1. Introduction

Let $K_n = (V, E)$ be a complete undirected graph with a set $V$ of $n = |V|$ vertices indexed by $\{1, 2, \ldots, n\}$ and a set $E$ of $m = |E|$ edges. A cycle is a sequence of vertices of $K_n$ containing no repeated elements apart from the very first and the very last vertices on the sequencing. If lengths $\{c_e : e \in E\}$ are associated with the edges of $K_n$, the length of a cycle equals the sum of the lengths for all the edges implied by two contiguous vertices in sequencing. A cycle is Hamiltonian if it includes all the vertices of $K_n$. The symmetric traveling salesman problem (STSP) is to find a Hamiltonian cycle of $K_n$ with the shortest length. A number of linear integer programming formulations for the STSP do exist. Among these, the most celebrated and widely used one is that of Dantzig, Fulkerson and Johnson [6]. In order to present that formulation, denote by $\gamma(S)$ the set of edges of $E$ with both end vertices in $S \subset V$. Accordingly, denote by $\delta(S)$, $S \subset V$, the set of edges with one end point in $S$ and another in $V \setminus S$. Real valued variables $\{x_e : e \in E\}$ are used to control the inclusion (or not) of edges into a Hamiltonian cycle and an associated polyhedral region, $\mathcal{R}$, is defined as

\begin{equation}
\begin{align*}
    x(\delta(\{v\})) &= 2, & v \in V & \quad (a) \\
    x(\delta(S)) &\geq 2, & S \subset V, |S| < n & \quad (b) \\
    0 \leq x_e \leq 1, & e \in E & \quad (c)
\end{align*}
\end{equation}

where for a set of edges $M \subset E$, $x(M) = \sum_{e \in M} x_e$.

The TSP formulation of Dantzig, Fulkerson, and Johnson is given by

\begin{equation}
\min \{c^T x : x \in \mathcal{R} \cap \mathbb{Z}^m\} \tag{1.2}
\end{equation}

with the Held and Karp bound being equal to

\begin{equation}
\min \{c^T x : x \in \mathcal{R}\}, \tag{1.3}
\end{equation}

i.e., the value of the Linear Programming relaxation of (1.2). Polyhedral algorithms for solving (1.3) are normally based on the idea of initially eliminating the Subtour Elimination
Constraints (SECs), (1.1.(b)), from $\mathcal{R}$ and then optimizing over the resulting polyhedral region. Violated SECs are then introduced as cutting planes, until 1.3 is attained.

As proposed in [9], the Held and Karp bound can be computed, alternatively, by a combination of Lagrangian relaxation and subgradient optimization. It can be derived by optimizing over the space of conveniently defined 1-Trees of $K_n$. Assume that one (any) given vertex of $K_n$ is set aside, say vertex $v_1$. In association, consider one (any) spanning tree of the subgraph of $K_n$ induced by the vertices $K_n \setminus \{v_1\}$. A 1-Tree of $G$ is obtained by adding any two edges incident on $v_1$ to that spanning tree. Considering the variables previously defined, let $\mathcal{X}$ be the convex hull of the incidence vectors of all the 1-Trees of $G$ just introduced. In association, define the set $\mathcal{R}_1$ of Hamiltonian cycles of $H_n$ as

$$x(\delta(\{v\}) = 2, \quad v \in V \setminus \{v_1\} \quad (1.4)$$

$$x \in \mathcal{X}. \quad (1.5)$$

The STSP can then be reformulated as

$$\min \left\{ c^T x : x \in \mathcal{R}_1 \right\}. \quad (1.6)$$

Attaching Lagrangian multipliers $\lambda_v, v \in V \setminus \{v_1\}$ to the degree constraints in (1.4), one ends up with the relaxed problem

$$\min \left\{ \bar{c}^T x + \beta : x \in \mathcal{X} \right\}, \quad (1.7)$$

where $\bar{c}$ are the Lagrangian modified edge costs and $\beta = 2 \sum_{v \in V \setminus \{v_1\}} \lambda_v$. Setting a value zero to $\lambda_{v_1}$, Lagrangian modified costs are computed as

$$\{ \bar{c}_e = c_e - \lambda_i - \lambda_j : e = (i, j) \in E \}. \quad (1.8)$$

Problem (1.7) can be solved in polynomial time in a two-step procedure. Firstly a minimum spanning tree (under the Lagrangian modified edge costs) is computed for the subgraph of $K_n$ induced by vertices in $V \setminus \{v_1\}$. Then the two cheapest edges (again under the Lagrangian modified edges costs) are added to that spanning tree.

