# A Preprocessing that Combines Heuristic and Surrogate Constraint Analysis to Fix Variables in TSP

María A. Osorio, David Pinto

School of Computer Sciences, Universidad Autónoma de Puebla,
72560 Puebla, México
{aosorio, dpinto}@cs.buap.mx

**Abstract.** A preprocessing procedure that uses a local guided search defined in terms of a neighborhood structure to get a feasible solution (UB) and the Osorio and Glover[18, 20] exploiting of surrogate constraints and constraint pairing is applied to the traveling salesman problem. The surrogate constraint is obtained by weighting the original problem constraints by their associated dual values in the linear relaxation of the problem. The objective function is made a constraint less or equal than a feasible solution (UB). The surrogate constraint is paired with this constraint to obtain a combined equation where negative variables are replaced by complemented variables and the resulting constraint is used to fix variables to zero or one before solving the problem.

## 1 Introduction

The TSP has received great attention from the operations research and computer science communities because is very easy to describe but very hard to solve [2]. The problem can be formulated saying that the traveling salesman must visit every city in his territory exactly once and then return to the starting point. Given the cost of travel between all cities, he should plan his itinerary for a minimum total cost of the entire tour.

Space solution for TSP is the *n*-cities permutation, *n*!. Any simple permutation is a different solution. The optimum is the permutation that correspond to a travel with the minimum cost. The evaluation function is very simple, because we only need to add the cost profit associated with each segment in the itinerary, to obtain the total cost for that itinerary.

The TSP is a relatively old problem. It was already documented in 1759, with a different name, by Euler. The term 'traveling salesman' was first used in 1932 in a German book written by a veteran traveling salesman. The TSP, in the way we know it now, was introduced by the RAND Corporation in 1948. The Corporation's reputation helped to make the TSP a well known and popular problem. The TSP also became popular at that time due to the apparition of linear programming and the attempts to solve combinatorial problems.

In 1979, it was probed that the TSP is NP-hard, a special kind of NP-complete problems (see Garey *et al*, [1]). All NP problems can be reduced polynomialy to them. It means that if one can find a solution in polynomial time to one of them, with a deterministic procedure, it may find it for all NP and then, P=NP. Nobody has been able to find efficient algorithms for NP-complete problems until now, and nobody has demonstrated that such algorithms do not exist.

The TSP can be symmetric or asymmetric. In the symmetric case, departure and return costs are the same and can be represented with an undirected graph. For the asymmetric case, the more common one, the departure and return costs are different and can only be represented by a directed graph. Because the symmetric problem can be seen as a special case of the asymmetric one, this research was directed to the asymmetric case and all references to TSP correspond to the asymmetric case.

The TSP has become a classic problem because it serves to represent a great number of applications in real life, as the coloring sequence in textile industry, the design of insulating material and optic filters, the impression of electronic circuits, the planning of trajectories in robotics and many other examples that can be represented using sequences (see Salkin [21]). Besides, it may represent a big number of combinatorial problems that cannot be solved in polynomial time and are NP hard.

The exponential nature of the time needed to solve this problem in an exact way has originated, during the last decades, the development of heuristic algorithms to approximate its optimal solution (see Gass [2]).

To relate the experience obtained in this research, we structured the present paper in the following way. In section 2, we present the Integer Programming formulation for TSP. In section 3, we describe the Dual Surrogate Constraint, and the Paired Constraint in section 4. In Section 5 we present an example solved with our approach. Section 6, shows experimental results and Section 7, the Conclusion.

## 2 Integer Programming Formulation

As we mentioned before, a traveling salesman must visit *n* cities, each exactly once. The distance between every pair of cities *ij*, denoted by $d_{ij}$ $(i \neq j)$, is known and may depend on the direction traveled (*i.e.*, $d_{ij}$ does not necessarily equal $d_{ji}$). The problem is to find a tour which commences and terminates at the salesman's home city and minimizes the total distance traveled.

