lower bound (NLB), show that the Jurema Method's proposed search way, a combination of DFS, BeFS and leaf-root search is a more effective search strategy than the widely used DFS.

All tests of the step number one considering upper and lower bounds are superior to the tests that considerer only upper bound as a way to restrict the search, emphasizing the importance of using upper and lower bounds to make the search more efficient.

In the tests considering the lower bounds (Graphic 3-2), the Jurema Method performed some memorable reductions in comparison to the DFS-BnB. In instances 15, 22, 23 and 24, for example, Jurema Method performed a reduction of almost ninety percent, compared to the size of DFS-BnB tree. It shows that Jurema Method can promote a really premature pruning of subtrees.

On the second stage it can be seen that the Jurema Method's initial proposal, to find new solutions (upper bounds) really quickly, is accomplished. In almost all tested instances the Jurema Method found a burst of new upper bounds in less than one second (see tables 3-2 and 3-3).

After finding a good amount of new solutions quickly, the Jurema Method behaves just like the DFS-BnB, in other words, spend much time without finding new upper bounds (see Table 3-2 and 3-3). As was presented (section 2), the Jurema Search uses this same DFS to branch the nodes discovered by the BFS (algorithm 2-2). Despite being able to find solutions quickly (ZHANG, 2000), the DFS have some issues: to get stuck in branches that do not led to the optimal solution (Van LE, 1993), the most recently discovered node is always explored first, even though it shows signs that its subtree will be pruned soon, and the recursiveness. The great number of recursive calls causes a very large computational effort to handle them.

Since the Jurema Method uses this DFS implementation to branch the nodes discovered by the BFS step, as the search approaches the root (the search is performed from leaf to root), the more the Jurema Search suffers due to the DFS's issues presented before.

4. Conclusions

It was presented in this paper an algorithm to solve the Asymmetric Travelling Salesmen Problem, different from everything that has been proposed in the specialized literature, called Jurema Method.

The proposed Branch-and-Bound algorithm accomplishes its original mission, to find, starting from a solution previously found, new solutions quickly (upper bounds). This property makes the Jurema Method a better anytime algorithm than the DFS-BnB. In spite of finding new solutions quickly, the implementation of Jurema Method is compromised by the DFS implementation used, but this problem can be easily solved.

The Jurema Method, even in tests that did not consider the lower bounds, once its search is a leaf-root based and uses a composition of search strategies, is a more efficient search algorithm than the popular DFS-BnB, because as it approaches the root, since the Jurema Method can find new solutions quickly, few steps down to the leafs to find new solutions are necessary.

Any method of lower and upper bound calculation can be used with Jurema Method, but is important to say that Jurema Method's performance is closely related to the quality of the Guiding Solution. Not the numeric quality, but structural quality. If the solution is not good in its structure, i.e. the optimal Hamilton Cycles' first vertexes are not in the Guiding Solution, the search must walk toward the root and return to the levels closer to the leaf in order to find better solutions, what makes Jurema Search inefficient.

The quality of upper and lower bounds used here are lower than the expected, but they were used since they are easy to implement and the purpose of this paper is to evaluate the search step of the algorithm and the algorithm's anytime capability. Bounds presented in (FISCHETTI; TOTH, 1992; TURKENSTEEN et al., 2007) will be considered further.

The DFS can, in some scenarios, be more effective than Jurema Search. For example, in a scenario where the DFS, in one of its first step down, finds the global optimum, promoting really premature pruning of subtrees and proving optimality quickly or in a scenario where the structure of the Guiding Solution is not good.

It is possible to be more effective in the searching space using the proposed method. A step forward in the future is to parallelize the Jurema Method. In this case, different starting trunk solutions (guiding solutions) would be built in different processors, and then the Jurema search can be placed, communicating the bounds in a master-slave parallel architecture. To make this work, a non recursive BnB using Jurema search may be designed. This method is already scratched and called the Black Jurema. It may be used to solve other ATSP difficult instances still not solved.