The Held and Karp bound can be computed as

$$\max_{\lambda} \min_{\lambda} \left\{ \bar{c}^T x + \beta : x \in \mathcal{X} \right\}, \quad (1.9)$$

where $\Lambda = \mathbb{R}^{|V-1|}$. 
2. Relax and Cut

A large number of combinatorial optimization problems can be generically described in
terms of the following linear integer programming formulation:

$$\min \{ c^T y : Ay \geq b, y \in Y \} , \quad (2.10)$$

where $y \in \mathbb{B}^p$ is a $p$-dimensional binary $\{0, 1\}$ array of variables, $c \in \mathbb{R}^p$, $b \in \mathbb{R}^q$, $A$ is a real valued matrix of conformable dimensions and, finally, $Y \subseteq \mathbb{B}^p$ is an associated polyhedral region. Assume, as it is customary in Lagrangian relaxation, that

$$\min \{ c^T y : y \in Y \} \quad (2.11)$$

is an easy (polynomial time) problem to solve. On the other hand, in what is unusual for the application of Lagrangian relaxation, let $q$ be very large (exponential in $p$). In spite of that, assume that one wishes to dualize

$$\{ a_i y \geq b_i : i = 1, 2, \ldots, q \} \quad (2.12)$$
in a Lagrangian fashion and let $\lambda \in \mathbb{R}^q_+$ be the corresponding array of Lagrangian multipliers. Subgradient Optimization (SO) could then be used to solve

$$\max_{\lambda \geq 0} \min \{ (c - A^T \lambda)^T y + \lambda^T b : y \in Y \} . \quad (2.13)$$

Optimization is typically conducted here in an iterative way with multipliers being updated so that the optimal values of (2.13) is attained (see [5]). For the sake of completeness, let us briefly review the Subgradient Method (SM) (see [14]) which is used here. At any given iteration of the SM, for given feasible values of Lagrangian multipliers $\lambda$, let $\bar{y}$ be an optimal solution to

$$\min \{ (c - A^T \lambda)^T y + \lambda^T b : y \in Y \} . \quad (2.14)$$

Denote by $z_{lb}$ the value of this solution and let $z_{ub}$ be a known upper bound on (2.10). Additionally, let $G \in \mathbb{R}^q$ be a set of subgradients associated with the relaxed constraints. For this current solution, $\bar{y}$, an element of $G$ is evaluated as

$$g = b - Ay \in \mathbb{R}^q \quad (2.15)$$

In the literature (see [14], for instance), Lagrangian multipliers are usually updated by firstly determining a step size $\theta$,

$$\theta = \frac{\pi(z_{ub} - z_{lb})}{\| g \|^2} , \quad (2.16)$$

where $\pi \in (0, 2]$. One would then proceed to redefine $\lambda_i$ as

$$\lambda_i \equiv \max\{ 0; \lambda_i + \theta g_i \} , \quad i = 1, 2, \ldots, q \quad (2.17)$$

and then move on to the following iteration of the SM.
Under the conditions imposed here, the straightforward use of updating formulas (2.15)-(2.17) is not as simple as it might appear, the reason being the very large number of inequalities that are typically dualized. At every SM iteration, subgradients, as described above, could be divided in three groups. The first one would involve subgradients for those inequalities in (2.12) that are violated by $\bar{y}$. The second group would involve those subgradients $g_i$ for which, currently, $\lambda_i > 0$. Notice that a subgradient may be, simultaneously, in the two groups just described. Finally, the third group consists of the remaining subgradients and their evaluation would account for most of the computational burden at a SM iteration. One should notice that, under the proposed classification, subgradients may change groups from one SM iteration to another. It should also be noticed that the only multipliers that may contribute to the Lagrangian costs, $(c - AT\lambda)$, at a given SM iteration, are the ones associated with subgradients in groups one and two. Those multipliers are denoted active multipliers, and their associated subgradients, active subgradients. Conversely, subgradients on the third group are denoted as inactive multipliers. Finally, another point to be made is that, from (2.17), no multiplier associated with subgradients in group three will change its null value after the application of that updating formula.

Assume that one is at a typical iteration of the SM for a CO problem of nontrivial dimension, as described above. Furthermore, assume that one is capable of overcoming the (practically speaking) impossible task of computing the values for all the exponentially many inactive subgradients and then updating their values. If, on the one hand, inactive multipliers would not contribute to Lagrangian costs, inactive subgradients, on the other hand, would play a decisive role in determining the step size value $\theta$ in (2.16). Typically, for our application, $\theta$ would tend to be extremely small, leaving multiplier values virtually unchanged from iteration to iteration and thus jeopardizing the speed of convergence of the SO Method. Bearing this in mind, one may choose to apply (2.15)-(2.17) exclusively to active subgradients and multipliers. That results in a dynamic scheme where the set of active multipliers may continuously change. Notice, in association, that a multiplier may become active at one given SM iteration, then become inactive at a subsequent SM iteration and, yet again, become active at a later iteration. The introduction of this scheme (very much akin to cutting planes generation) into implicit enumeration for a problem with exponentially many inequalities as candidates for Lagrangian dualization has been firstly proposed and successively applied to the Steiner Problem in Graphs by Lucena ([15, 16]). Later on this approach has been used in Hunting, Faigle and Kern [12] for the Edge-Weighted Clique problem, in Moraes, Palmeira, Lucena and Porto for the Quadratic Knapsack problem, in Belloni and Lucena [1] for the Linear Ordering problem, and in Martinhon, Lucena and Maculan [17] for the Vehicle Routing problem.
A different approach, which also allows exponentially many inequalities as candidates for Lagrangian relaxation, has been introduced by Escudero, Guignard and Malik. It has been, very appropriately, coined \textit{relax and cut}. That approach differs from the one in \cite{15, 16}. Nevertheless, both of them operate within a Lagrangian relaxation framework and dualize inequalities \textit{on the fly}. Therefore the term \textit{relax and cut} appears adequate enough to label the algorithm in \cite{15, 16} as well, and that is adopted here.

For Lagrangian Relaxation based methods, the first attempt to allow exponentially many inequalities as candidates to dualization was suggested by Balas and Christofides for the Traveling Salesman Problem. The second such attempt was probably that of Gavish for Capacitated Minimum Spanning Tree Problem. Finally, more recently, Fisher proposed the dualization of an a \textit{priori} selected subset of a family of valid inequalities (with exponentially many members). All of these approaches are significantly different from the one suggested in Lucena \cite{15, 16}, which bears some resemblance with branch and cut algorithms (see Padberg and Rinaldi).

3. Separating Violated Inequalities

For our scheme to be successful, not only we need to be able to solve (2.14) efficiently for a fixed value of the multipliers $\lambda$, but also we need to be able to generate violated inequalities efficiently. That is, we need to assume that we can construct or we are given a \textit{Separation Oracle}, a procedure that returns a subset of inequalities violated by a given solution of the subproblem. Of course, the complexity of such oracle will depend on the particular family of inequalities that one is working with and on the structure of the subproblems’ solutions.

In many applications of relax and cut methods, the relaxed solution obtained on each iteration has some structure to be exploited. Very often the problem that is being solved on each iteration is another combinatorial problem itself.

3.1. Path Inequalities. The polytope associated with the TSP has been an object of extensive research during the past decades. A variety of facet-defining inequalities has been studied and is available in the literature (see \cite{14} for a survey). We will concentrate\footnote{In preliminary studies we also used Comb inequalities.} on subfamily of the Path inequalities proposed by Naddef and Rinaldi in \cite{18} called the p-regular k-wheelbarrow inequalities. More precisely, for any odd $k \geq 3$, $p \geq 2$, we choose a vertex $A$ and sets of vertices $\{H_i\}_{i=1}^p$ and $\{T_j = \{T_j^1, \ldots, T_j^p\}\}_{j=1}^k$, called \textit{handles} and \textit{teeth} respectively, which satisfy the following relations:

\begin{align*}
H_1 \subset H_2 \subset \cdots \subset H_p \\
H_1 = \left( A \cup \bigcup_{j=1}^k T_j^1 \right)
\end{align*}
\( T_j \setminus H_k \neq \phi \) for \( j = 1, \ldots, k \)

\( H_{i+1} \setminus H_i = \bigcup_{\ell=1}^{k} T_{i+1}^{\ell} \) for \( 1 \leq i \leq p - 1 \).

The corresponding closed form inequality is given by

\[
\sum_{i=1}^{p} x(\gamma(H_i)) + \sum_{j=1}^{k} x(\gamma(T_j)) \leq \sum_{i=1}^{p} |H_i| + \sum_{j=1}^{p} |T_j| - k(1 + p)/2.
\]

3.2. Separation Heuristic. For this test problem, the Separation Oracle can only use heuristics, since no efficient exact separation procedure is known. Fortunately, the Separation Oracle is called to separate inequalities from 1-Tree structures. In this case, thanks to integrality and the existence of exactly one cycle, the heuristic search for violated inequalities is much easier than for general sets.

We start by searching for candidates for the vertex \( A \), i.e., we search for all vertices with degree greater or equal to 3 in the 1-Tree solution. From this vertex, we try to expand an odd number of paths with the same number of nodes on each path.

4. A Lagrangian Heuristic for the TSP

As indicated by (2.16), upper bounds on the problem are required to conduct the proposed method. Regarding heuristics, the TSP is probably the richest problem of all. Our Lagrangian setup allows us to add dual information to any desired heuristic.

For instance, consider the Double Tree heuristic which traditionally guarantees a 2-approximation for the Euclidean case. Since on each iteration we have a 1-Tree solution available, we can double those edges and easily extract a tour which inherited most of our dual information.

In our implementation, we keep switching between this double tree variant and a traditional greedy heuristic which uses the Lagrangian costs instead of the original costs matrix. We call one of those heuristics on every iteration in which we obtain an improvement on the dual bound.

4.1. Local Search. The literature has shown the importance of local searches for the TSP and we agree with that appraisal. The remarkable work of Keld Helsgaun in [11] generates feasible bounds close enough to the optimal solution to fulfill our needs. Since we are not concerned with primal bounds in this study, we refer to [11] for details.

We do emphasize that we have more dual information than in [11], meaning that in principle we can generate more accurate search neighborhoods. In [11] implementation, the 5-opt procedure is called \(|V|\) times. Instead, we call the 5-opt moves only once after forming an initial tour from a Lagrangian heuristic.
5. Implementation Details

5.1. Dealing with Sparse Graphs. In order to address larger instances of the problem, it will be necessary not to deal with the complete graph. To compute the MSTs, we will restrict ourselves to a sparse subgraph which is dynamically updated accordingly to the reduced costs.

Only on the very first iteration we compute a MST with the complete graph. Once the solution is computed, all the reduced costs are generated. The subgraph that we are going to start with will contain the MST solution and the twenty edges with smallest reduced costs for each vertex. Later, we discovered that Keld Helsgaun proposed independently a similar scheme in [11]. Notice that such subgraph is guaranteed to be connected.

The disadvantage of any subgraph is that any bound obtained using it cannot be claimed to be a bound for the original problem. To handle this shortcoming, on each iteration on which a better bound is found using the subgraph, the reduced costs for the complete graph are computed. Clearly, if all of them are nonnegative, the bound was valid in the first place. If not, we have a set of edges with negative reduced costs. Using the main result in [2], the edges of the minimal spanning tree will be contained in the union of the edges with negative reduced costs and the solution found using the subgraph, thus obtaining the minimal spanning tree for the complete graph. These edges are also included in our subgraph for the next iteration.

Typically, many iterations of SM are needed in order to improve the current bound. It is exactly in this kind of situation that the previous procedure can improve running time the most by avoiding the MST computation with the complete graph in most iterations, and, occasionally, recomputing the reduced costs.

5.2. Computing Lagrangian Costs on the Fly. The traditional Held and Karp scheme has a really nice symmetry between the edges cost and the vector of Lagrangian multipliers. To compute the Lagrangian cost of a particular edge \( e = (i, j) \), one just needs to add the components \( \lambda_i \) and \( \lambda_j \) to the real cost \( c_e \). This is no longer true in our case. Fortunately, the path inequalities separated by our procedures tend to be concentrated among a small number of nodes, and thus only the costs of few edges will be affected. These modified edges are kept in a hash table with all the Lagrangian costs associated with all the dualized path inequalities. Such table needs to be verified every time we try to access the Lagrangian cost of any edge.

5.3. Fixing Variables. Through our algorithm, we keep generating primal and dual bounds for the problem. Once this duality gap becomes relatively small, we also try to fix variables in zero or in one.
Our main procedure is based on the reduced cost of an edge. We define the reduce cost of a variable as the change of cost it imposes in the current solution if we change its value. Let $z_{lb}$ and $z_{ub}$ denote respectively the dual bound obtained in the current iteration and a primal bound for the problem. Consider an edge $e$ not in the 1-Tree solution of the current iteration with reduced cost $r_e > z_{ub} - z_{lb}$. Thus, a lower bound for the original problem with the additional constraint of imposing $e$ on the optimal solution is greater than $z_{ub}$. So, $e$ can be permanently fixed in zero. Analogously, if an edge $e$ is in the solution of the current iteration with reduced cost $r_e > z_{ub} - z_{lb}$, such edge can be permanently fixed in one. We refer to [11] for a complete description on how to efficiently compute the reduced costs on a minimum spanning tree.

Moreover, we also perform the optimality tests proposed by Jonker and T. Volgenant in [13] if the previous method was capable of making the graph sparse. We impose this restriction since the running time of our implementation of these tests was prohibitively large to be used in large instances with a dense graph. It turns out that these test were useful only on small instances (smaller than 300 cities).

5.4. Advantages of Subgradient Optimization. Recently, Belloni and Sagastizábal in [3] built on this approach to propose the substitution of the Subgradient Method by a dynamic version of the Bundle Method in [5]. Under assumptions on the Separation Oracle, they manage to prove convergence to the linear relaxation with all constraints. Their proof relied heavily on the stability properties of traditional Bundle Methods and cannot be easily extended to the SM.

Numerical results in [3] reveal a remarkably small number of iterations reflecting the quality of the directions generated by their method. Unfortunately, in order to maintain consistency, it is necessary to keep track of previous solutions to generate subgradient components of new inequalities of old cutting planes. This process, combined with the quadratic problem to be solved on each iteration, increases dramatically the running time of each iteration for larger instances.

Since the SM has no memory, fixing variables along the iterations does not generate any additional problem. In the case of Bundle Methods, additional work would also be required to preserve the consistency of the affine cuts from previous iterations.

6. Computational Experiments

In order to validate our scheme, we conducted numerical tests on 31 Euclidean STSP instances from the TSPLIB, ranging from 1000 to 33810 cities. There are tables below showing performance with respect to bounds and running times.
The first table shows the value of the optimal tour\(^2\) (Optimal), the exact Held and Karp duality gap\(^3\) (H&K), the duality gap obtained by our dynamic scheme (R&C), the Gap Reduction Factor (Gap RF, defined as the ratio between the R&C and H&K gaps), and the number of dualized Path inequalities at the end of the algorithm (# Path).

We tested all the 31 instances with the same set of parameters, not performing any fine tuning procedure. Since a heuristic is being used as our separation oracle, and the SM is not guarantee to converge with our particular choice of step size, we cannot ensure to beat the exact Held and Karp bound. From our original set of 31 problems, only in four instances we failed to improve on the Held and Karp bound, i.e., in dsj1000, fl1400, fl1577, and fl3795. For the remaining 27 instances, we reduced the duality gap by a factor of 0.596 on average. In those four problems, we fail to improve the Held and Karp gap due to slow convergence of the subgradient method (see Conclusion for remarks).

Using our separation heuristic, the number of dualized path inequalities seems to grow roughly linearly with the number of vertices. More importantly, the total number of edges modified by the path inequalities seems also to exhibit a linear growth rate. In fact, we note that the Path Inequalities separated by our heuristic tended to be concentrated on a small subgraph. This suggests that there are hard regions in the graph where the degree constraints lack the flexibility to select the optimal edges, exactly where the Path Inequalities can play a major role. We illustrate this behavior using the instance ts225.

Table 6 reports the total running time (last column), and the fraction of that time spent on each main segment of the method: finding good feasible solution (Heuristic), separating violated inequalities (Separation), solving the minimum spanning tree problems (MST), constructing the data structure for the Lagrangian costs induced by the Path inequalities (Costs), and computing the reduced costs (Reduced).

The most important remark to make is that the time consumed in the separation oracle was always less than 2% of the total time. Thus, we are still allowed to try more sophisticated heuristics to find more inequalities and possibly obtain better bounds.

The time spent on finding feasible solutions is still very large. Our implementation doesn’t require a precise bound and we could possibly save a fraction of the total running time. Unfortunately, the relative time spent on heuristics tend to decrease as the instances increase. So, we cannot expect to rely only on that for larger instances.

\(^2\)Available at the TSPLib website.
\(^3\)Obtained by the Concord code through Linear Programming.
<table>
<thead>
<tr>
<th>Problem</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dsj1000</td>
<td>1000</td>
<td>18659688</td>
<td>0.604</td>
<td>1.52</td>
<td>2.527</td>
<td>370</td>
</tr>
<tr>
<td>pr1002</td>
<td>1002</td>
<td>259045</td>
<td>0.883</td>
<td>0.4852</td>
<td>0.549</td>
<td>364</td>
</tr>
<tr>
<td>u1060</td>
<td>1060</td>
<td>224094</td>
<td>0.644</td>
<td>0.412</td>
<td>0.639</td>
<td>341</td>
</tr>
<tr>
<td>vm1084</td>
<td>1084</td>
<td>239297</td>
<td>1.309</td>
<td>0.759</td>
<td>0.579</td>
<td>384</td>
</tr>
<tr>
<td>pcb1173</td>
<td>1173</td>
<td>56892</td>
<td>0.950</td>
<td>0.404</td>
<td>0.425</td>
<td>460</td>
</tr>
<tr>
<td>d1291</td>
<td>1291</td>
<td>50801</td>
<td>1.166</td>
<td>0.895</td>
<td>0.767</td>
<td>661</td>
</tr>
<tr>
<td>r11304</td>
<td>1304</td>
<td>252948</td>
<td>0.901</td>
<td>0.539</td>
<td>0.598</td>
<td>508</td>
</tr>
<tr>
<td>r11323</td>
<td>1323</td>
<td>270199</td>
<td>1.622</td>
<td>0.940</td>
<td>0.579</td>
<td>548</td>
</tr>
<tr>
<td>nrw1379</td>
<td>1379</td>
<td>56638</td>
<td>0.426</td>
<td>0.188</td>
<td>0.442</td>
<td>561</td>
</tr>
<tr>
<td>fl1400</td>
<td>1400</td>
<td>20127</td>
<td>1.709</td>
<td>2.861</td>
<td>1.670</td>
<td>198</td>
</tr>
<tr>
<td>u1432</td>
<td>1432</td>
<td>152970</td>
<td>0.284</td>
<td>0.140</td>
<td>0.494</td>
<td>239</td>
</tr>
<tr>
<td>fl1577</td>
<td>1577</td>
<td>22249</td>
<td>1.631</td>
<td>3.604</td>
<td>2.209</td>
<td>735</td>
</tr>
<tr>
<td>d1655</td>
<td>1655</td>
<td>62128</td>
<td>0.931</td>
<td>0.589</td>
<td>0.632</td>
<td>620</td>
</tr>
<tr>
<td>vm1748</td>
<td>1748</td>
<td>336556</td>
<td>1.335</td>
<td>0.658</td>
<td>0.492</td>
<td>675</td>
</tr>
<tr>
<td>u1817</td>
<td>1817</td>
<td>57201</td>
<td>0.896</td>
<td>0.620</td>
<td>0.692</td>
<td>825</td>
</tr>
<tr>
<td>r11889</td>
<td>1889</td>
<td>316536</td>
<td>1.526</td>
<td>0.741</td>
<td>0.485</td>
<td>707</td>
</tr>
<tr>
<td>d2103</td>
<td>2103</td>
<td>80450</td>
<td>1.420</td>
<td>1.373</td>
<td>0.966</td>
<td>1248</td>
</tr>
<tr>
<td>u2152</td>
<td>2152</td>
<td>64253</td>
<td>0.614</td>
<td>0.460</td>
<td>0.749</td>
<td>658</td>
</tr>
<tr>
<td>pr2392</td>
<td>2392</td>
<td>378032</td>
<td>1.201</td>
<td>0.361</td>
<td>0.300</td>
<td>886</td>
</tr>
<tr>
<td>pcb3038</td>
<td>3038</td>
<td>137694</td>
<td>0.803</td>
<td>0.388</td>
<td>0.483</td>
<td>1262</td>
</tr>
<tr>
<td>fl3795</td>
<td>3795</td>
<td>28772</td>
<td>1.024</td>
<td>4.038</td>
<td>3.941</td>
<td>1314</td>
</tr>
<tr>
<td>fnl4461</td>
<td>4461</td>
<td>182566</td>
<td>0.546</td>
<td>0.228</td>
<td>0.419</td>
<td>1829</td>
</tr>
<tr>
<td>rl5915</td>
<td>5915</td>
<td>565530</td>
<td>1.535</td>
<td>0.899</td>
<td>0.586</td>
<td>2014</td>
</tr>
<tr>
<td>rl5934</td>
<td>5934</td>
<td>556045</td>
<td>1.362</td>
<td>0.935</td>
<td>0.686</td>
<td>2121</td>
</tr>
<tr>
<td>pla7397</td>
<td>7397</td>
<td>23260728</td>
<td>0.577</td>
<td>0.437</td>
<td>0.7577</td>
<td>1444</td>
</tr>
<tr>
<td>rl11849</td>
<td>11849</td>
<td>923288</td>
<td>1.008</td>
<td>0.601</td>
<td>0.596</td>
<td>4015</td>
</tr>
<tr>
<td>usa13509</td>
<td>13509</td>
<td>19982859</td>
<td>0.657</td>
<td>0.384</td>
<td>0.584</td>
<td>4947</td>
</tr>
<tr>
<td>brd14051</td>
<td>14051</td>
<td>469388</td>
<td>0.481</td>
<td>0.448</td>
<td>0.930</td>
<td>6511</td>
</tr>
<tr>
<td>d15112</td>
<td>15112</td>
<td>1573084</td>
<td>0.521</td>
<td>0.244</td>
<td>0.469</td>
<td>6938</td>
</tr>
<tr>
<td>d18512</td>
<td>18512</td>
<td>[645198,645244]</td>
<td>[0.477,0.484]</td>
<td>[0.212,0.219]</td>
<td>[0.438,0.459]</td>
<td>8511</td>
</tr>
<tr>
<td>pla33810</td>
<td>33810</td>
<td>[66005185,66050535]</td>
<td>[0.454,0.522]</td>
<td>[0.347,0.416]</td>
<td>[0.764,0.796]</td>
<td>7096</td>
</tr>
</tbody>
</table>

The times for computing the Reduced Costs and the minimum spanning tree should be seen together. If we can keep the number of reduced costs computations small with respect to the total number of minimum spanning trees computations, we can expect to save a lot
Figure 1. The figure on the left displays the optimal tour for the instance ts225, while the figure on the right displays not only the best 1-Tree solution found but also all edges modified by 256 Path Inequalities. Each vertex with a circle around it was the center for a Path Inequality. In this instance, the duality gap reduced from 8.71% to 0.94% when the Path Inequalities are introduced.

of computation time. For example, for the instance rl11849 we computed the reduced costs 196 times while we computed 5000 minimum spanning trees. In this case, the reduced costs computation represented more than 60% of the total time while the minimum spanning trees represented less than 30%. We note that the computation of the reduced costs grows faster than the number of edges of the complete graph, while the minimum spanning tree computed has a number of edges which is linear with the number of vertices. So as the instances grow, we would like the ratio between the number of reduced costs computation and minimum spanning tree computation to decrease. For example, in fl3795 we computed the reduced costs 167 times against 5000 computations of minimum spanning trees. In this instance, the reduced costs were responsible for less than 23% of the running time while the minimum spanning tree more than 34%. A way to accomplish the decease in the ratio between the number of reduced costs computation and minimum spanning tree computation is to not compute the reduced costs on every improvement on the lower bound using the subgraph as our implementation is performing.
7. Conclusions and Future Work

In this paper we have presented a method to improve on the celebrated Held and Karp bound for the TSP without increasing the complexity of solving the subproblem, that is, the
computation of the 1-tree. Herein the path inequalities were used to strength the relaxation reducing the duality gap by a factor of 0.6 on average without increasing the running times significantly.

The resulting nondifferentiable problem obtained has the peculiar property that its dimension is allowed to vary. This raises several questions regarding to the convergence properties of the Subgradient Method proposed here. In particular, four instances exhibited bad convergence properties (yielding a bound worse than the original Held and Karp bound!). There are at least two possible sources for this behavior. First, the Separation Oracle used was not exact. Second, possible zig-zag behavior of the traditional subgradient method could be worse if we are adding and removing the same inequalities several times. These questions are definitely relevant and will be investigated in future works (see [3]).

References