Suppose we label the home city as city 0 and as city $n+1$. (Then we may think of the salesman's initial location as city 0 and the desired final location as city $n+1$). Also, introduce the zero-one variables $x_{ij}$ $(i=0,1,\ldots,n, j=1,\ldots,n+1, i \neq j)$, where $x_{ij} = 1$ if the salesman travels from city $i$ to $j$, and $x_{ij} = 0$ otherwise. To guarantee that each city (except city 0) is entered exactly once, we have $\sum_{i=0,n} x_{ij} = 1$ $(j=1,\ldots,n+1, i \neq j)$.

Similarly, to ensure that each city (except city $n+1$) is left exactly once, we have $\sum_{j=1,n+1} x_{ij} = 1$ $(i = 0,\ldots,n, i \neq j)$. These constraints, however, do not eliminate the possibility of subtours or "loops". One way of eliminating the subtour possibility is to add the constraints $\alpha_i - \alpha_j + (n+1) x_{ij} \leq n$ $(i = 0,\ldots,n, j=1,\ldots,n+1, i \neq j)$.

Where $\alpha_i$ is a real number associated with city *i*. To complete the model we should minimize the total distance between the cities. An integer programming formulation of the traveling salesman problem is to find variables $x_{ij}$ and arbitrary real numbers $\alpha_i$ which

Minimize $\sum_{i=0,n} \sum_{j=1,n+1} d_{ij} x_{ij}$

Subject to $\sum_{i=0,n} x_{ij} = 1 \qquad (j=1,\ldots,n+1,\ i \neq j)$

$\sum_{j=1,n+1} x_{ij} = 1 \qquad (i = 0,\ldots,n,\ i \neq j)$

$\alpha_i - \alpha_j + (n+1)\, x_{ij} \leq n \qquad (i = 0,\ldots,n, j=1,\ldots,n+1,\ i \neq j)$

$\alpha_i \geq 0 \qquad (i = 0,\ldots,n+1)$

$x_{ij} \in \{0,1\}, \qquad (i = 0,\ldots,n, j=1,\ldots,n+1,\ i \neq j)$

Where $x_{0,n+1}=0$ (since $x_{ij} = 0$ for $i = j$). This formulation originally appeared in Tucker [22], and avoids subtours successfully, but enlarge considerable the model that now has $(n+1)^2+2$ variables with $(n+1)^2+n$ binaries and $(n+2)+(n+1)^2$ constraints.

## 3 Dual Surrogate

As defined by Glover [4], a surrogate constraint is an inequality implied by the constraints of an integer program and designed to capture useful information that cannot be extracted from the parent constraints individually, but which is nevertheless a consequence of their conjunction.

The integer program can be written as:

Minimize $\boldsymbol{cx}$
subject to $\boldsymbol{Ax} \leq \boldsymbol{b}$
$\boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{e}$
and $\boldsymbol{x}$ integer

Since $\boldsymbol{Ax} \leq \boldsymbol{b}$ implies $\boldsymbol{b} - \boldsymbol{Ax} \geq \boldsymbol{0}$, we have for a nonnegative weighting vector $\boldsymbol{u}$ that $\boldsymbol{u}(\boldsymbol{b} - \boldsymbol{Ax}) \geq \boldsymbol{0}$ is a surrogate constraint. A value of $\boldsymbol{u}$ is selected which satisfies a most useful or a "strongest" surrogate constraint definition as given in Glover[4,5]. It has been shown by Glover [5] that $\boldsymbol{u}$ comprises the optimal values of the variables of the dual linear program of the corresponding relaxed LP and that the weighting vector in a strongest constraint consists of the optimal dual variables of the associated linear program.

Optimality conditions for surrogate duality are the requirements that the surrogate multiplier vector $\boldsymbol{u}$ is nonnegative, $\boldsymbol{x}$ is optimal for the surrogate problem, and $\boldsymbol{x}$ is feasible for the primal problem. 'Strong' optimality conditions add the requirement of complementary slackness. A complete derivation of this theory can be seen in Glover [5]. The methodology proposed here relies on these fundamental results.

## 4 Paired Constraint

The main ideas about constraint pairing in integer programming were exposed by Hammer *et al.* [9]. Based on the objective of getting bounds for most variables, the strategy is to pair constraints in the original problem to produce bounds for some variables.

Based on the results exposed about surrogate constraints, the dual surrogate constraint provides the most useful relaxation of the constraint set, and can be paired with the objective function. If we name $K = (n+1)^2+(n+2)$, the total number of constraints and $L = (n+1)^2+2$ , the total number of variables, the resulting surrogate is:

$$\sum_{k=1,K} u_k(a_{kl} z_k) \leq \sum_{k=1,K} u_k b_k \qquad l = 1,\dots, L \tag{1}$$

Where $u_k$ are the dual values for every surrogate, $a_{kl}$, the coefficient in row $k$ and column $l$, $z_k$ the $k^{th}$ variable (it may be $x_{ij}$ or $\alpha_i$), $b_k$ the $k^{th}$ right hand side. Now, we define

$$s_l = \sum_{k=1,K} u_k(a_{kl} z_k) \qquad l = 1,\dots, L \tag{2}$$

Besides, we made the objective function less or equal than a known feasible integer solution (UB). This integer solution was obtained using a guided local search defined in terms of neighborhood structure, where tour B is a neighbor of tour A and it can be obtained from A by specific type of perturbation or move. It takes infinitesimal CPU times to get a feasible tour with this procedure [14 ].

The paired constraint between the surrogate and the objective function will be,

$$\sum_{l=1,L} (c_l - s_l)\, z_l \leq \text{UB} - \sum_{k=1,K} u_k b_k \tag{3}$$

To be able to use constraint (3) to fix variables in both bounds, all coefficients must be positive or zero. We substitute $y_l=1- z_l$ in the negative coefficients $(c_l-s_l)$ to get positive ones $(c_l-s_l)$' and add the equivalent value in the right hand side. The right hand side of the surrogate is the LP optimal solution (LB), and the right hand side of this paired constraint becomes the difference between the best known solution, the upper bound (UB), and the LP solution, the lower bound (LB). The resultant paired constraint used to fix variables to zero or one, is

$$\sum_{l=1,L,cl-sl\geq 0} (c_l-s_l)\, z_l + \sum_{l=1,L,cl-sl<0} (c_j-s_j)'\, y_l \leq \text{UB} - \text{LB} \tag{4}$$

If coefficients $(c_l-s_l)$ of $z_l$ are greater to the difference (UB-LB), those variables must be zero in the integer solution; if the coefficients $(c_l-s_l)$' of $y_l$ are greater to the same difference, those variables must be one in the integer solution because its complement, $y_l$ must be zero. Variables whose coefficients are smaller than the difference remain in the problem. Because we depend on the gap UB–LB and LB can not be changed because it is the LP continuous relaxed solution of the problem, a better UB given by the best integer solution known, can increase the number of integer variables fixed.

## 5 Example

We illustrate the procedure in the following example. Table 1 shows the distances for a traveling salesman problem with 3 cities.

**Table 1.** Distances for the 3-cities example

| From/To | 1 | 2 | 3 |
|---|---|---|---|
| 1 | $\infty$ | 26 | 82 |
| 2 | 134 | $\infty$ | 117 |
| 3 | 38 | 13 | $\infty$ |

The Integer Programming formulation for this example is:

$$
\begin{array}{lll}
\text{Minimize} & 26\,x_{12} + 82\,x_{13} + 134\,x_{21} + 117\,x_{23} + 38\,x_{31} + 13\,x_{32} & \\
\text{Subject to} & x_{01} + x_{02} + x_{03} + x_{04} = 1 & \\
& x_{12} + x_{13} + x_{14} = 1 & \\
& x_{21} + x_{23} + x_{24} = 1 & \\
& x_{31} + x_{32} + x_{34} = 1 & \\
& x_{01} + x_{21} + x_{31} = 1 & \\
& x_{02} + \mathrm{x}_{12} + x_{32} = 1 & \\
& x_{03} + x_{13} + x_{23} = 1 & \\
& x_{04} + x_{14} + x_{24} + x_{34} = 1 & \\
& 4x_{01} + \alpha_0 - \alpha_1 \le 3 & \\
& 4x_{02} + \alpha_0 - \alpha_2 \le 3 & \\
& 4x_{03} + \alpha_0 - \alpha_3 \le 3 & \\
& 4x_{04} + \alpha_0 - \alpha_4 \le 3 & \\
& 4x_{12} + \alpha_1 - \alpha_2 \le 3 & \\
& 4x_{13} + \alpha_1 - \alpha_3 \le 3 & \\
& 4x_{14} + \alpha_1 - \alpha_4 \le 3 & \\
& 4x_{21} + \alpha_2 - \alpha_1 \le 3 & \\
& 4x_{23} + \alpha_2 - \alpha_3 \le 3 & \\
& 4x_{24} + \alpha_2 - \alpha_4 \le 3 & \\
& 4x_{31} + \alpha_3 - \alpha_1 \le 3 & \\
& 4x_{32} + \alpha_3 - \alpha_2 \le 3 & \\
& 4x_{34} + \alpha_3 - \alpha_4 \le 3 & \\
& \alpha_i \ge 0 & (i = 0,\ldots,4) \\
& x_{ij} \in \{0,1\}, & (i = 0,\ldots,3, j=1,\ldots,4,\ i \ne j)
\end{array}
$$

The relaxed LP problem substitutes $x_{ij} \in \{0,1\}$ by $0 \leq x_{ij} \leq 1$. The LP optimal solution is 64 and the dual values for the constraints (not including the bounds) are: $u_i$={-51,0,0,-13,51,26,51,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}. The surrogate constraint, the paired constraint and the variables fixed can be seen in Table 2.

**Table 2.** Surrogate and Paired Constraints

| | $x_{01}$ | $X_{02}$ | $x_{03}$ | $x_{04}$ | $x_{12}$ | $x_{13}$ | $x_{14}$ | $x_{21}$ | $x_{23}$ | $x_{24}$ | $x_{31}$ | $x_{32}$ | $x_{34}$ | | RHS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $c_l$ | 0 | 0 | 0 | 0 | 26 | 82 | 0 | 134 | 117 | 0 | 38 | 13 | 0 | $\leq$ | 80 | UB |
| $s_l$ | 0 | -25 | 0 | -51 | 26 | 51 | 0 | 51 | 51 | 0 | 38 | 13 | -13 | $\leq$ | 64 | LB |
| $c_l$–$s_l$ | 0 | 25 | 0 | 51 | 0 | 31 | 0 | 83 | 66 | 0 | 0 | 0 | 13 | $\leq$ | 16 | UB-LB |
| $x_{ij}$ | | 0 | | 0 | | 0 | | 0 | 0 | | | | | | *fixed* | |

## 6 Experimental Results

We tested our procedure with 30 instances generated with a random exponential distribution that produces specially hard instances [19]. The average values obtained for every set of five instances with the same number of cities but generated with different seeds, are reported in Table 3. The problems were solved in Pentium III with 1066 MHz and 248 MB in RAM. To obtain the LP solution and to solve the problem to optimality, we utilized ILOG CPLEX 8.0. The feasible solution used as UB was obtained with a guided local search defined in terms of neighborhood structure [16].

### 6.1 Hard Problem Generation for TSP

We developed a generator that produces challenging TSP problems. Following the ideas presented in Osorio and Glover [19], our approach uses independently exponential distributions over a wide range to generate the distances between the cities. This kind of instances takes at least 10 times the number of CPU seconds and 100 times the number of nodes in the searching tree, required for CPLEX to get optimality than the instances generated with a random uniformly distribution.

The problem generator used to create the random instances of TSPs is designed as follows. The distances between the cities, $d_{ij}$, are integer numbers drawn from the exponential distribution $d_{ij}$=1.0–1000 $ln(U(0,1))$.

**Table 3.** Results for Hard Instances

| Number of Cities | Best Known | Fixed Variables | % Fixed Variables | % Rel.Dif between Soln's | Optimal Solution |
|---|---|---|---|---|---|
| 10 | 1696 | 57.8 | 51.88 | 7.17 | 1582 |
| 20 | 2321 | 174.4 | 37.67 | 20.62 | 1924 |
| 30 | 3095 | 205.8 | 20.72 | 58.78 | 1949 |

## 7 Conclusions

Our procedure is a very easy way to fix binary variables to their bounds in TSP instances. It can be seen as an effective preprocessing that reduces the binary number of variables to be fixed in a searching tree. The procedure is simple and utilizes a local guided search defined in terms of neighborhood structure to get a feasible tour and surrogate analysis with results from the solution of the LP relaxed problem. The results obtained shows that a percentage of variables can be fixed in a short amount of time for many different instances.

## References

1. Garey, M. and D. Johnson: Computers and Intractability, W.H. Freeman, San Francisco (1979)
2. Gass, S. (ed.): Encyclopedia of Operations Research and Management Sciences. Kluwer Academic Publishers, New York (1997)
3. Glover, F.: Flows in Arborescences. Management Science, 17 (1971) 568-586
4. Glover, F: Surrogate Constraints. Operations Research 16 (1968) 741-749
5. Glover, F.: Surrogate Constraint Duality in Mathematical Programming. Operations Research 23 (1975) 434-451
6. Glover, F., Sherali, H., Lee, Y.: Generating Cuts from Surrogate Constraint Analysis for Zero-One and Multiple Choice Programming. Computational Optimization and Applications 8 (1997) 151-172
7. Greenberg, H. and W. Pierskalla, W.: Surrogate Mathematical Programs. Operations Research 18 (1970) 924-939
8. Granot, F., Hammer, P. L.: On the use of boolean functions in 0-1 linear programming. Methods of Operations Research (1971) 154-184
9. Hammer, P., Padberg, M. and Peled, U.: Constraint Pairing in Integer Programming, INFOR 13 (1975) 68-81
10. Hooker, J.N.: Logic-based methods for optimization. In: Borning, A. (ed.): Principles and Practice of Constraint Programming. Lecture Notes in Computer Science Vol. 874 Springer-Verlag, Berlin Heidelberg New York (1994) 336-349
11. Hooker, J.N.: A Framework for combining solution methods. Working Paper, Carnegie Mellon University (2003)
12. Hooker, J. N., Osorio, M. A.: Mixed Logical/Linear Programming. Discrete Applied Mathematics 96-97 (1999) 395-442
13. Jeroslow, R. E., and J. K. Lowe: Modeling with integer variables. Mathematical Programming Studies 22 (1984) 167-184
14. Johnson, D.S.: "Local Optimization and the Traveling Salesman Problem", in Proceedings of the 17th International Colloquium on Automata, Languages and Programming, pp. 446-461, Springer, Berlin, 1990.
15. Johnson, D. S., McGeoch, L.A.: The Traveling Salesman Problem: A Case Study in Local Optimization. In: E. H. L. Aarts, E.H.L, Lenstra, J.K. (eds.): Local Search in Combinatorial Optimization. John Wiley and Sons, Ltd., (1997) 215-310
16. Johnson, D.S., Gutin, G., McGeoch, L.A., Yeo, A., Zhang,W., Zverovich, A.: Experimental Analysis of Heuristics for the ATSP. In: G. Gutin, G. and A. Punnen, A. (eds.): The

Traveling Salesman Problem and its Variations. Kluwer Academic Publishers, Dordrecht (2002) 445-487

17. Karwan, M. H., Rardin, R. L. Some relationships between Lagrangean and surrogate duality in integer programming. Mathematical Programming 17 (1979) 230-334
18. Osorio, M.A., Glover, F., Hammer, P.: Cutting and Surrogate Constraint Analysis for Improved Multidimensional Knapsack Solutions. Annals of Operations Research 117(2002), 71-93
19. Osorio, M.A., Glover, F.: Hard Problem Generation for MKP. Proceedings of the XI CLAIO. Concepción, Chile (2002)
20. Osorio, M.A., Glover, F.: Exploiting Surrogate Constraint Analysis for Fixing Variables in both bounds for Multidimensional Knapsack Problems. In: Chávez, E., Favela, J., Mejía, M., Oliart, A. (eds.): Proceedings of the Fourth Mexican International Conference on Computer Science. IEEE Computer Society. New Jersey (2003) 263-267
21. Salkin, M.: Integer Programmaing. Adisson-Wesley Publishing Company. New York (1975)
22. Tucker, A.: On Directed Graphs and Integer Programs. IBM Mathematical Research Projecft Technical Report, Princeton University (1960)