Finally Jurema Method can be largely used as a BnB Anytime algorithm to find new better solutions after a metaheuristic or heuristic performs its evaluation over an ATSP instance, even for large scale instances. The gain is that it is still possible to find new bounds in reasonable time, knowing better the distance from the optimum solution (not enclosed in any heuristic), and also explore more efficiently the searching space.

References

Bakke, I. A. et al. (2006), "Regeneração Natural da Jurema Preta em Áreas Sob Pastejo Bovino". *Caatinga* (Mossoró, Brasil), v.19, n.3, pp.228-235.

Blum, C and Roli, A. (2003). "Metaheuristics in Combinatorial Optimization: Overviewand Conceptual Comparison". ACM Computing Surveys, Vol. 35, No. 3, pp. 268–308.

Crainic, G. T; Le Cun, B; Roucairol, C. (2006),"Parallel Branch-and-Bound Algorithms". *Parallel Combinatorial Optimization*. John Wiley & Sons, Inc.

Croes, A. (1954)**,** "A method for solving traveling salesmen problems", *Operations Research* vol 2, pp.392-410.

Fischetti, M; Toth, P. (1992), "An additive bounding procedure for the asymmetric travelling salesman problem". *Mathematical Programming: Series A and B*, v.53 n.2, pp.173-197.

Grass, J. (1996), "Reasoning about computational resource allocation". *Crossroads 3, 1*. pp 16-20.

Gutin, G; Prunner, A. (2002), "The Travelling Salesman Problem and Its Variatons". *Series: Combinatorial Optimization*, Volume 12. Kluwer Academic Publishers. New York, Boston, Dordrecht, London, Moscow.

Helsgaun, K. (2006), "An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP Heuristic". *Datalogiske Skrifter* (Writings on Computer Science), No. 109.

Kanabogy, A. (2009), "Caboclo Tupinambá da Cobra Coral". Disponível em <http://www.kanabogy.com.br/jurema_sagrada_4.html>. Acesso em: 22 de setembro de 2009.

Laporte, G. (2006), "A short history of the traveling salesman problem". Molde University College, Norvège.

Little, J. D. C. et. al. (1963), "An Algorithm for The Traveling Salesman Problem". *Operations Research* Vol. 11, No. 6, pp.972-989.

Papadimitriou, C. H; Steiglitz. (1998), "Combinatorial Optimization. Algorithms and Complexity". *Dover Science*. p. 443-452.

Turkensteen, M. et al. (2007), "Tolerance-based Branch and Bound algorithms for the ATSP. European Journal of Operation Research, volume 189, issue 3. pp.755-788.

Van LE, T., (1993). "Techniques of Prolog Programming". *John Wiley and Sons inc*. New York, NY, USA. pp. 272 – 340.

Volpatto, R. (2009), "A Cerimônia do Jucá". Disponível em <http://www.rosanevolpatto.trd.br/lendacerimoniaajuca.html>. Acesso em: 21 de Setembro de 2009.

Voudoris, C; Tsang, E. (1999), "Guided local search and its application to the traveling salesman problem". *European Journal of Operational Research 113*.

Zhang, W. (1996), "Branch-and-Bound search algorithms and their computational complexity". *Research report* no. ISI-RR- 96-44318, South-Carolina University, USA.

Zhang, W. (2000), "Depth-first branch-and-bound vs. local search: A case study". *Proc. 17-th National Conf. on Artificial Intelligence* (AAAI-2000). pp.930-5. Austin, Texas, July 30-August 3.

Zhang, W. (2004), "Phase transitions and backbones of the asymmetric Traveling Salesman Problem". J. *Artificial Intelligence Research*, 20:471-97.

Zhang, W; Korf, R.E. (1996), "A study of complexity transitions on the asymmetric traveling salesman problem". *Artificial Intelligence*, 81(1-2):223-39.