European Journal of Operational Research 253 (2016) 280–289

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Exact and heuristic algorithms for the Hamiltonian *p*-median problem

Güneş Erdoğan[a,*], Gilbert Laporte[b], Antonio M. Rodríguez Chía[c]

[a] School of Management, University of Bath, Claverton Down, Bath, BA2 7AY, United Kingdom
[b] Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montreal, H3T 2A7, Canada
[c] Departamento Estadística e Investigación Operativa, University of Cádiz, Pol. Río San Pedro, 11510 Puerto Real, Cádiz, Spain

ARTICLE INFO

Article history:
Received 29 April 2015
Accepted 6 February 2016
Available online 12 February 2016

Keywords:
Hamiltonian
p-median
Branch-and-cut
Metaheuristic

ABSTRACT

This paper presents an exact algorithm, a constructive heuristic algorithm, and a metaheuristic for the *Hamiltonian p-Median Problem* (HpMP). The exact algorithm is a branch-and-cut algorithm based on an enhanced *p*-median based formulation, which is proved to dominate an existing *p*-median based formulation. The constructive heuristic is a giant tour heuristic, based on a dynamic programming formulation to optimally split a given sequence of vertices into cycles. The metaheuristic is an iterated local search algorithm using 2-exchange and 1-opt operators. Computational results show that the branch-and-cut algorithm outperforms the existing exact solution methods.

1. Introduction

This paper studies the *Hamiltonian p-Median Problem* (HpMP), defined on a complete undirected graph $G = (V, E)$, where V is the vertex set, and $E = \{(i, j) : i, j \in V, i < j\}$ is the edge set. There is a cost c_{ij} associated with every edge (i, j). The aim is to partition the graph into p subsets of vertices, each connected by a single cycle. The objective is to minimize the total cost of edges belonging to the cycles. Following the convention of Gollowitzer, Gouveia, Laporte, Pereira, and Wojciechowski (2014), we only consider subsets (cycles) of cardinality 3 or more. The *Traveling Salesman Problem* (TSP) is a special case of the HpMP with $p = 1$, and consequently the HpMP is NP-hard. It is worth mentioning that the 2-matching problem, which returns an arbitrary number of cycles is solvable in polynomial time (see, for example Miller & Pekny, 1995).

Laporte, Nobert, and Pelletier (1983) introduced a series of location-routing problems and provided computational results for exact algorithms using cutting planes. One of these problems was to locate no more than p non-intersecting cycles in a graph with minimum cost, which was the precursor of the HpMP. The HpMP was introduced by Branco and Coelho (1990). It has received relatively little attention from researchers, and the existing studies have mostly focused on exact algorithms. Motivated by an application in laser multi-scanners, (Glaab & Pott, 2000) have studied the HpMP and presented a three-index formulation, together with results on the dimension of the associated polytope. Zohrehbandian (2007) has proposed a formulation for the HpMP based on a three-index *Vehicle Routing Problem* formulation, but did not provide any computational results. Gollowitzer, Pereira, and Wojciechowski (2011) have provided three formulations for the HpMP together with valid inequalities and branch-and-cut algorithms. In a later study, (Gollowitzer et al., 2014) have introduced seven different formulations for the HpMP which they have compared in terms of dominance relationships. They have also presented computational results for up to $|V| = 100$. Hupp and Liers (2013) have conducted a polyhedral analysis of an HpMP formulation using only edge variables and proved that a subset of the well-known 2-matching inequalities from the TSP define facets of the HpMP polytope.

There exist very few studies on heuristics for the HpMP and its variants. Glaab (2002) provided fast heuristics and improved lower bounds for a variant of the HpMP that arises in cutting problems. Uster and Kumar (2010) have studied the *Balanced Ring Problem*, which is another variant of the HpMP requiring the number of vertices on each cycle to be almost equal. They have provided a GRASP-based constructive algorithm as well as a local search heuristic. To the best of our knowledge, no metaheuristics have yet been proposed for the HpMP.

The remainder of this paper is organized as follows. In Section 2, we recall a integer linear programming formulation for the HpMP proposed by Gollowitzer et al. (2014); we also introduce an alternative formulation with several reinforcements and we develop a branch-and-cut algorithm based on this formulation. In Section 3.1, we provide a giant tour heuristic based on a Dynamic Programming formulation. In Section 3.2, we provide an

* Corresponding author. Tel.: +447428608370.
E-mail addresses: G.Erdogan@bath.ac.uk (G. Erdoğan), gilbert.laporte@cirrelt.ca (G. Laporte), antonio.rodriguezchia@uca.es (A.M. Rodríguez Chía).

http://dx.doi.org/10.1016/j.ejor.2016.02.012
0377-2217/

Iterated Local Search (ILS) algorithm for the HpMP. In Section 4, we present the computational results for our algorithms on benchmark instances. Conclusions follow in Section 5.

2. Enhanced *p*-median based formulation

Gollowitzer et al. (2014) have proposed a p-median based formulation, which they call Model 3. It uses variables for assigning vertices to other vertices. For the sake of completeness, we present their formulation below, which we call HpMP1. The authors denote the ordered vertex pairs of every edge $(i, j) \in E$ as $\gamma(i, j) = \{(i, j), (j, i)\}$, and the edges between a subset of vertices $W \subset V$ and the remaining vertices as $\delta(W)$. Let x_{ij} be equal to 1 if edge (i, j) belongs to the solution, and 0 otherwise. Let y_i be equal to 1 if it is selected as a depot, and 0 otherwise. Finally, let v_{ij} be equal to 1 if vertex i is assigned to depot j, and 0 otherwise. The formulation is then:

(HpMP1)

$$\text{minimize} \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{obj}$$

subject to

$$\sum_{i\in V} y_i = p \tag{pm1}$$

$$\sum_{j\in V\setminus\{i\}} v_{ij} + y_i = 1 \quad i \in V \tag{pm2}$$

$$v_{ij} \le y_j \quad i, j \in V : i \ne j \tag{pm3}$$

$$\sum_{j\in\delta(i)} x_{ij} = 2 \quad i \in V \tag{deg}$$

$$\sum_{(i,j)\in\delta(W)} x_{ij} \ge 2\sum_{l\in V\setminus W} v_{kl} \quad W \subset V, k \in W \tag{pm $\le$}$$

$$v_{ka} + x_{ij} \le 1 + v_{la} \quad (i, j) \in E, (k, l) \in \gamma(i, j), a \in V \setminus \{k, l\} \tag{pm $\ge$}$$

$$y_k + x_{ij} \le 1 + v_{lk} \quad (i, j) \in E, (k, l) \in \gamma(i, j) \tag{pm $\ge'$}$$

$$v_{ij} = 0 \quad i, j \in V : i > j \tag{sb}$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E \tag{bin}$$

$$v_{ij} \in \{0, 1\} \quad i, j \in V : i \ne j \tag{pm4}$$

$$y_i \in \{0, 1\} \quad i \in V. \tag{pm5}$$

The objective function (obj) minimizes the total cost of cycles. Constraint set (pm1) sets the number of cycles to p. Constraint sets (pm2) and (sb) require every vertex to be assigned to itself or to a vertex having a higher index. Constraint set (pm3) forces a vertex to be chosen as a depot if another vertex is assigned to it. Constraint set (deg) states that every vertex must have a degree of 2 which, in conjunction with (bin), enforces the minimum cycle size to be 3. Constraints (pm $\le$) connect the vertices assigned to the same cycle by forcing two edges between the two complementary subsets if a vertex in one subset is assigned to a vertex in the other subset. Constraints (pm $\ge$) and (pm $\ge'$) eliminate connections between vertices that have been assigned to different depots. Constraints (sb) cut off symmetrical solutions by forcing all vertices in a cycle to be assigned to the vertex with the highest index. Finally, (bin), (pm4), and (pm5) are the integrality constraints on the variables.

