Backtracking Algorithms for Constructing the Hamiltonian Decomposition of a 4-Regular Multigraph

A. V. Korostil*, (ORCID: 0000-0003-1881-0207)
and A. V. Nikolaev**, (ORCID: 0000-0003-4705-2409)

a Demidov Yaroslavl State University, Yaroslavl, 150003 Russia
*e-mail: av.korostil@gmail.com
**e-mail: andrei.v.nikolaev@gmail.com
Received February 15, 2021; revised March 10, 2021; accepted March 12, 2021

Abstract—We consider a Hamiltonian decomposition problem of partitioning a regular graph into edge-disjoint Hamiltonian cycles. It is known that verifying vertex non-adjacency in the 1-skeleton of the symmetric and asymmetric traveling salesperson polytopes is NP-complete. On the other hand, a sufficient condition for two vertices to be non-adjacent can be formulated as a combinatorial problem of finding a second Hamiltonian decomposition of a 4-regular multigraph. We present two backtracking algorithms for constructing a second Hamiltonian decomposition and verifying vertex non-adjacency: an algorithm based on a simple path extension and an algorithm based on the chain edge fixing procedure. Based on the results of computational experiments for undirected multigraphs, both backtracking algorithms lost to the known general variable neighborhood search heuristics. However, for directed multigraphs, the algorithm based on chain fixing of edges showed results comparable to heuristics on instances with an existing solution and better results on infeasible instances where the Hamiltonian decomposition does not exist.

Keywords: Hamiltonian decomposition, traveling salesperson polytope, 1-skeleton, vertex adjacency, backtracking

DOI: 10.3103/S0146411622070082

INTRODUCTION

Hamiltonian decomposition of a regular multigraph is a partition of its edge set into Hamiltonian cycles. The problem of finding edge-disjoint Hamiltonian cycles in a given regular graph finds applications in combinatorial optimization [23], coding theory [5, 6], algorithms for distributed data mining [12], analysis of interconnected networks [18] and other areas. See also theoretical results on estimating the number of Hamiltonian decompositions of regular graphs [16]. In this paper, the problem of constructing a Hamiltonian decomposition arises from the field of polyhedral combinatorics.

1. TRAVELING SALESPERSON POLYTOPE

We consider the classic formulation of the traveling salesperson problem: given a complete weighted graph (or digraph) $K_n = (V, E)$, find a Hamiltonian cycle of minimum weight. Denote by $HC_n$ the set of all Hamiltonian cycles in the graph $K_n$ and assign to each Hamiltonian cycle $x \in HC_n$ the characteristic vector $x^e \in \mathbb{R}^E$ according to the following rule:

$$x^e_v = \begin{cases} 1, & \text{if the cycle } x \text{ contains an edge } e, \\ 0, & \text{otherwise}. \end{cases}$$

Polytope

$$\text{STSP}(n) = \text{conv}\{x^e | x \in HC_n\}$$

is called the symmetric traveling salesperson polytope.

The asymmetric traveling salesperson polytope $\text{ATSP}(n)$ is defined similarly as the convex hull of the characteristic vectors of all possible Hamiltonian cycles in the complete digraph $K_n$. 

688
The integer linear programming approach for the traveling salesperson problem was introduced by Dantzig, Fulkerson, and Johnson in their classical work for 49 US cities [14]. State-of-the-art exact algorithms for the traveling salesperson problem are based on a partial description of the facets of the traveling salesman polytope and the branch and cut method for integer linear programming [2].

The 1-skeleton of a polytope is a graph whose vertices are the vertices of the polytope and edges are geometric edges (one-dimensional faces). The study of 1-skeleton is of interest, since, on the one hand, some combinatorial algorithms for such problems as perfect matching, set covering, independent set, object ranking, problems with fuzzy measures, and some others are based on the adjacency relation in a 1-skeleton and the local search technique (when we move from the current solution to the “best solution” among adjacent ones) [1, 7, 11, 13, 15]. On the other hand, some characteristics of the 1-skeleton of the problem, such as the diameter and the clique number (the number of vertices in the largest clique), estimate time complexity for various computational models and classes of algorithms [8, 9, 17].

Unfortunately, the classical result by Papadimitriou prevents the study of the 1-skeleton of the traveling salesperson polytope.

**Theorem 1** (Papadimitriou). The question of whether two vertices of the polytopes STSP(n) or ATSP(n) are non-adjacent is NP-complete.

Note that the complementary problem of finding whether two vertices of the 1-skeleton of the traveling salesperson polytope are adjacent is co-NP-complete.

### 2. HAMILTONIAN DECOMPOSITION AND SUFFICIENT CONDITION FOR VERTEX NON-ADJACENCY

As a result of Papadimitriou’s theorem on the NP-completeness of verifying the vertex non-adjacency in the 1-skeleton of the traveling salesperson polytope, sufficient conditions for non-adjacency are of interest. In particular, polynomial sufficient conditions are known for pyramidal tours [10], pyramidal tours with step-backs [24], and pedigrees [3, 4]. In this paper, we consider the most general of the known – sufficient condition by Rao [28].

Let \( x = (V, E_x) \) and \( y = (V, E_y) \) be two Hamiltonian cycles on the vertex set \( V \). We denote by \( x \cup y \) a union multigraph \( (V, E_x \cup E_y) \) that contains a copy of each edge of \( x \) and \( y \). Note that if two cycles contain the same edge \( e \), then both copies of the edge are added to the multigraph \( x \cup y \).

**Lemma 2** (Rao [28]). Given two Hamiltonian cycles \( x \) and \( y \), if the union multigraph \( x \cup y \) contains two edge-disjoint Hamiltonian cycles \( z \) and \( w \) different from \( x \) and \( y \), then the corresponding vertices \( x^r \) and \( y^r \) of the traveling salesperson polytope STSP(n) (or ATSP(n)) are not adjacent.

From a geometric point of view, the sufficient condition by Rao means that the segment connecting two vertices \( x^r \) and \( y^r \) intersects the segment connecting two other vertices \( z^r \) and \( w^r \) of the traveling salesperson polytope, therefore, they cannot be adjacent. An example of a satisfied sufficient condition is shown in Fig. 1.

Thus, verifying the vertex non-adjacency in the 1-skeleton of the traveling salesperson polytope can be reduced to finding a Hamiltonian decomposition of the union multigraph \( x \cup y \) that is different from the given one. Let us formulate a sufficient condition for vertex non-adjacency in the form of a combinatorial problem.

**Second Hamiltonian decomposition problem.**

**Instance:** let \( x \) and \( y \) be two Hamiltonian cycles.

**Question:** does the union multigraph \( x \cup y \) contain a pair of edge-disjoint Hamiltonian cycles \( z \) and \( w \) different from \( x \) and \( y \)?

Note that checking whether an arbitrary graph contains a Hamiltonian decomposition is an NP-complete problem already for 4-regular undirected multigraphs and 2-regular directed multigraphs [27].

Previously, the second Hamiltonian decomposition problem and its application to verifying the vertex non-adjacency in the 1-skeletons of the polytopes STSP(n) and ATSP(n) were considered in [22, 25], where several heuristic algorithms were proposed based on constructing vertex-disjoint cycle covers of a graph: simulated annealing and general variable neighborhood search. Heuristic algorithms have proven to be very efficient on instances that have a solution, especially on undirected graphs. However, in instances that do not have a solution, heuristics face significant difficulties. In this paper, we consider two exact backtracking algorithms for constructing the second Hamiltonian decomposition of a 4-regular multigraph.
3. BACKTRACKING ALGORITHM BASED ON SIMPLE PATH EXTENSION

Recall that backtracking is one of the general methods for finding a solution to a problem by exhaustive search. The procedure consists of a successive extension of a partial solution. If at the next step such an extension fails, then we backtrack to a shorter partial solution and continue the search further [29].

In [21], a backtracking algorithm was presented for the problem of finding a second Hamiltonian decomposition of a 4-regular multigraph based on extending a simple path. Below is a modified version of this algorithm.

Let here and below the partial solution consist of two components \( z \) and \( w \). The idea of the algorithm is to sequentially construct a simple path in the component \( z \). Whereas edges not included in \( z \) are sent to the component \( w \).

Consider the example of the \( x \cup y \) multigraph shown in Fig. 1. Let us construct a partial solution for it, corresponding to the simple path 2–3–5–6 (Fig. 2). Here, the edges of the \( z \) component are solid, and
the edges of the $w$ component are dashed. Since the degree of each vertex in the multigraph $x \cup y$ is equal to 4, then we can continue a simple path in $z$ from the vertex 6 in no more than three ways: $(6,1)$, $(6,2)$ and $(6,4)$. Moreover, whichever edge we choose to continue the path, since two edges incident to the vertex 6 have already been added to $z$, the other two edges can only fall into the component $w$.

We sequentially consider all three options and for each we check the correctness of the partial solution:

- Extension 1 is invalid because it contains a cycle at vertices 2, 3, 4, 6 in the component $w$.
- Extension 2 is invalid because it contains a cycle at vertices 2, 3, 5, 6 in the component $z$.
- Extension 3 is correct.

In the general case, the correctness conditions for a partial solution have the following form:

- $z$ contains a simple path (by construction) or a Hamiltonian cycle.
- $w$ contains a forest (a graph without cycles) with vertex degrees at most 2 or a Hamiltonian cycle.

If the extension of the partial solution turned out to be incorrect, then we backtrack and consider the next option. If all three extensions are invalid, then this partial solution cannot be extended, and we backtrack to a shorter partial solution.

The general scheme for undirected graphs is presented in the Algorithm 1.

The only difference for directed graphs is that the outdegree of each vertex of the multigraph $x \cup y$ is equal to two, hence there are at most two options for extending a simple path in $z$. All other steps are completely similar.

**Algorithm 1. Backtracking based on simple path extension**

1: procedure Backtracking_Simple_Path ($z, w, (i, j), x \cup y$)
2:   Add the edge $(i, j)$ to $z$
3:   if vertex $i$ in $z$ has 2 incident edges then
4:     Add free edges incident to $i$ to the component $w$
5:   end if
6:   if partial solution $z, w$ is not correct then
7:     return $z, w$ is not correct → Backtrack to the previous step
8:   end if
9:   if $z$ and $w$ are Hamiltonian cycles (different from $x$ and $y$) then
10:      return Hamiltonian decomposition of $z$ and $w$ → Solution is found
11:   end if
12:  Sort edges $(j, k)$ from $j$ in ascending order of free degrees of vertices $k$
13:  for each free edge $(j, k)$ from the vertex $j$ do
14:    $z, w \leftarrow$ Backtracking_Simple_Path ($z, w, (j, k), x \cup y$)
15:    if Hamiltonian decomposition is found then
16:      return Hamiltonian decomposition of $z$ and $w$
17:    end if
18:  end for
19: end procedure

20: procedure Algorithm_Simple_Path ($x \cup y$)
21:   $z, w \leftarrow \emptyset$
22:   Choose an initial edge $(i, j)$ of the multigraph $x \cup y$
23:   $z, w \leftarrow$ Backtracking_Simple_Path ($z, w, (i, j), x \cup y$)
24:   if $z$ and $w$ are found then
25:      return Hamiltonian decomposition $z$ and $w$ is found (vertices are not adjacent)
26:   end if
27:   return Hamiltonian decomposition not found (vertices are probably adjacent)
28: end procedure
4. BACKTRACKING ALGORITHM BASED ON CHAIN EDGE FIXING

The second backtracking algorithm is based on the chain fixation of edges in the components $z$ and $w$. Algorithms for directed and undirected graphs are slightly different, so we describe them separately.

4.1. Directed Multigraphs

We consider a directed 2-regular multigraph $x \cup y$ for which the indegrees and outdegrees of each vertex are equal to two. Let us choose some edge $(i, j)$ and fix it in the component $z$, then the second edge $(i, k)$ outgoing from $i$ and the second edge $(h, j)$ incoming into $j$ cannot get into $z$. We will fix these edges in $w$ (Fig. 3).

The main idea of the algorithm is that the edges $(i, k)$ and $(h, j)$ fixed in $w$, in turn, start recursive chains of fixing edges in $z$, and so on.

As an example, consider the directed 2-regular multigraph shown in Fig. 4. Note that the multigraph contains two copies of the edge $(2, 3)$. Multiple edges cannot get into the same Hamiltonian cycle, so we fix one copy each in $z$ and $w$. Let us choose some edge, for example $(1, 2)$, and fix it in the $z$ component (solid edges), then:

1. Edge $(1, 2)$ is fixed in $z$, hence edges $(1, 4)$ and $(6, 2)$ go to $w$ (dashed edges).
2. Edge $(1, 4)$ is fixed in $w$, hence edge $(3, 4)$ goes to $z$, edge $(6, 2)$ is fixed in $w$, hence edge $(6, 1)$ goes to $z$.
3. Edge $(3, 4)$ is fixed in $z$, hence edge $(3, 5)$ goes to $w$, edge $(6, 1)$ is fixed in $z$, hence edge $(5, 1)$ goes to $w$.
4. Edge $(3, 5)$ is fixed in $w$, hence edge $(4, 5)$ goes to $z$, edge $(5, 1)$ is fixed in $w$, hence edge $(5, 6)$ goes to $z$.

At the output, given only one edge $(1, 2)$ fixed in $z$, we obtain unique Hamiltonian cycles $z$ and $w$ (Fig. 4). Considering that the edge $(1, 2)$ must belong to at least one Hamiltonian cycle, this 2-regular directed multigraph contains a unique Hamiltonian decomposition, which was found.
The general backtracking scheme for directed multigraphs is given in the pseudocode of the Algorithm 2.

**Algorithm 2.** Backtracking based on chain edge fixing for directed graphs

1. procedure Chain_Edge_Fixing_Directed \((i, j)\ in z\)  
   \(\triangleright\) Recursive edge fixing
2. Fix the edge \((i, j)\ in z\)
3. if edge \((i, k)\) is not fixed then
4. Chain_Edge_Fixing_Directed \((i, k)\ in w\)
5. end if
6. if edge \((h, j)\) is not fixed then
7. Chain_Edge_Fixing_Directed \((h, j)\ in w\)
8. end if
9. end procedure
10. procedure Backtracking_Chain_Edge_Fixing_Directed \((z, w, (i, j), x \cup y)\)
11. Chain_Edge_Fixing_Directed \((i, j)\ in z\)
12. if \(z\) and \(w\) are Hamiltonian cycles (different from \(x\) and \(y\)) then
13. \(\triangleright\) Solution is found
14. return \(z\) and \(w\)
15. end if
16. if \(z\) or \(w\) contains a non-Hamiltonian cycle then
17. \(\triangleright\) Partial solution is incorrect
18. return
19. end if
20. Select vertex \(i'\) with free outcoming edges
21. for each free edge \((i', j')\) outcoming from the vertex \(i'\) do
22. \(z, w \leftarrow \) Backtracking_Chain_Edge_Fixing_Directed \((z, w, (i', j'), x \cup y)\)
23. if Hamiltonian decomposition is found then
24. \(\triangleright\) Backtrack to the previous step
25. return Hamiltonian decomposition \(z\) and \(w\)
26. end if
27. end for
28. end procedure
29. procedure Algorithm_Chain_Edge_Fixing_Directed \((x \cup y)\)
30. \(z, w \leftarrow \emptyset\)
31. Find multiple edges in \(x \cup y\) and fix one copy in \(z\) and \(w\)
32. Choose a free edge \((i, j)\) of the multigraph \(x \cup y\)
33. \(z, w \leftarrow \) Backtracking_Chain_Edge_Fixing_Directed \((z, w, (i, j), x \cup y)\)
34. if \(z\) and \(w\) are found then
35. \(\triangleright\) Hamiltonian decomposition of \(z\) and \(w\) is found (vertices are not adjacent)
36. return Hamiltonian decomposition not found (vertices are probably adjacent)
37. end if
38. end procedure

At the data preprocessing step (line 28, Algorithm 2), we find all multiple edges in the multigraph \(x \cup y\) and fix one copy in \(z\) and \(w\), since these edges cannot get into the same Hamiltonian cycle.

A partial solution is considered correct if the components \(z\) and \(w\) are directed acyclic graphs with indegrees and outdegrees at most one or directed Hamiltonian cycles. By construction, the indegrees and outdegrees of vertices in \(z\) and \(w\) cannot be equal to two, so to check the correctness it suffices to verify that there are no non-Hamiltonian cycles.

In contrast to the backtracking algorithm based on the simple path extension (Algorithm 1), we choose the vertex \(i'\) to extend the partial solution (line 18, Algorithm 2) in such a way that the chain edge fixing...
starts from both head and tail of the edge \((i', j')\). The more edges are fixed at one step, the smaller the recursion depth will be.

Note that we do not branch when choosing and fixing the first free edge \((i, j)\) (line 29, