Prize Collecting Travelling Salesman Problem
Fast Heuristic Separations

Kamyar Khodamoradi and Ramesh Krishnamurti
School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC, Canada

Keywords: Prize Collecting TSP, Linear Programming, Generalized Subtour Elimination Constraints, Primitive Comb Inequalities, Integrality Gap, Shrinking Heuristic.

Abstract: The Prize Collecting Travelling Salesman Problem (PCTSP) is an important generalization of the famous Travelling Salesman Problem. It also arises as a sub problem in many variants of the Vehicle Routing Problem. In this paper, we provide efficient methods to solve the linear programming relaxation of the PCTSP. We provide efficient heuristics to obtain the Generalized Subtour Elimination Constraints (GSECs) for the PCTSP, and compare its performance with an optimal separation procedure. Furthermore, we show that a heuristic to separate the primitive comb inequalities for the TSP can be applied to separate the primitive comb inequalities introduced for the PCTSP. We evaluate the effectiveness of these inequalities in reducing the integrality gap for the PCTSP.

1 INTRODUCTION

The Prize Collecting Travelling Salesman Problem (PCTSP) is a variant of the Travelling Salesman Problem (TSP), and is important in its own right (Wolsey, 1998). It also arises as a sub problem that needs to be solved in order to extract a column in column generation formulations of various vehicle routing problems. PCTSP is an NP-hard problem, and has received a lot of attention in the literature. The PCTSP comprises a complete graph with a depot node. In addition to costs (distances) between nodes, each node has a 'prize' associated with it. The objective is to derive a tour which includes the depot, and maximizes the sum of the prizes associated with the nodes in the tour, less the cost of the tour. It can be considered as a generalization of the TSP, since the TSP is the PCTSP with a prize of 0 associated with each node. Several variants of the PCTSP have been studied in the literature. The version of the PCTSP we study in this paper was first introduced by Balas (Balas, 1989) in a more general setting to model the scheduling of the daily activities of a steel rolling mill.

In the version introduced by Balas, a tour profits (to the extent of a prize associated with the node) from each node it visits, while it is penalized (to the extent of a penalty associated with the node) for every node it does not visit. The profit/penalty for each node corresponds to the prize associated with the node. The objective is to obtain a tour such that the total prize collected exceeds a prescribed amount, while minimizing the sum of the travel cost and penalties. The travel cost for a tour is the sum of the distances between consecutive nodes in the tour. Balas (Balas, 1989) derives the cuts for the PCTSP corresponding to the subtour elimination constraints for the TSP. The cuts we use in this paper, called the Generalized Subtour Elimination Constraints (GSECs) was first proposed by Goemans (Goemans, 1994) in the context of the Steiner tree problem. See also Wolsey (Wolsey, 1998) for a clear treatment of our version of the PCTSP. In a continuation of his work on PCTSP, Balas (Balas, 1995) derives, among other cuts, the cuts corresponding to the primitive comb inequalities for the TSP. The separation heuristic we use for these cuts is a heuristic given by Padberg and Hong (Padberg and Hong, 1980) for detecting blossoms for the TSP. We show that these heuristics can also be used as separation procedures for the primitive comb inequalities for the PCTSP. We use these heuristics to obtain LP solutions with improved integrality gap.

The first known solution procedure to solve the PCTSP exactly was a branch-and-bound procedure (Fischetti and Toth, 1988). Fischetti et al. (Fischetti et al., 1997) also study a branch-and-cut algorithm for the symmetric case of Generalized Traveling Salesman problem, in which the nodes are split into clusters, and any feasible solution to the problem should cover at

Khodamoradi, K. and Krishnamurti, R.
Prize Collecting Travelling Salesman Problem - Fast Heuristic Separations.
DOI: 10.5220/0005758103800387
In *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems (ICORES 2016)*, pages 380-387
ISBN: 978-989-758-171-7

least one node from each cluster. The symmetric property refers to the fact that the cost of going from a node u to another node v is the same is the cost of v to u. In more recent work, Chaves and Lorena (Chaves and Lorena, 2008) propose a hybrid metaheuristic for the problem and compare its performance with CPLEX. Bérubé, Gendreau, Potvin (Bérubé et al., 2009) propose a branch-and-cut algorithm to solve a variant of the PCTSP. In this variant, the objective is to obtain a tour with minimum travel cost, subject to the constraint that the total prize collected exceeds a predetermined amount. They incorporate many valid inequalities for their variant of the PCTSP, and evaluate the performance of a branch-and-cut algorithm by introducing these inequalities.

Our version of the PCTSP arises directly as a sub problem in a variant of the vehicle routing problem, called the Skill Vehicle Routing Problem (Cappanera et al., 2011). In this paper, we focus on the LP solution procedure for this version of the PCTSP. Furthermore, we use fast separation heuristics for both the GSECs and the primitive comb inequalities. For the GSECs, we adapt a fast heuristic, and compare its performance with an exact procedure, both in the quality of its solution, as well as in its running time. For the primitive comb inequalities, we show that a well-known separation heuristic for the TSP (Padberg and Hong, 1980) also works as a separation procedure for the PCTSP. We also compare the GSEC cuts with the primitive comb inequalities in their quality of solutions.

2 NOTATION AND PROBLEM FORMULATION

We let $G = (S,E)$ denote the complete graph representing the instance of the problem, with node $r \in S$ denoting the depot which every tour must visit. Cost c_e associated with each edge $e \in E$ is the Euclidean distance between the two endpoints of edge e. In the PCTSP, the salesman may choose to visit city $j \in S$. If he does so, then he obtains a prize β_j but incurs a travel cost c_e if he traverses edge $e = (i, j)$. The salesman must start and end his tour at node r, and maximize the total prize he collects, less the cost of the tour.

We provide below the ILP formulation for the PCTSP (Wolsey, 1998). In the formulation below, we let decision variables y_j be 1 if the salesman visits city j (and 0 otherwise), and x_e be 1 if he traverses edge e (and 0 otherwise). We also let set $S' = S \cup \{r\}$ and $E' = E \backslash \{\delta(r)\}$, where $\delta(r)$ is the set of edges incident on the depot node r.

2.1 Problem Formulation

$$\max \sum_{j \in S} \beta_j y_j - \sum_{e \in E} c_e x_e \tag{2.1}$$

subject to

$$\sum_{e \in \delta(i)} x_e = 2y_i \qquad \forall i \in S \tag{2.2}$$

$$\sum_{e \in E(V)} x_e \leqslant \sum_{i \in V \backslash \{k\}} y_i \qquad \forall k \in V, V \subseteq S' \tag{2.3}$$

$$y_r = 1 \tag{2.4}$$

$$x_e \in \{0,1\}, y_j \in \{0,1\} \quad \forall e \in E, \forall j \in S \tag{2.5}$$

The integer variable y_l is set to 1 (0) if node $l \in S$ is included (not included) in the tour. Similarly, the integer variable x_e is set to 1 (0) if edge $e \in E$ is included (not included) in the tour. Note that $\sum_{j \in S} \beta_j y_j = \sum_{j \in T} \beta_j$ and $\sum_{e \in E} c_e x_e = \sum_{e \in T} c_e = C(T)$. Constraint 2.2 ensures that if node l is included in the tour, then two edges of the tour must be incident on it. Constraint 2.3 is the generalized sub tour elimination constraint (GSEC). Constraints of this form are used to prevent any sub tour that does not include root node r. These are generalizations of the sub tour elimination constraints for the standard travelling salesman problem.

Clearly, there are an exponential number of constraints included in the GSECs and we cannot include them all to solve the sub problem. To get around this problem, we include these constraints as they are needed (whenever there is a sub tour that does not include node r). It is easy to detect such sub tours when the decision variables take $0/1$ values. However, because we solve the LP relaxation of the sub problem, fractional values may be assigned to the decision variables y_j and x_e. We outline below the method that can be used to detect for such sub tours (which do not include node r) when fractional values are assigned to the decision variables.

Let the LP solution to the sub problem be given by $(x*, y*)$. Then the generalized sub tour elimination constraint is violated for a subset $W \subseteq S'$ of nodes if the inequality $\sum_{e \in E'(W)} x_e^* > \sum_{l \in W \backslash \{k\}} y_l^*$ is satisfied. Such a subset W can be extracted by solving the following integer program. We call the problem below the separation problem for GSECs. The formulations for the separation problem for GSECs for the PCTSP are given in Wolsey 1998 (Wolsey, 1998).

3 THE SEPARATION PROBLEM FOR GENERALIZED SUBTOUR ELIMINATION CONSTRAINTS

A formulation for the separation problem for GSECs is given below (Wolsey, 1998).

$$\zeta_k = max \sum_{e \in E'} x_e^* w_e - \sum_{j \in W \setminus \{k\}} y_j^* z_j \tag{3.1}$$

subject to

$$w_e \leqslant z_i, w_e \leqslant z_j \qquad \forall e = (i,j) \in E' \tag{3.2}$$

$$w_e \geqslant z_i + z_j - 1 \qquad \forall e = (i,j) \in E' \tag{3.3}$$

$$w_e \in \{0,1\}, z_j \in \{0,1\} \quad \forall e \in E', \forall j \in W \tag{3.4}$$

$$z_k = 1 \tag{3.5}$$

Constraint 3.3 above can be dropped because it is implied by Constraint 3.2 if $x_e^* \geqslant 0$ for $e \in E'$ (Wolsey, 1998). It turns out that the constraint matrix for the above separation problem (after Constraint 3.3 is dropped) is totally unimodular. Thus, in polynomial time, we solve the LP relaxation of the separation problem to obtain constraint(s) to introduce into the sub problem.

The optimal solution to the LP relaxation of the modified separation problem is integral. This optimal solution corresponds to the subset $W \subseteq S'$ and node $k \in W$ such that the inequality $\sum_{e \in E'(W)} x_e^* \leqslant \sum_{l \in W \setminus \{k\}} y_l^*$, corresponding to Constraint 2.3 of the sub problem is violated. This inequality is introduced into the sub problem and the LP relaxation of the PCTSP is again solved. This is repeated until there is no subset $W \subseteq S'$ and node $k \in W$ such that the inequality $\sum_{e \in E'(W)} x_e^* \leqslant \sum_{l \in W \setminus \{k\}} y_l^*$ is violated.

In practice, solving an LP for each $k \in W$ can be very time consuming. Therefore, a *shrinking* heuristic is used to speed up the separation algorithm. The shrinking heuristic is described below.

3.1 A Shrinking Heuristic Algorithm for the Separation of GSECs

The shrinking heuristic we describe below helps find many violated GSECs quickly. We generalize the heuristic developed by Crowder and Padberg (Crowder and Padberg, 1980) and Land (Land, 1979) for the TSP. Recall that $S' = S \setminus \{r\}$ and $E' = E \setminus \{\delta(r)\}$. In this approach, the GSEC inequalities are transformed into the equivalent cut-set inequalities. In the following, $E(V_1, V_2)$ is used to denote the set of edges which have one endpoint in V_1 and the other in V_2.

$$\sum_{e \in E(V, S' \setminus V)} x_e \geqslant 2y_k \qquad \forall k \in V, V \subseteq S' \tag{3.6}$$

Note that the cut-set inequalities can be derived from Constraints 2.2 and 2.3. These inequalities allow us to transform the problem to a network flow problem in which we seek a minimum cut in a rather special sense. We are looking for violated cut-set constraints. Equivalently, we wish to find a subset $V \subseteq S'$ of vertices for which, $\sum_{e=(i,j), i \in V, j \in S' \setminus V} x_e < 2y_k$ for some $k \in V$. Alternatively, we look for a cut $(V, S' \setminus V)$ that minimizes

$$\sum_{e \in E(V, S' \setminus V)} x_e - 2 \max_{k \in V} \{y_k\}. \tag{3.7}$$

There exists a k for which Inequality 3.6 is violated if and only if the value of a cut that minimizes 3.7 is negative. To find such a cut, we repeatedly reduce the size of the graph by shrinking subsets of vertices into a single vertex, provided that certain conditions are met.

Consider two subsets, $V_1, V_2 \subseteq S'$. If V_1 and V_2 are to be merged, the value contributed to the cut value by V_1 should not be worsened after the merge. This means that we should have:

$$\sum_{e \in E(V_1, S' \setminus V_1)} x_e - 2 \max_{k \in V_1} \{y_k\} \geqslant \sum_{e \in E(V_1 \cup V_2, S' \setminus (V_1 \cup V_2))} x_e - 2 \max_{k \in V_1 \cup V_2} \{y_k\}, \tag{3.8}$$

By symmtery, a similar inequality can be written for V_2. Without going into too much detail, we note that we can merge (or shrink) V_1 and V_2 if

$$\sum_{e \in E(V_1, V_2)} x_e \geqslant \sum_{i \in V_2} y_i + \max_{k \in V_1} \{y_k\} - \max_{k \in V_1 \cup V_2} \{y_k\} - \sum_{e \in E(V_2)} x_e. \tag{3.9}$$

and

$$\sum_{e \in E(V_1, V_2)} x_e \geqslant \sum_{i \in V_1} y_i + \max_{k \in V_2} \{y_k\} - \max_{k \in V_1 \cup V_2} \{y_k\} - \sum_{e \in E(V_1)} x_e. \tag{3.10}$$

The inequalities (3.9) and (3.10) is the basis of our shrinking heuristic. We start with the original graph $G = (S', E')$, and consider pairs of vertices for shrinking. As we proceed with the shrinking algorithm, a vertex v_i may become a *super node* and represent a set of vertices in the original graph G. Thus, we associate a value m_i to every vertex of the graph v_i which denotes the maximum value of y_k's in the corresponding component of v_i in the original graph. Whenever we merge two vertices v_i and v_j connected via edge e into a new vertex $v_{i'}$, we set the new values $y_{i'} = y_i + y_j - x_e$, and

$m_{i'} = \max\{m_i, m_j\}$ and also adjust the edge weights for the newly created node. The following pseudo-code best describes our shrinking algorithm:

Algorithm 1: The Shrinking Heuristic.

Data: Graph $G = (S'V')$, Edge weights $w : E' \to \mathbb{R}^{\geqslant 0}$, and vertex values $y : S' \to \mathbb{R}^{\geqslant 0}$
Result: The cut V that maximizes $\sum_{e \in E(V, S' \setminus V)} x_e - \max_{k \in V)}\{y_k\}$
Initialize each node to be a separate component
for *every edge* $e = (i, j)$ *in* E' **do**
  if $(x_e > y_i - \max\{0, m_i - m_j\})$ ***and*** $(x_e > y_j - \max\{0, m_j - m_i\})$ **then**
    Merge nodes i and j into a new node ℓ
    for *every node* u *in* S' **do**
      if u *is adjacent to* i *or* j **then**
        Let e' denote the edge (ℓ, u)
        $x_{e'} \leftarrow x_{(u,i)} + x_{(u,j)}$
      end
    end
    $y_\ell \leftarrow y_i + y_j - x_e$
    $m_\ell \leftarrow \max\{m_i, m_j\}$
  end
end

4 PRIMITIVE COMB INEQUALITIES FOR THE TSP

To improve the quality of fractional solution to the PCTSP, valid inequalities can be introduced. Primitive comb inequalities are a class of valid inequalities. In this section, we study these valid inequalities for our version of the PCTSP. Later in Section 6, we investigate how effective the primitive comb inequalities are in improving the quality of fractional solutions to the PCTSP.

4.1 Formulation

Comb inequalities are valid inequalities that have been introduced successfully to obtain LP solutions with reduced integrality gaps for the TSP. In addition to this, exact separation algorithms for the comb inequalities have been proposed for the TSP. Because the LPs are solved many times in a branch and bound algorithm to solve the TSP optimally, efficient heuristics have also been proposed for the separation problem which find violated comb inequalities.

In this section, we look at inequalities for the simplest form of such structures known as *primitive combs*. A general comb consists of a set of nodes H known as the *handle*, and a set of t *teeth*, each tooth being a set of nodes that spans out of the handle. A primitive comb restricts each tooth to be a single edge. Primitive comb inequalities have been shown to improve the quality of the fractional solution to the PCTSP.

Let us first look at the primitive comb inequalities for the Travelling Salesman polytope. In what follows, $x(A, B) = \sum_{e=(i,j): i \in A, j \in B} x_e$, where x_e is 1 if the edge e is incorporated in the tour, and x_e is 0 otherwise.

$$x(H, H) + \sum_{j=1}^{t} x(T_j, T_j) \leqslant |H| + \frac{t-1}{2}, \qquad (4.1)$$

where H, the *handle*, and T_j for $j = 1, 2, \ldots, t$, the *teeth*, are sets of nodes satisfying the following conditions (Gutin and Punnen, 2002):

$$\left|H \cap T_j\right| = \left|T_j \setminus H\right| = 1 \qquad j = 1, \ldots, t, \text{ with } t \geqslant 3 \, odd \qquad (4.2)$$

$$T_i \cap T_j = \emptyset \qquad i, j = 1, \ldots, t \qquad (4.3)$$

Padberg and Hong (Padberg and Hong, 1980), among others, provide a heuristic separation algorithm for primitive comb inequalities for the TSP. We refer to this as the *odd-component heuristic*.

4.2 Odd-component Heuristic for TSP

In this heuristic, given an optimal LP solution x^* to a TSP instance over a complete graph (V, E), the graph $G^*_{1/2}$ is constructed. This graph has the vertex set V and edge set $\{e \in E : 0 < x^*_e < 1\}$. Let the resulting graph comprise of q components, whose vertex sets are $V_1, V_2, \ldots, V_q$. For each component $i, 1 \leqslant i \leqslant q$, the heuristic determines the subset of edges T_i in the set $\delta(V_i)$ (the set of edges which cross the set V_i) whose LP values are 1. T_i is thus given by $T_i = \{e \in E : x^*_e = 1 \wedge e \in \delta(V_i)\}$. If the cardinality of this set is odd, then the following simple procedure determines a set that either violates the subtour inequality for TSP, or a primitive comb inequality. If two edges in set T_i share a vertex $v \in V \setminus V_i$, then vertex v is included in the set V_i and the procedure is repeated until the edges in T_i are pairwise disjoint. Such a set V_i violates the subtour inequality if $|T_i| = 1$, and the primitive comb inequality if $|T_i| \geqslant 3$. Note that if two teeth T_i and T_j intersect outside if H, adding the common vertex to H would get rid of exactly two teeth, therefore the parity of the teeth of H is preserved. Therefore if a handle has odd number of teeth (which indicated either a violated GSEC or a violated primitive comb inequality) before the removal of a pair of teeth, it will still have odd many teeth afterwards.

Balas (Balas, 1995) shows an equivalent version of the comb inequalities are facet defining for the

Prize Collecting Travelling Salesman polytope (Balas, 1995). The following section provides the details of these comb inequalities for our version of the PCTSP.

5 PRIMITIVE COMB INEQUALITIES FOR PCTSP

5.1 Formulation

Balas shows in (Balas, 1995) that the following inequality defines a facet of the Prize Collecting TS polytope. Therefore inequality (4.1) translates to inequality (5.1) in the case of the Prize Collecting TSP.

$$x(H,H)+\sum_{j=1}^{t}x(T_j,T_j)+z(H)\leqslant|H|+\frac{t-1}{2}, \tag{5.1}$$

Here, $z(H)=\sum_{v\in H}z(v)$, and $z(v)$ is 1 if the vertex v is left out of the optimal tour (we need to a pay a penalty for it), and $z(v)$ is 0 otherwise. When compared to the formulation of the sub problem (Objective Function 2.1 and Constraints 2.2–2.5), one can write $y(v)=1-z(v)$ for all vertices v.

We show below that a connected component heuristic which has been used by Hong (Hong, 1972) and Land (Land, 1979) can be applied to separate the primitive comb inequalities introduced by Balas for the PCTSP.

5.2 Odd-component Heuristic for PCTSP

The odd-component heuristic for Prize Collecting TSP (Padberg and Hong, 1980) works with an equivalent formulation of the primitive comb inequalities for the Travelling Salesman polytope:

$$x(\delta(H))+\sum_{j=1}^{t}x(\delta(T_j))\geqslant 3t+1. \tag{5.2}$$

To be able to use the same heuristic, we first translate Balas's inequality into a similar form:

$$x(\delta(H))+\sum_{j=1}^{t}x(\delta(T_j))\geqslant 3t+1-2\sum_{j=1}^{t}z(T_j).$$

Lemma 1. *The following inequality defines a facet of the Prize Collecting TS polytope.*

$$x(\delta(H))+\sum_{j=1}^{t}x(\delta(T_j))\geqslant 3t+1-2\sum_{j=1}^{t}z(T_j).$$

Proof. By (Balas, 1995), we know that inequality (5.1) is facet defining. Also, from Constraint 2.2, the fractional degree of each node v is $2\cdot y(v)=2-2\cdot z(v)$. Therefore, we can write

$$2\,x(H,H)+x(\delta(H))=\sum_{v\in H}deg(v)=2|H|-2\,z(H).$$

Thus,

$$x(H,H)=|H|-z(H)-\frac{1}{2}\,x(\delta(H)).$$

Similarly, we have that

$$2\,x(T_j,T_j)+x(\delta(T_j))=\sum_{v\in T_j}deg(v)=4-2\,z(T_j),$$

for $j=1,\ldots,t$, which yields

$$x(T_j,T_j)=2-z(T_j)-\frac{1}{2}\,x(\delta(T_j)).$$

Therefore,

$$x(H,H)+\sum_{j=1}^{t}x(T_j,T_j)+z(H)=$$
$$|H|-\frac{1}{2}\,x(\delta(H))+2t-\sum_{j=1}^{t}z(T_j)-\frac{1}{2}\,\sum_{j=1}^{t}x(\delta(T_j)) \tag{5.3}$$

From equation (5.3) and inequality (5.1),

$$\frac{1}{2}\,x(\delta(H))+\frac{1}{2}\,\sum_{j=1}^{t}x(\delta(T_j))\geqslant 2t-\sum_{j=1}^{t}z(T_j)-\frac{t-1}{2}.$$

Thus,

$$x(\delta(H))+\sum_{j=1}^{t}x(\delta(T_j))\geqslant 3t+1-2\,\sum_{j=1}^{t}z(T_j).$$

□

We show below why the odd-component heuristic used by Padberg and Hong to separate primitive comb inequalities for the TSP can be used to separate primitive comb inequalities for the PCTSP without any modifications.

Assume that after running the odd-component heuristic on $G^*_{1/2}$, a component V_i has an odd number of teeth. We remove all non-disjoint teeth and add their common vertex to V_i. Let the handle H be V_i. We assume H has $t\geqslant 3$ teeth as we are more interested in finding violated primitive comb inequalities rather than violated GSECs. By the structure of $G^*_{1/2}$, all the outgoing edges of H have weight exactly equal to 1. Thus, $x(\delta(H))=t$. Consider a tooth of H, say $T_k=(u_k,v_k)$ (See Figure 1). Because of the degree constraints, and the fact that the edge (u_k,v_k) is taking a capacity q from each of the two endpoints, we

can write $x(\delta(T_k)) = 2(1-y_{u_k}) - 1 + 2(1-y_{v_k}-1) = 2t - 2(y_{u_k} + y_{v_k}$. Summing over all t edges and adding $x(\delta(H))$ we get:

$$x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j)) = 3t - 2\sum_{j=1}^{t} y(T_j)$$

This is exactly 1 short of satisfying the corresponding primitive comb inequality, therefore any component of $G^*_{1/2}$ with odd number of teeth would correspond to a violated inequality that we can introduce as a cut.

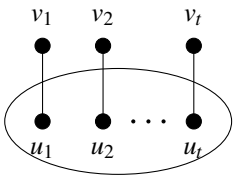

Figure 1: A component of $G^*_{1/2}$.

In the next section, we empirically analyze the performance of this heuristic alongside with the shrinking heuristic for GSECs.

6 COMPUTATIONAL RESULTS

In this section, we report the efficacy of the two adapted heuristics and analyze the behaviour of the corresponding separations. The heuristics have been implemented in C++ and the LP was solved using CPLEX 12.5. Then the code was run on an Intel 2.8 GHz processor with a maximum time limit of 14,400 seconds (4 hours).

In this section, we first describe the instance that we have used. Then we discuss the experiments we designed to test the behaviour and performance of the two heuristics, namely the shrinking heuristic for separation of the GSECs and the odd-component heuristic for separation of the primitive comb inequalities. We next explain how we compared these results and analyze their relative gap with respect to the optimal integral solution whenever such a solution is available. We summarize the results in Tables 1 and 2.

6.1 Instances

A total of 42 instances out of the TSPLIB library (Reinelt, 1991) have been transformed for our experiments. Out of the 43 instances, 10 were VRP instances and 32 symmetric TSP instances. In the VRP instances, each node is associated with a demand value. We use this demand as the prize of covering that node. For the TSP instances, which do not come with demands, we merely generate node prizes independently and uniformly at random in the range of 1 to 200.

6.2 Designing the Experiments

We analyze the adapted heuristics based on three difference aspects: *i.* the solution quality, *ii.* the number of cuts they detect, *iii.* the running time. The major factor in the success of a heuristic is indeed if it can produce near optimal solutions in a timely manner. Nevertheless, the number of cuts a heuristic can generate can give us insight about how well that heuristic is adapted for a certain type of polytope. If near optimality can be achieved with fewer cuts, we have some empirical indications that the heuristic is well-suited for the polytope of the given problem. For this reason, we also report the number of cuts for both the heuristics, as well as for an optimal procedure that uses LP for separation of GSECs. In what follows, we explain the method used for making the comparisons.

Shrinking Heuristic. To evaluate the efficacy of the shrinking heuristic, we compare them against a procedure that uses linear programming for separation of the GSECs, that is, a formulation that solves a cut-set equivalent of (3.1)–(3.5) for each node to find the violated GSECs. Three parameters mentioned above are reported for the heuristic GSEC separation, as well as the LP separation.

Odd-component Heuristic. To evaluate this heuristic, we first solve instances with the GSEC heuristic only, and then use odd-component heuristic to find violated primitive comb constraints. This way, we can measure how the solution quality may improve and how many new cuts are introduced, and then weigh it against the extra time we need to spend for running the second heuristic. The improvements are also reported in the right-most column of Table 2. the third column of Table 2 shows the number of extra cuts detected by using the second heuristic on top of the first. We also solve the ILP version of the problem using the built-in branch-and-bound method of CPLEX using regular Subtour Elimination Constraints (SECs) and compare them against the optimal LP solutions obtained by using both heuristics.

Note that to separate SEC inequalities, all we need to do is to find an integral subtour in a solution, which can be done in polynomial time using a connected component algorithm. This method can be slow for larger instances, therefore in many cases we have not been able to solve the ILP optimally in the time limit of 4 hours. For such instances, the gap percentages has been left blank. Also, in Table 1, "t.l." stands for *too long*, and is used for running times that are greater than 14,400 seconds.

Table 1: Computational results for the heuristic GSEC separation.

	LP GSEC Sep.			Heuristic GSEC Sep.		
Instance	Obj.	Cuts	Time	Obj.	Cuts	Time
eil13	3228.61	0	<1	3228.61	0	<1
ulysses22	-1964.93	16	1	-1964.93	13	<1
bayg29	-44.35	33	4	-44.35	14	<1
eil33	-2924.76	44	7	-2924.76	18	<1
att48	1358.75	0	7	1358.75	0	<1
hk48	551.65	1	7	551.65	1	<1
eil51	-4454.6	2	32	-4454.6	2	<1
st70	-6433.23	94	423	-6433.23	26	1
eilB76	-7442.17	28	964	-7442.31	11	3
eilC76	-6898.96	27	354	-6900.36	7	1
eilD76	-6774.68	33	434	-6774.74	15	2
pr76	5568.42	0	50	5568.42	0	<1
gr96	-9531.81	83	2740	-9531.81	26	2
rat99	9260.24	80	4802	-9260.87	15	14
rd100	-10119.1	189	2501	-10119.1	35	2
eil101	-8988.51	73	1108	-8988.51	17	5
eilA101	-9160.86	60	3153	-9160.86	17	1
eilB101	-9710.61	61	1107	-9710.61	18	2
lin105		N/A	t.l.	-87.93	354	4371
pr107	339.84	0	238	339.84	0	<1
gr120	-10451.1	85	3367	-10451.1	31	80
gr137	-13070.5	112	4306	-13070.5	26	19
pr144	532.69	11	4677	532.69	6	<1
ch150		N/A	t.l.	-8891.09	75	68
pr152	984.54	13	10780	984.54	13	2
rat195		N/A	t.l.	-18301.7	39	972
d198		N/A	t.l.	-19183.9	31	3
korB200		N/A	t.l.	-138.68	111	7
gr202		N/A	t.l.	-19859.1	48	70
ts225	1296.11	0	8466	1296.11	0	14
gr229		N/A	t.l.	-20520.4	67	8
gil262		N/A	t.l.	-24784.5	96	27
a280		N/A	t.l.	-24933.6	60	67
lin318		N/A	t.l.	-1883.21	365	1906
fl417		N/A	t.l.	-41406.8	48	6
gr431		N/A	t.l.	-41751	123	823
pr439		N/A	t.l.	322.41	20	70
pcb442		N/A	t.l.	-43489.5	142	338
d493		N/A	t.l.	-48934.3	43	25
p654		N/A	t.l.	-65478.8	71	1068
d657		N/A	t.l.	-64671.9	125	1162
gr666		N/A	t.l.	-65273.1	169	54

Table 2: Computational results for the heuristic Primitive Comb Inequalities separation.

	Heuristic GSEC + Primitive Comb				
Instance	Obj.	Extra Cuts	Time	% Gap	Improv.
eil13	3228.61	0	<1	0	0
ulysses22	-1964.93	5	<1	0	0
bayg29	-44.35	0	1	0	0
eil33	-2924.76	0	1	0	0
att48	1358.75	0	1	0	0
hk48	551.65	0	1	2.01	0
eil51	-4451.36	10	1	0.03	3.24
st70	-6431.62	6	2	0	1.61
eilB76	-7441.27	4	4	0	1.04
eilC76	-6900.17	6	1	0.03	0.19
eilD76	-6773.72	7	7	0	1.02
pr76	5568.42	0	1	0	0
gr96	-9529.7	36	5	0.02	2.11
rat99	-9260.32	4	21	0.04	0.55
rd100	-10119.1	20	3	0	0
eil101	-8988.02	4	6	0.03	0.49
eilA101	-9160.59	2	2	0.01	0.27
eilB101	-9710.34	2	7	0	0.27
lin105	-87.93	0	5565		0
pr107	339.84	0	<1	0	0
gr120	-10451.1	2	185	0	0
gr137	-13068.3	10	31	0.01	2.2
pr144	532.69	0	<1		0
ch150	-8874.46	8	85	0.53	16.63
pr152	984.54	0	2		0
rat195	-18296.1	119	6567	0.09	5.6
d198	-19183.9	5	6	0	0
korB200	-138.68	0	8		0
gr202	-19858.2	38	281	0	0.9
ts225	1296.11	0	20	0	0
gr229	-20516.7	16	13	0.02	3.7
gil262	-24777	32	49	0.04	7.5
a280	-24921.1	49	122	0.02	12.5
lin318	-1842.91	94	4420		40.3
fl417	-41406.8	40	8		0
gr431	-41748.6	38	2041	0.01	2.4
pr439	322.41	0	108		0
pcb442	-43489.5	20	963		0
d493	-48934.3	39	38		0
p654	-65478.8	35	1317		0
d657	-64671.9	21	1482		0
gr666	-65266	80	113	0.03	7.1

It is natural to assume the heuristics would be most beneficial for large enough instances. As Tables 1 and 2 show, there is not much room for improvement for small instances as usually even the linear programming formulation can come up with an integral solutions without the aid of GSEC or primitive comb inequality constraints.

As soon as the size of the instance goes beyond 50 points, the time spent on the LP separation shows a huge increase while the cost of heuristic separation is still relatively low. This is not so surprising as the LP separation has to solve an entire linear programming problem for each node of the graph. For small instances, the overhead caused by a few extra problems is negligible, but for larger instances it can be prohibitive. Also, for the instances for which an integral solution is available, the gap between the two heuristic solutions combined and the optimal solution is very small. In most cases the gap is very close to 0 and only for one instance it goes as high as %2. In larger instances, both heuristics perform consistently well while the LP separation of GSECs and the ILP formulation do not return with a solution in the period of 4 hours.

A comparison between using GSEC heuristic separation alone and using both heuristics together reveals that although there are cases for which the solution quality can benefit significantly from running both heuristics instead of one, the GSEC separation seems to be promising on its own in many cases. The relatively long extra time spent on the separation of primitive combs results in the introduction of a few extra cuts indeed. However, the improvement in the solution quality is not very significant. This is perhaps to be

expected since we run the primitive comb separation after all the GSEC cuts are introduced. As a result, the LP has already closed the gap between the fractional and the optimal integral objective value, and therefore, improvements at this point come in very small portions. We conjecture that by using different sequences of running the heuristics, one can expect to speed up the running time and get the best of the two heuristics.

7 CONCLUSIONS

The Prize Collecting Travelling Salesman Problem (PCTSP) is an important generalization of the famous Travelling Salesman Problem. It also arises as a sub problem in many variants of the Vehicle Routing Problem. In this paper, we provide efficient methods to solve the linear programming relaxation of the PCTSP. We provide efficient heuristics to obtain the Generalized Subtour Elimination Constraints (GSECs) for the PCTSP, and compare its performance with linear programming, which can be used to solve the separation problem for GSECs for the PCTSP optimally. In this paper, we also show that the connected component heuristic of Padberg and Hong can be applied to separate the primitive comb inequalities introduced by Balas for the PCTSP. We evaluate the effectiveness of introducing these inequalities for reducing the integrality gap for the PCTSP. Through empirical analysis, we have been able to verify the importance of two separation heuristics in finding near optimal solutions to the Prize Collecting TSP in a timely manner. As the instance sizes grow, the two heuristics show great promise by finding a significant number of violated inequalities.

In this paper, we have looked at two basic heuristic. As a possible future work, one can continue this line of work by adapting/designing other heuristics for broader classes of cuts. One extension can be incorporating more general comb inequalities. Furthermore, the order of introducing the cuts can have a great impact on the running time. One can study various ways of mixing different separation procedures to find the one that is most suitable for specific types of instances.

REFERENCES

Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, 19(4):621–636.

Balas, E. (1995). The prize collecting traveling salesman problem ii: Polyhedral results. *Networks*, 25(4):199–216.

Bérubé, J.-F., Gendreau, M., and Potvin, J.-Y. (2009). A branch-and-cut algorithm for the undirected prize collecting traveling salesman problem. *Networks*, 54(1):56–67.

Cappanera, P., Gouveia, L., and Scutellà, M. G. (2011). The skill vehicle routing problem. In *Proceedings of the 5th International Conference on Network Optimization, 2011*, number LNCS 6701, pages 354–364. Springer-Verlag.

Chaves, A. A. and Lorena, L. A. N. (2008). Hybrid metaheuristic for the prize collecting travelling salesman problem. In *Proceedings of the EvoCOP 2008*, number LNCS 4972, pages 123–134. Springer-Verlag.

Crowder, H. and Padberg, M. W. (1980). Solving large-scale symmetric travelling salesman problems to optimality. *Management Science*, 26:495–509.

Fischetti, M., Salazar, J. J., and Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized travelling salesman problem. *Operations Research*, 45(3):378–393.

Fischetti, M. and Toth, P. (1988). An additive approach for the optimal solution of the prize-collecting travelling salesman problem. In Golden, B. L. and Assad, A. A., editors, *Vehicle routing: Methods and studies*, pages 319–343. Elsevier Science Publishers.

Goemans, M. X. (1994). The steiner polytope and related polyhedra. *Mathematical Programming*, 63:157–182.

Gutin, G. and Punnen, A. P., editors (2002). *The Traveling Salesman Problem and its Variations*. Combinatorial optimization. Kluwer Academic, Dordrecht, London.

Hong, S. (1972). *A Linear Programming Approach for the Traveling Salesman Problem*. PhD thesis, John Hopkins University, Baltimore, Maryland, USA.

Land, A. (1979). The solution of some 100-city travelling salesman problems. Technical report, London School of Economics, London, UK.

Padberg, M. W. and Hong, S. (1980). On the symmetric traveling salesman problem: a computational study. *Mathematical Programming Study*, 12:778–107.

Reinelt, G. (1991). Tsplib: A traveling salesman problem library. *ORSA J. Comput.*, pages 376–384.

Wolsey, L. A. (1998). *Integer programming*. Wiley interscience series in discrete mathematics and optimization. Wiley.