2.1. Valid inequalities

To facilitate our analysis, we propose an alternative formulation, called HpMP2, for the HpMP. It is obtained by unifying the variables v_{ij} and y_j into the variable w_{ij}, i.e. w_{ij} is a binary variable equal to 1 if and only if vertex i is assigned to vertex j, with $w_{ii} = 1$ meaning that vertex i has been chosen as a depot. This transformation results in a simpler presentation of (pm $\ge$) and (pm $\ge'$), and the new sets of inequalities we subsequently propose. For the sake of clarity, we present the resulting formulation in its entirety, including constraints that are not affected by the change of variables:

(HpMP2)

$$\text{minimize} \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{i\in V} w_{ii} = p \tag{2}$$

$$\sum_{j\in V} w_{ij} = 1 \quad i \in V \tag{3}$$

$$w_{ij} \le w_{jj} \quad i, j \in V \tag{4}$$

$$\sum_{j\in\delta(i)} x_{ij} = 2 \quad i \in V \tag{5}$$

$$\sum_{(i,j)\in\delta(W)} x_{ij} \ge 2\sum_{l\in V\setminus W} w_{kl} \quad W \subset V, k \in W \tag{6}$$

$$w_{ka} + x_{ij} \le 1 + w_{la} \quad (i, j) \in E, (k, l) \in \gamma(i, j)\ a \in V \tag{7}$$

$$w_{ij} = 0 \quad i, j \in V : i > j \tag{8}$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E \tag{9}$$

$$w_{ij} \in \{0, 1\} \quad i, j \in V. \tag{10}$$

Note that the transformation unifies constraints (pm $\ge$) and (pm $\ge'$) into (7). We now state our first result.

Proposition 1. *The following inequalities are valid for HpMP2, and dominate (7):*

$$\sum_{k\in S} w_{ik} + x_{ij} \le 1 + \sum_{k\in S} w_{jk} \quad (i, j) \in E, S \subset V. \tag{11}$$

Proof. Since $\sum_{k\in S} w_{ik} \le 1$ and $x_{ij} \le 1$, this inequality is valid whenever $\sum_{k\in S} w_{ik} = 0$ or $x_{ij} = 0$. Thus, we only have to analyze the case where $\sum_{k\in S} w_{ik} = 1$ and $x_{ij} = 1$. In this case, $x_{ij} = 1$ implies that i and j are assigned to the same depot and hence $\sum_{k\in S} w_{jk} = 1$. Therefore, the inequality (11) is valid. Note that (7) is a special case of (11) if $|S| = 1$ or $|S| = |V| - 1$, and is therefore dominated by (11). □

Although the number of constraints (11) is exponential, these can be separated in polynomial time. For any given edge $(i, j) \in E$, start with $S = \emptyset$ and include a vertex $k \in V$ into S if and only if $w_{ik} > w_{jk}$. This results in an overall complexity of $O(|V|^3)$.

We now focus on (6). Define $\mathcal{F}(W)$ as the set of all sets of pairs (i, j): $i \in W, j \in V\setminus W$ or $i \in V\setminus W, j \in W$ such that for every element of $\mathcal{F}(W)$ there is at most one pair containing any vertex $k \in V$ as its second component. We now state our second result.

Proposition 2. *The following inequalities are valid for HpMP2, and dominate (6):*

$$\sum_{(i,j)\in\delta(W)} x_{ij} \ge 2\sum_{(k,l)\in F} w_{kl} \quad W \subset V, F \in \mathcal{F}(W). \tag{12}$$

Proof. Consider a partition of $\{W, V\setminus W\}$ of V, as depicted in Fig. 1, where the positive x variables are denoted with thin lines, the positive w variables are denoted with arrows, and the partition is denoted by a dashed line. In order to check that constraints (12) are valid, we will prove that a feasible solution of HpMP2, $(\bar{x}, \bar{w})$, satisfies them. Let $F \in \mathcal{F}(W)$. If for a pair $(k, l) \in F$ we have that $\bar{w}_{kl} = 1$, then either vertex $k \in W$ is assigned to depot $l \in V\setminus W$

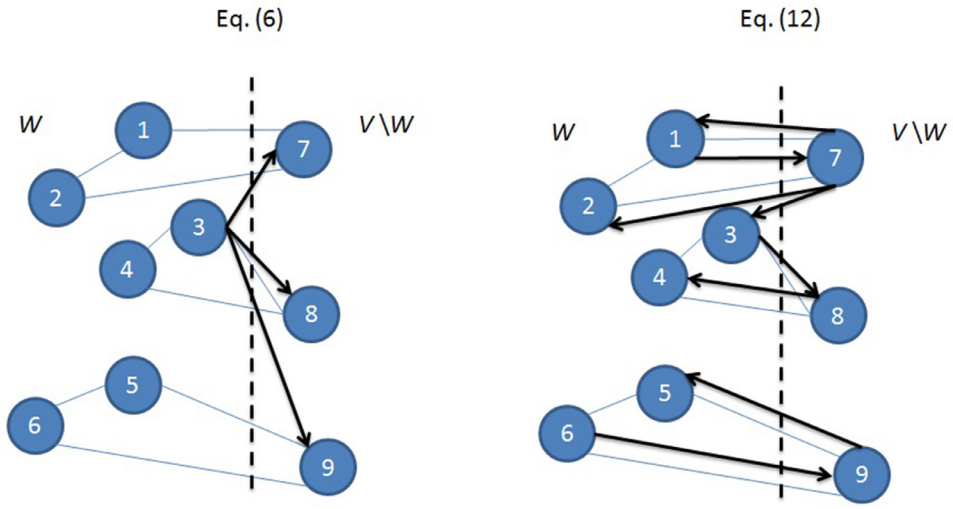

Fig. 1. Visualization of the assignment sets for constraint sets (6) and (12).

or vertex $k \in V\backslash W$ is assigned to depot $l \in W$. In both cases the cycle represented by vertex $l \in V$ has vertices in W and in $V\backslash W$. Thus, there are at least two edges of that cycle with end vertices in W and $V\backslash W$, i.e. there exist vertices $i_1, i_2 \in W$ and $j_1, j_2 \in V\backslash W$, such that, $\bar{x}_{i_1,j_1} = \bar{x}_{i_2,j_2} = 1$ and $\bar{w}_{i_1,l} = \bar{w}_{i_2,l} = \bar{w}_{j_1,l} = \bar{w}_{j_2,l} = 1$. Note that it is possible to have $i_1 = i_2$ or $j_1 = j_2$, but not both. Moreover, by construction of F, no two pairs (k, l) and (k', l) belong to F, i.e., each variable w equal to 1 in the right-hand side of constraints (12) represents a different cycle. Hence, for a given $F \in \mathcal{F}(W)$, the summation in the right-hand side of (12) gives a number of different cycles crossing $\delta(W)$. Since the left-hand side counts the number of edges crossing $\delta(W)$, the left-hand side is at least twice the right-hand side (the number of different cycles crossing W represented by a vertex that is the second component of a pair of F). Hence, constraints (12) are valid. Finally, (6) is a special case of (12) when F is restricted to those subsets of $\mathcal{F}(W)$ where the pairs have an identical component of W and the second component belongs to $V\backslash W$. □

Despite our best efforts, we were neither able to find a polynomial time algorithm for the exact separation of this set of valid inequalities for the fractional solutions, nor to prove the NP-hardness of the separation problem. The closest problem in the literature is the separation of the cutset inequalities for the network loading problem (Barahona, 1996), which is NP-hard. However, the constraint imposing a maximum of one inflow arc per vertex and the unit capacities change the problem structure. We conjecture that the separation problem for fractional solutions is NP-hard. We have implemented a heuristic separation algorithm for the fractional solutions and applied it only at the root node of the branch-and-cut tree, due to the its high computational cost. The separation algorithm starts with a random $W \subset V$ and greedily adds and removes vertices to maximize the violation, until no further improvement is found. To find a maximally violated inequality, it is sufficient to search over the space of the vertex subset W, since a matching F can be constructed optimally by a greedy algorithm that selects one w_{kl} variable for each l. Let us denote the violation of an inequality defined by the set W as $f(W)$. We now present the pseudocode for the separation algorithm, which we call Algorithm 1.

For the solutions that satisfy integrality constraints, we have executed a depth-first search from each vertex to the depot it is assigned, and added members of (12) to ensure connectivity between the two vertices when necessary. This procedure separates all violated members of (12) and has a complexity of $O(|V|^3)$. We now provide the pseudocode of the separation algorithm, which we call Algorithm 2.

The final set of valid inequalities we present is based on the observation that the fact that every cycle is composed of at least three vertices. We state the result without proof.

Proposition 3. *The following inequalities are valid for HpMP2:*

$$\sum_{j \in V: j \neq i} w_{ji} \geq 2w_{ii} \quad i \in V. \tag{13}$$

2.2. Branch-and-cut algorithm

We now present our branch-and-cut algorithm for HpMP2, which we call Algorithm 3. Denote the ith subproblem as s_i, the solution of the subproblem as (x_i, w_i), and the best known solution as (x^*, w^*). Furthermore, denote the objective function value of a solution as $z(x, w)$.

3. Heuristic algorithms

In this section, we provide a giant tour heuristic and an ILS algorithm for the HpMP.

3.1. Giant tour heuristic

Beasley (1983) proposed a giant tour heuristic that optimally splits a TSP tour to construct a solution for the *Capacitated Vehicle Routing Problem*. The splitting algorithm was later used as a local search operator within a genetic algorithm by Prins (2004), and more recently within an ILS algorithm by Afsar, Prins, and Santos (2014). Notably, Love (1976) studied the problem of locating a number of facilities in the continuous search space on a line and proposed a dynamic programming formulation, which bears some resemblance to our dynamic programming formulation in terms of stage and state definitions. However, the cost of a solution for this problem is computed based on the distances of the existing facilities and the new facilities rather on than the sequential order of locations. We now present our splitting algorithm for the HpMP.

Consider a sequence of the vertex indices $\sigma = (\sigma_1, \ldots, \sigma_n)$, which corresponds to a TSP solution on the vertices. We propose a dynamic programming algorithm to optimally partition this sequence into p cycles of the form $(\sigma_i, \sigma_{i+1}, \ldots, \sigma_j, \sigma_i)$, of cost $\hat{c}_{ij}$. Define $g_k(i)$ as the sum of the optimal cost of the kth, …, pth cycles if the kth cycle starts at the ith element of the sequence of

Algorithm 1

$W = \emptyset$
for $i = 1$ **to** $|V|$
  Generate a uniform random number $u = U[0, 1]$.
  If $u \leq 0.5$ **then**
  $W = W \cup \{i\}$
  end if
end for

do
  $W^* = W$.

  for $i = 1$ **to** $|V|$
    if $i \in W$ **then**
      if $f(W \setminus \{i\}) > f(W^*)$ **then**
        $W^* = W \setminus \{i\}$
      end if
    else
      if $f(W \cup \{i\}) > f(W^*)$ **then**
        $W^* = W \cup \{i\}$
      end if
    end if
  end for

  for $i = 1$ **to** $|V|$
    for $j = i + 1$ **to** $|V|$
      if $i \in W$ and $j \notin W$ **then**
        if $f((W \setminus \{i\}) \cup \{j\}) > f(W^*)$ **then**
          $W^* = (W \setminus \{i\}) \cup \{j\}$
        end if
      end if
      if $i \notin W$ and $j \in W$ **then**
        if $f((W \cup \{i\}) \setminus \{j\}) > f(W^*)$ **then**
          $W^* = (W \cup \{i\}) \setminus \{j\}$
        end if
      end if
    end for
  end for
while $W^* \neq W$

if $f(W^*) > 0$ **then**
  $F^* = \emptyset$
  for $i = 1$ **to** $|V|$
    if $i \in W^*$ **then**
      $j = \text{argmax}_{k \in V \setminus W^*} w^*_{ki}$
    else
      $j = \text{argmax}_{k \in W^*} w^*_{ki}$
    end if
    $F^* = F^* \cup \{(j, i)\}$
  end for
  Add the member of (12) with W^* and F^*
end if
return

TSP solution. The dynamic programming formulation is then:

$$g_k(i) = \begin{cases} \min\{\hat{c}_{ij} + g_{k+1}(j+1) : & \text{if } k < p, 3k - 2 \leq i \\ \quad j \in \{i+2, \ldots, n - 3(p-k)\}\} & \quad \leq n - 3(p-k) - 2 \\ \hat{c}_{in} & \text{if } k = p, 3k - 2 \\ & \quad \leq i \leq n - 2 \\ \infty & \text{otherwise.} \end{cases} \tag{14}$$

Algorithm 2

Construct a directed graph $G' = (V, A)$
  where $(i, j) \in A \iff ((i, j) \in E)$ or $(j, i) \in E))$ and $x^*_{ij} = 1$
for $i = 1$ **to** $|V|$
  $j = \text{argmax}_{k \in V} w^*_{ik}$
  if $i \neq j$ **then**
    Execute a depth-first search on G' starting from i
    if j is not reachable **then**
      Construct W^* as the set of vertices reachable from i
      $F^* = \emptyset$
      for $i = 1$ **to** $|V|$
        if $i \in W^*$ **then**
          $j = \text{argmax}_{k \in V \setminus W^*} w^*_{ki}$
        else
          $j = \text{argmax}_{k \in W^*} w^*_{ki}$
        end if
        $F^* = F^* \cup \{(j, i)\}$
      end for
      Add the member of (12) with W^* and F^*
    end if
  end if
end for

return $(W^*, f(W^*))$

Algorithm 3

Construct the root node subproblem s_0 by adding all members of the (13) to HpMP2.
Do
  Solve the LP relaxation of s_0.
  Separate and add all violated members of the valid inequality set (11) to s_0.
  Heuristically separate and add violated members of the valid inequality set (12) to s_0.
While At least one valid inequality is added to s_0
Initialize the branch-and-cut node set $\mathcal{S} = \{s_0\}$.
While $\mathcal{S} \neq \emptyset$
  Pick $s_i \in \mathcal{S}$ and set $\mathcal{S} := \mathcal{S} \setminus s_i$.
  Do
    Solve s_i.
    Separate and add all violated members of the valid inequality set (11) to s_i.
    If $(x_i, w_i) \in \{0, 1\}^{|E|+|V|^2}$ **then**
      Separate and add violated members of the valid inequality set (12) to s_i.
    End If
  While At least one valid inequality is added to s_i **and** $z(x_i, w_i) < z(x^*, w^*)$

  If $z(x_i, w_i) > z(x^*, w^*)$ **then**
    Discard s_i.
  Else If $(x_i, w_i) \in \{0, 1\}^{|E|+|V|^2}$ **then**
    $(x^*, w^*) = (x_i, w_i)$.
  Else
    Create s_{i+1} and s_{i+2} by branching on a binary variable with a fractional value
    and set $\mathcal{S} := \mathcal{S} \cup \{s_{i+1}\} \cup \{s_{i+2}\}$.
  End If
End While

return (x^*, w^*)

Algorithm 4 (sequence σ).

// compute $\hat{c}_{ij}$
for $i = 1$ **to** n
  for $j = i+1$ **to** n
    if $j = i+1$ **then**
      $\hat{c}_{ij} = c_{\sigma(i)\sigma(j)}$ // cost of an edge
    else
      $\hat{c}_{ij} = \hat{c}_{i,j-1} + c_{\sigma(j-1)\sigma(j)}$ // cumulative cost
    end if
  end for
  for $j = i+1$ **to** n
    $\hat{c}_{ij} = \hat{c}_{ij} + c_{\sigma(j)\sigma(i)}$ // cost of completing the cycle
  end for
end for

// dynamic programming algorithm
for $k = p$ **down to** 1 // backward recursion
  for $i = 1$ **to** n
    $u_k(i) = 0$ // initializing the control
    if $(i < 3k-2)$ or $(i > n-3(p-k)-2)$ **then** // a cycle cannot fit here
      $g_k(i) = \infty$
    else if $k = p$ **then**
      $g_k(i) = \hat{c}_{in}$
      $u_k(i) = n$ // the cycle finishes at the end of the sequence
    else
      $g_k(i) = \infty$
      for $j = i+2$ to $n-3(n-k)$ // searching for the best element to finish the cycle
        if $g_k(i) > \hat{c}_{ij} + g_{k+1}(j+1)$ **then**
          $g_k(i) = \hat{c}_{ij} + g_{k+1}(j+1)$
          $u_k(i) = j$
        end for
    end if
  end for
end for

// determine the value of x

Initialize $x = 0$

for $k = 1$ **to** p
  for $j = i$ **to** $u_k(i) - 1$
    if $\sigma_j < \sigma_{j+1}$ **then** $x_{\sigma_j,\sigma_{j+1}} = 1$ // ensuring that we select the correct edge index
    else $x_{\sigma_{j+1},\sigma_j} = 1$
  end for
  if $\sigma_{u_k(i)} < \sigma_i$ **then** $x_{\sigma_{u_k(i)},\sigma_i} = 1$ // ensuring that we select the correct edge index for return
  else $x_{\sigma_i,\sigma_{u_k(i)}} = 1$

  $i = u_k(i) + 1$
end for

return x

An optimal partitioning can be determined by computing $g_1(1)$. We now provide an algorithm (Algorithm 4) that computes $\hat{c}_{ij}$ as well as $g_k(i)$. Denote by $u_k(i)$ the optimal endpoint of partition k starting at the ith element of the TSP sequence.

The first part of Algorithm 4 computes $\hat{c}_{ij}$ in $O(|V|^2)$ time. The space complexity of the dynamic programming algorithm is $O(|V|^2)$ and its time complexity is $O(|V|^3)$. We first compute a TSP solution on G, from which we extract the sequence of the vertices. Note that the dynamic programming formulation does not allow for cycles of the type $(\sigma_i, \sigma_{i+1}, \ldots, \sigma_n, \sigma_1, \ldots, \sigma_{i-1})$. To make up for this shortcoming, the giant tour heuristic executes the dynamic programming algorithm n times, changing the sequence by relocating the last element to the first position and shifting the remaining elements to the next position. We now present the detailed pseudocode of the heuristic, called Algorithm 5, where $z_{TSP}(\sigma)$ denotes the cost of the TSP tour defined by the sequence σ.

An example run of Algorithm 5 is depicted in Fig. 2, for an instance of the HpMP with $n = 9$ and $p = 2$, where the vertices are reindexed in the order in which they appear in the first sequence for the sake of simplicity. The best partitioning of each sequence into two cycles is denoted by the rectangles enclosing parts of the sequence. The best solution among all these partitions (highlighted) is chosen as the result of Algorithm 5. Although this brings the overall complexity up to $O(|V|^4)$, the CPU time requirement is still negligible since it is executed only once.

3.2. Iterated local search algorithm

We now present the details of our ILS algorithm, which is based on perturbing the best known solution through random swaps and relocations of vertices among the cycles, and reoptimizing through

Algorithm 5

for $i = 1$ **to** n
  $\sigma_i^* = i$
end for do
  $\sigma = \sigma^*$.

  for $i = 1$ **to** $|V|$ // 2-exchange
    for $j = 1$ **to** $|V|$
      Construct σ' by relocating the i^{th} element of σ to the j^{th} position
      if $z_{TSP}(\sigma') < z_{TSP}(\sigma^*)$ **then**
        $\sigma^* = \sigma'$
      end if
    end for
  end for

  for $i = 1$ **to** $|V|$ // 1-opt
    for $j = i+1$ **to** $|V|$
      Construct σ' by swapping the i^{th} and j^{th} elements of σ
      if $z_{TSP}(\sigma') < z_{TSP}(\sigma^*)$ **then**
        $\sigma^* = \sigma'$
      end if
    end for
  end for
while $\sigma \neq \sigma^*$

Determine x^* using σ as an input to Algorithm 4
for $i = 1$ **to** $n-1$
  $k = \sigma_n$
  for $j = 2$ **to** n
    $\sigma_j = \sigma_{j-1}$
  end for
  $\sigma_1 = k$
  Determine x using σ as an input to Algorithm 4
  if $z(x) < z(x^*)$ **then** $x^* = x$
end for

return x^*

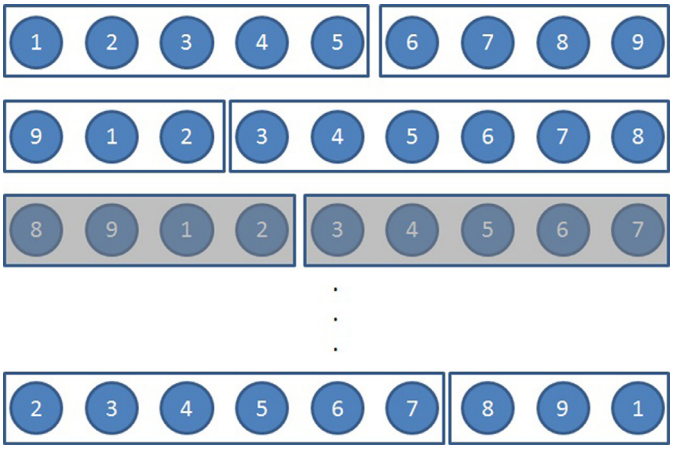

Fig. 2. Visualization of a run of Algorithm 5.

Table 1
Computational results for the giant tour heuristic and the ILS metaheuristic.

$\|V\|$	p^*	p	Giant tour optimality gap (percent)	ILS optimality gap (percent)	ILS CPU time (seconds)
20	3.80	2	6.56	0.00	0.17
20	3.80	3	5.45	0.00	0.14
20	3.80	4	3.10	0.00	0.16
20	3.80	5	5.06	0.00	0.13
20	3.80	6	7.76	0.00	0.10
40	7.40	2	10.78	0.00	1.52
40	7.40	5	12.65	0.86	1.21
40	7.40	8	10.86	0.57	1.20
40	7.40	11	14.24	0.00	1.03
40	7.40	13	16.76	0.00	0.58
60	10.40	2	18.15	0.66	5.63
60	10.40	7	19.07	0.99	4.38
60	10.40	12	19.24	0.46	4.28
60	10.40	17	22.24	0.04	3.49
60	10.40	20	37.77	0.00	1.74
80	12.60	2	22.75	2.33	15.10
80	12.60	8	23.77	2.92	13.46
80	12.60	14	23.40	2.02	10.84
80	12.60	20	25.06	0.07	10.20
80	12.60	26	36.55	0.01	5.28
100	12.20	2	28.07	1.12	34.92
100	12.20	10	26.79	1.84	23.26
100	12.20	18	26.39	1.52	20.81
100	12.20	26	27.53	0.73	20.48
100	12.20	33	42.32	1.02	10.43
Average			19.69	0.69	

local search. The random feature of the ILS algorithm acts as a diversification mechanism, which reduces the likelihood that the search process will become trapped into a local minimum. Let us denote the solution as p sequences of vertices, each sequence corresponding to the vertices within the corresponding cycle, with $x_{[i][j]} \in \{1, \ldots, |V|\}$ denoting the jth element of the ith sequence. Let us also denote the number of vertices in cycle i as $y_{[i]}$. Finally, we write $U[a, \ldots, b]$ to denote a discrete uniform random variable that returns values in the range $\{a, \ldots, b\}$.

The CPU time requirement is determined by the choice of the parameter k_{max}, the total number of iterations. At each iteration, a copy of the best known solution is created, and perturbed through randomly swapping α pairs of vertices from two cycles as well as relocating β vertices from their cycle to a position in another randomly chosen cycle. The choices of α and β are important for the performance of the algorithm, since very low values do not provide enough diversification, and very large values destroy the structure of the best known solution and inhibit intensification. After the perturbation, the resulting solution is reoptimized using the well-known local search operators 1-opt and 2-exchange, by selecting the best move among the operators at every step. If the objective function value of the local search is better than that of the best known solution, the best known solution is replaced by the current solution.

Algorithm 6

// constructive phase
Determine x^* using Algorithm 5

// improvement phase
for $k = 1$ **to** $k_{\max}$
  $x = x^*$
  for $i = 1$ **to** α // perturbation by random 2-exchange
    Randomly select the first cycle $q = U[1, \ldots, p]$
    Randomly select a vertex in cycle q as $r = U[1, \ldots, y_{[q]}]$
    Randomly select the second cycle $s = U[1, \ldots, p]$
    Randomly select a vertex in cycle s as $t = U[1, \ldots, y_{[s]}]$
    Swap the values of $x_{[q][r]}$ and $x_{[s][t]}$
  end for
  for $i = 1$ **to** β // perturbation by random relocation
    Randomly select a cycle $q = U[1, \ldots, p]$
    Randomly select a position in cycle q as $r = U[1, \ldots, y_{[q]}]$
    Store the value $j = x_{[q][r]}$ of and update x by removing $x_{[q][r]}$ from cycle q
    Randomly select a cycle $q = U[1, \ldots, p]$
    Randomly select a position in cycle q as $r = U[1, \ldots, y_{[q]}]$
    Update x by inserting j in position r of cycle q
  end for
 do
  $x'' = x$
  for $q = 1$ **to** p // 2-exchange
    for $r = 1$ **to** $y_{[q]}$**then**
      for $s = q + 1$ **to** p
        for $t = 1$ **to** $y_{[s]}$
          $x' = x$
          Swap the values of $x'_{[q][r]}$ and $x'_{[s][t]}$
          if $z(x') < z(x'')$ **then**
            $x'' = x'$
          end if
        end for
      end for
    end for
  end for
  for $q = 1$ **to** p // 1-opt
    for $r = 1$ **to** $y_{[q]}$**then**
      $x' = x$
      Store the value $j = x'_{[q][r]}$ of and update x' by removing $x'_{[q][r]}$ from cycle q
      for $s = 1$ **to** p
        for $t = 1$ **to** $y_{[s]}$
          Update x' by inserting j in position t of cycle s
          if $z(x') < z(x'')$ **then**
            $x'' = x'$
          end if
        end for
      end for
    end for
  end for
 while $x'' \neq x$

end for

return x^*

4. Computational results

We have conducted our computational experiments on a Lenovo T440p laptop with an i7 2.50 gigahertz CPU and 8 gigabyte RAM. A CPU time limit of 1 hour was imposed on the branch-and-cut algorithm. ILS were also coded in C++ and executed on the same computer. We have used the instances of Gollowitzer et al. (2014), in order to have a benchmark to compare the computational performance of our model. We have also preformed computational experiments on three TSPLIB (Reinelt, 1991) instances:

Table 2
Comparison of the performance of the branch-and-cut algorithm for the HpMP2 with Model 1 and Model 3.

$\|V\|$	p	Model 1	Model 3	HpMP2	
		Solved	Solved	Solved	CPU time (seconds)
20	2	5/5	5/5	5/5	0.25
20	3	5/5	5/5	5/5	0.25
20	4	5/5	5/5	5/5	0.31
20	5	5/5	5/5	5/5	0.31
20	6	5/5	5/5	5/5	0.15
40	2	5/5	5/5	5/5	2.07
40	5	5/5	5/5	5/5	1.92
40	8	5/5	5/5	5/5	3.44
40	11	5/5	5/5	5/5	1.54
40	13	1/5	2/5	5/5	76.25
60	2	5/5	4/5	5/5	7.81
60	7	5/5	4/5	5/5	8.23
60	12	4/5	2/5	5/5	50.71
60	17	1/5	1/5	5/5	8.80
60	20	0/5	0/5	5/5	169.38
80	2	3/5	1/5	5/5	19.70
80	8	5/5	0/5	5/5	32.30
80	14	5/5	0/5	5/5	19.72
80	20	1/5	0/5	5/5	30.53
80	26	0/5	0/5	5/5	1585.62
100	2	4/5	0/5	5/5	45.10
100	10	5/5	0/5	5/5	50.07
100	18	2/5	0/5	5/5	226.63
100	26	0/5	0/5	5/5	961.86
100	33	0/5	0/5	2/5	2982.87
Total		86/125	59/125	122/125	

dantzig42, gr96, and u159. In what follows, we compute the optimality gap as the ratio of the difference between the best solution value and the best lower bound found by the branch-and-cut algorithm to the best lower bound found by the branch-and-cut algorithm.

Based on a number of initial experiments, we have chosen the values of $\alpha = \lfloor |V|/5 \rfloor$ and $\beta = \lfloor |V|/5 \rfloor$. We have observed that the best solution found by the ILS algorithm did not change much after a few thousand iterations, hence we have used $k_{max} = 10,000$. The aggregate results for the heuristic algorithms are presented in Table 1. For each instance, we have solved a 2-matching problem to determine the optimal number of cycles, and we report the average value of this result in the column labeled "Average p^*" as a measure of how hard the instances are. We have omitted the details of the CPU time requirement of Algorithm 5, since Algorithm 5 did not require more than 0.01 CPU seconds for any of the instances. Algorithm 6 has an average optimality gap of 0.69 percent, and requires no more than a minute of CPU time. Based on these results and its simplicity, we conclude that Algorithm 6 has a satisfactory performance.

The aggregate results for HpMP2 are presented in Tables 2 and 3. The best found solution from Algorithm 6 was fed in as an initial incumbent to the branch-and-cut algorithm. The improved valid inequalities and the tight upper bound have contributed to the performance of our algorithm, resulting in finding the optimal solution of 122 out of the 125 benchmark instances. The CPU time requirement of the branch-and-cut algorithm is also provided in Table 2. Note that all the CPU times we report for the branch-and-cut algorithm include the CPU time for the ILS algorithm that was used for generating the initial solution. The columns Model 1 and Model 3 correspond to the performance of the best two models of Gollowitzer et al. (2014) as reported by the authors, the former being the formulation in the space of natural variables, and the latter being the p-median based formulation our formulation is based upon. Model 1 has successfully solved 86 instances out of 125, whereas Model 3 could only solve 59. We thereby conclude that HpMP2 performs better than both of these models.

The details of the initial optimality gaps computed at the root node of the branch-and-cut tree are presented in Table 3. We have analyzed the effect of separating only (11) and separating both (11) and (12). The results show that the difference between the

Table 3
Optimality gaps.

$\|V\|$	p	Model 1	Model 3	HpMP2		
		Initial gap (percent)	Initial gap (percent)	Initial gap (percent) with (11)	Initial gap (percent) with (11) and (12)	Final gap (percent)
20	2	1.85	1.03	0.33	0.18	0.00
20	3	3.00	1.89	1.24	1.09	0.00
20	4	2.72	1.67	0.73	0.76	0.00
20	5	3.14	2.51	0.47	0.44	0.00
20	6	3.24	1.60	0.00	0.00	0.00
40	2	3.07	0.98	0.46	0.44	0.00
40	5	2.73	1.93	2.05	1.99	0.00
40	8	2.90	2.36	1.53	1.48	0.00
40	11	3.01	2.00	0.08	0.12	0.00
40	13	10.68	7.32	5.06	5.20	0.00
60	2	4.12	1.33	2.30	2.30	0.00
60	7	1.53	0.91	1.73	1.73	0.00
60	12	1.94	1.46	0.89	0.95	0.00
60	17	3.84	2.25	0.59	0.59	0.00
60	20	20.27	16.67	4.60	4.20	0.00
80	2	2.79	0.77	3.20	3.20	0.00
80	8	1.48	0.88	3.55	3.55	0.00
80	14	1.08	0.67	2.26	2.26	0.00
80	20	3.36	2.37	0.50	0.46	0.00
80	26	20.40	17.47	2.72	2.49	0.00
100	2	2.68	2.04	2.00	2.00	0.00
100	10	1.44	1.35	2.30	2.30	0.00
100	18	2.77	2.12	2.03	2.13	0.00
100	26	9.89	8.27	1.50	1.45	0.00
100	33	35.61	32.20	2.91	2.74	1.02
Average		5.98	4.56	1.80	1.76	0.04

Table 4
Computational results for the TSPLIB instances.

Instance	p^*	p	Model 1			Model 3			HpMP2		
			Initial gap (percent)	Final gap (percent)	CPU time (seconds)	Initial gap (percent)	Final gap (percent)	CPU time (seconds)	Initial gap (percent)	Final gap (percent)	CPU time (seconds)
dantzig42	3	3	1.08	0.00	0.07	0.23	0	1.42	0.00	0.00	0.28
dantzig42	3	10	1.99	0.00	1.70	1.04	0.92	3600.00	0.00	0.00	0.35
gr96	9	5	1.25	0.00	58.28	N/A	N/A	3600.00	1.93	0.00	12.75
gr96	9	20	6.45	5.00	3600.00	N/A	N/A	3600.00	2.29	1.97	3619.94
u159	20	5	5.04	3.72	3600.00	N/A	N/A	3600.00	2.32	1.64	3703.49
u159	20	30	9.55	8.16	3600.00	N/A	N/A	3600.00	3.91	3.59	3706.86

Table 5
Computational results for the new TSPLIB instances, small size.

Instance	p^*	p	Giant tour result	ILS result	Initial bound	Final bound	Best solution	Initial gap (percent)	Final gap (percent)	CPU time (seconds)
gr21	1	2	2922.00	2773.00	2747.36	2773.00	2773.00	0.93	0.00	0.49
		3	2873.00	2774.00	2752.20	2774.00	2774.00	0.79	0.00	0.34
		4	2814.00	2757.00	2757.00	2757.00	2757.00	0.00	0.00	0.19
		5	3118.00	2832.00	2809.10	2832.00	2832.00	0.82	0.00	0.46
		7	3782.00	3043.00	3043.00	3043.00	3043.00	0.00	0.00	0.45
ulysses22	6	2	73.68	68.33	68.33	68.33	68.33	0.00	0.00	0.39
		3	70.04	66.43	65.99	66.43	66.43	0.67	0.00	0.38
		4	66.75	64.23	64.23	64.23	64.23	0.00	0.00	0.19
		5	65.55	63.08	63.08	63.08	63.08	0.00	0.00	0.16
		7	74.26	65.08	65.08	65.08	65.08	0.00	0.00	0.18
gr24	3	2	1314.00	1238.00	1238.00	1238.00	1238.00	0.00	0.00	0.31
		3	1322.00	1246.00	1227.00	1227.00	1227.00	1.55	0.00	0.25
		4	1292.00	1227.00	1224.50	1227.00	1227.00	0.20	0.00	0.27
		6	1415.00	1278.00	1247.42	1266.00	1266.00	2.45	0.00	0.51
		8	1580.00	1317.00	1317.00	1317.00	1317.00	0.00	0.00	0.24
fri26	7	2	930.00	911.00	911.00	911.00	911.00	0.00	0.00	0.41
		3	931.00	903.00	903.00	903.00	903.00	0.00	0.00	0.31
		5	948.00	893.00	893.00	893.00	893.00	0.00	0.00	0.44
		6	921.00	886.00	886.00	886.00	886.00	0.00	0.00	0.37
		8	885.00	885.00	885.00	885.00	885.00	0.00	0.00	0.21
bayg29	3	2	1711.00	1562.00	1562.00	1562.00	1562.00	0.00	0.00	0.56
		4	1717.00	1549.00	1549.00	1549.00	1549.00	0.00	0.00	0.50
		5	1716.00	1555.00	1555.00	1555.00	1555.00	0.00	0.00	0.53
		7	1741.00	1618.00	1590.85	1618.00	1618.00	1.71	0.00	2.15
		9	1759.00	1676.00	1660.44	1676.00	1676.00	0.94	0.00	1.73
swiss42	7	4	1448.00	1232.00	1232.00	1232.00	1232.00	0.00	0.00	1.37
		6	1454.00	1231.00	1227.50	1231.00	1231.00	0.29	0.00	1.70
		8	1480.00	1231.00	1231.00	1231.00	1231.00	0.00	0.00	1.56
		10	1483.00	1238.00	1236.61	1238.00	1238.00	0.11	0.00	2.02
		14	1872.00	1292.00	1292.00	1292.00	1292.00	0.00	0.00	1.12
att48	14	4	38974.46	31903.30	31903.30	31903.30	31903.30	0.00	0.00	3.73
		6	39288.35	31896.89	31836.12	31836.12	31836.12	0.19	0.00	3.41
		9	38861.55	32215.33	32181.18	32195.53	32195.53	0.11	0.00	3.99
		12	39886.51	32742.91	32687.14	32742.91	32742.91	0.17	0.00	3.99
		16	50048.34	37068.82	35226.33	37068.82	37068.82	5.23	0.00	285.90
gr48	6	4	5378.00	4875.00	4815.05	4841.00	4841.00	1.25	0.00	2.82
		6	5406.00	4940.00	4805.00	4805.00	4805.00	2.81	0.00	1.76
		9	5294.00	4958.00	4871.67	4926.00	4926.00	1.77	0.00	13.70
		12	5415.00	5011.00	4994.37	5011.00	5011.00	0.33	0.00	4.91
		16	5788.00	5445.00	5320.65	5445.00	5445.00	2.34	0.00	24.25
hk48	6	4	13239.00	11283.00	11234.16	11271.00	11271.00	0.43	0.00	3.48
		6	12918.00	11226.00	11197.00	11197.00	11197.00	0.26	0.00	2.88
		9	12920.00	11465.00	11254.17	11292.00	11292.00	1.87	0.00	3.05
		12	13598.00	11522.00	11386.43	11450.00	11450.00	1.19	0.00	3.41
		16	17038.00	12215.00	11973.75	12215.00	12215.00	2.01	0.00	10.04
eil51	6	5	450.81	424.78	421.62	422.32	422.32	0.75	0.00	4.58
		7	454.93	426.59	422.71	424.36	424.36	0.92	0.00	6.88
		10	461.65	435.49	427.57	432.49	432.49	1.85	0.00	41.32
		12	475.71	437.27	432.71	436.59	436.59	1.05	0.00	14.41
		17	523.33	473.98	457.13	473.98	473.98	3.69	0.00	50.96
berlin52	8	5	8931.54	7254.04	7170.33	7182.23	7182.23	1.17	0.00	3.66
		7	8706.83	7266.75	7167.20	7167.20	7167.20	1.39	0.00	2.57
		10	8541.64	7206.70	7200.48	7206.70	7206.70	0.09	0.00	4.43
		13	8790.80	7298.63	7287.14	7298.63	7298.63	0.16	0.00	4.68
		17	10026.53	7800.77	7632.78	7800.77	7800.77	2.20	0.00	48.81

Table 6
Computational results for the new TSPLIB instances, medium and large size.

Instance	p^*	p	Giant tour result	ILS result	Initial bound	Final bound	Best solution	Initial gap (percent)	Final gap (percent)	CPU time (seconds)
brazil58	12	5	26918.00	22578.00	21170.75	21744.00	21744.00	6.65	0.00	78.90
		8	25795.00	22367.00	21081.50	21289.00	21289.00	6.10	0.00	36.95
		11	25058.00	21080.00	21080.00	21080.00	21080.00	0.00	0.00	5.14
		14	25496.00	21221.00	21221.00	21221.00	21221.00	0.00	0.00	4.72
		19	36060.00	22635.00	22340.99	22635.00	22635.00	1.32	0.00	31.13
st70	12	7	781.60	647.95	633.29	638.22	638.22	2.31	0.00	18.11
		10	761.78	642.01	630.63	632.54	632.54	1.80	0.00	12.56
		14	755.61	634.48	630.90	630.90	630.90	0.57	0.00	8.66
		17	758.10	636.40	635.51	636.19	636.19	0.14	0.00	11.16
		23	936.65	694.49	664.05	694.49	694.49	4.58	0.00	1137.77
eil76	9	7	617.05	548.83	542.73	542.95	542.95	1.12	0.00	20.97
		10	624.52	551.98	544.51	545.02	545.02	1.37	0.00	18.60
		15	634.25	556.20	549.16	552.15	552.15	1.28	0.00	207.04
		19	661.99	563.95	557.28	563.95	563.95	1.20	0.00	371.35
		25	727.71	601.71	587.99	601.71	601.71	2.33	0.00	1025.73
pr76	15	7	115538.61	103315.32	101091.86	101401.33	101401.33	2.20	0.00	25.29
		10	116374.38	104034.03	101165.57	101779.42	101779.42	2.84	0.00	224.40
		15	118057.94	103867.82	102513.09	103097.47	103822.35	1.32	0.70	3608.81
		19	120906.74	104481.75	104036.62	104481.75	104481.75	0.43	0.00	45.62
		25	141308.46	110073.94	108095.83	110073.94	110073.94	1.83	0.00	867.49
rat99	12	9	1373.55	1222.85	1208.05	1209.14	1209.14	1.23	0.00	90.16
		14	1405.01	1249.35	1216.87	1217.48	1249.35	2.67	2.62	3622.70
		19	1417.42	1264.52	1235.64	1236.82	1264.52	2.34	2.24	3618.81
		24	1457.77	1276.13	1257.16	1261.17	1276.13	1.51	1.19	3621.86
		33	1640.90	1373.37	1325.59	1334.28	1373.37	3.60	2.93	3609.14
kroA100	19	10	24687.70	20293.87	19556.81	19900.87	19900.87	3.77	0.00	2993.41
		14	24571.39	20131.41	19568.28	19637.52	19637.52	2.88	0.00	40.47
		20	24762.78	20142.01	19777.63	19866.93	19868.64	1.84	0.01	57.24
		25	25013.19	20279.51	20226.30	20279.51	20279.51	0.26	0.00	77.87
		33	30298.85	22303.23	21445.62	21761.87	22303.23	4.00	2.49	3609.77
kroB100	23	10	28199.43	21147.13	20495.53	20823.12	20823.12	3.18	0.00	1575.86
		14	27887.13	20801.65	20495.10	20762.88	20762.88	1.50	0.00	1292.72
		20	27796.09	21608.68	20475.75	20660.05	20660.05	5.53	0.00	114.70
		25	28688.37	21086.63	20737.89	20786.92	20786.92	1.68	0.00	34.89
		33	34719.88	22923.42	21992.60	22412.71	22923.42	4.23	2.28	3610.08
kroC100	23	10	27112.43	20199.75	19841.12	19923.30	19923.30	1.81	0.00	93.61
		14	27258.71	19980.32	19855.85	19938.84	19938.84	0.63	0.00	77.78
		20	27420.27	20186.38	20013.27	20135.00	20135.00	0.86	0.00	229.62
		25	28174.46	20678.11	20305.76	20427.96	20427.96	1.83	0.00	197.60
		33	33693.89	22465.73	21371.77	21559.53	22465.73	5.12	4.20	3609.81
kroD100	23	10	26908.84	20460.34	20226.34	20270.57	20270.57	1.16	0.00	50.50
		14	26767.04	20790.60	20200.88	20267.23	20267.23	2.92	0.00	46.87
		20	27020.74	20753.74	20352.89	20457.00	20457.00	1.97	0.00	254.33
		25	27776.37	20761.87	20575.72	20671.19	20671.19	0.90	0.00	154.50
		33	34414.67	22238.56	21533.97	22011.87	22238.56	3.27	1.03	3609.46
kroE100	29	10	26647.12	20977.21	20766.43	20766.43	20766.43	1.02	0.00	28.92
		14	26540.41	20777.69	20760.78	20777.69	20777.69	0.08	0.00	28.45
		20	26873.22	20937.39	20924.83	20937.39	20937.39	0.06	0.00	51.43
		25	26990.05	21233.77	21110.75	21174.94	21174.94	0.58	0.00	62.60
		33	32270.60	22782.98	22157.75	22782.98	22782.98	2.82	0.00	3054.13
rd100	19	10	9750.83	7642.76	7489.73	7524.08	7524.08	2.04	0.00	177.19
		14	9387.75	7542.23	7479.44	7500.44	7500.44	0.84	0.00	42.96
		20	9351.41	7582.28	7507.01	7537.98	7537.98	1.00	0.00	149.61
		25	9222.18	7555.83	7550.19	7555.83	7555.83	0.07	0.00	51.30
		33	12006.89	8131.25	7837.31	7996.03	8131.25	3.75	1.69	3609.83

initial gaps is 0.04 percent on the average, with a maximum of 0.4 percent and a minimum of −0.13 percent. We conclude that separating (12) is marginally useful for the smaller instances, but its importance grows as $|V|$ increases. The initial optimality gap for the HpMP2 is quite small, 1.76 percent on average with a maximum of 13.87 percent. An analysis of the table shows that the initial gap for HpMP2 dominates Model 1 in 20 out of 25 parametric settings, and Model 3 in 17 out of 25 parametric settings. We emphasize that the initial gap depends on the strength of the lower bound, on the quality of the initial feasible solution, and on the performance separation heuristics for the valid inequalities for which we do not have a polynomial time exact separation algorithm. The quality of the lower bound is reflected into the computational performance, where our branch-and-cut algorithm for the HpMP2 successfully solves all instances for $|V| = 60$ and $p = 20$ and Models 1 and 3 cannot solve any.

The most striking difference between the initial optimality gaps occurs for the largest values of p, which force at most one cycle to include more than three vertices. This is due to the valid inequality set (13), which provides a lower bound on the number of vertices assigned to a vertex selected as a depot. Unlike (11) and (12) that replace (6) and (7) and are necessary for the validity of HpMP2, (13) is optional. During our computational experiments, we have observed that adding all members of (13) to the formulation slows down the branch-and-cut algorithm. However, we have also observed that it is not possible to solve large instances without these inequalities. In our final implementation, we have chosen to add these inequalities when the maximum

possible cycle size $|V| - 3(p-1)$ is less than or equal to 5 for the best performance.

The results of our experiments on the TSPLIB instances are presented in Table 4. HpMP2 outperforms both Model 1 and Model 3 in terms of the CPU time for the instances that could be solved within the time limit. For the instances that could not be solved, the best lower bounds are also provided by HpMP2. To increase the size of the test bed for HpMP, we have solved 22 instances from the TSPLIB with sizes ranging from 21 to 100. For instances with vertex coordinates, we have computed and used the Euclidean distances between the vertices. For the rest of the instances, we have used the edge weights provided. We have used $p \in \{\lfloor |V|/10 \rfloor, \lfloor |V|/7 \rfloor, \lfloor |V|/5 \rfloor, \lfloor |V|/4 \rfloor, \lfloor |V|/3 \rfloor\}$ for our experiments. The computational results of the new TSPLIB instances are reported in Tables 5 and 6. The performance of HpMP2 on these instances is similar to its performance on the instance set of Gollowitzer et al. (2014). We have successfully solved 100 instance out of 110, with the unsolved instances limited to $|V| \geq 76$. Notably, $p = \lfloor |V|/3 \rfloor$ for 6 out of the 10 unsolved instances. The hardest instance was observed to be *rat99*, which is a "rattled grid" consisting of grid coordinates with minor perturbations. We attribute this to the high degree of symmetry inherent to this instance.

5. Conclusions

We have studied the HpMP, provided an exact algorithm, a giant tour heuristic, and an ILS algorithm. Our exact algorithm is a branch-and-cut algorithm based on an enhanced p-median formulation. We have provided two sets of valid inequalities which we prove to dominate the ones in the literature, and a third set of inequalities that have not been proposed before. The giant tour heuristic is based on a dynamic programming formulation that optimally splits a TSP tour into p cycles. We have performed computational experiments on HpMP instances from the literature and new instances from the TSPLIB, and showed that the performance and the computational reach of our branch-and-cut algorithm is better than that in the literature. We have observed the ILS algorithm to be capable of returning high quality solutions within a minute, with an average optimality gap of 0.64 percent.

Acknowledgments

We thank Stefan Gollowitzer for providing the benchmark problem instances, and the two anonymous reviewers whose suggestions have improved the paper. This study was partially supported by Centre for Operational Research, Management Science and Information Systems based within the University of Southampton, by the Canadian Natural Sciences and Engineering Research Council under grant 2015-016189, and by the Spanish Ministry of Education and Science/FEDER grants numbers MTM2013-46962-C02-02. This support is gratefully acknowledged.

References

Afsar, H., Prins, C., & Santos, A. (2014). Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. *International Transactions in Operational Research, 21*(1), 153–175.

Barahona, F. (1996). Network design using cut inequalities. *SIAM Journal on Optimization, 6*(3), 823–837.

Beasley, J. (1983). Route first - cluster second methods for vehicle routing. *Omega, 11*(4), 403–408.

Branco, I., & Coelho, J. (1990). The Hamiltonian p-median problem. *European Journal of Operational Research, 47*(1), 86–95.

Glaab, H. (2002). A new variant of a vehicle routing problem: Lower and upper bounds. *European Journal of Operational Research, 139*(3), 557–577.

Glaab, H., & Pott, A. (2000). The Hamiltonian p-median problem. *The Electronic Journal of Combinatorics, 7*, 1–25.

Gollowitzer, S., Gouveia, L., Laporte, G., Pereira, D., & Wojciechowski, A. (2014). A comparison of several models for the Hamiltonian p-median problem. *Networks, 63*, 350–363.

Gollowitzer, S., Pereira, D. L., & Wojciechowski, A. (2011). New models for and numerical tests of the Hamiltonian p-median problem. In J. Pahl, T. Reiners, & S. Voss (Eds.), *Network optimization*. In *Lecture Notes in Computer Science: Vol. 6701* (pp. 385–394). Springer Berlin Heidelberg.

Hupp, L., & Liers, F. (2013). A polyhedral study of the Hamiltonian p-median problem. *Electronic Notes in Discrete Mathematics, 41*, 213–220.

Laporte, G., Nobert, Y., & Pelletier, P. (1983). Hamiltonian location problems. *European Journal of Operational Research, 12*(1), 82–89.

Love, R. (1976). Note – one-dimensional facility location-allocation using dynamic programming. *Management Science, 22*(5), 614–617.

Miller, D., & Pekny, J. (1995). A staged primal-dual algorithm for perfect b-matching with edge capacities. *ORSA Journal on Computing, 7*(3), 298–320.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research, 31*(12), 1985–2002.

Reinelt, G. (1991). TSPLIB–A traveling salesman problem library. *ORSA Journal on Computing, 3*(4), 376–384.

Uster, H., & Kumar, S. (2010). Algorithms for the design of network topologies with balanced disjoint rings. *Journal of Heuristics, 16*(1), 37–63.

Zohrehbandian, M. (2007). A new formulation of the Hamiltonian p-median problem. *Applied Mathematical Sciences, 1*(8), 355–361.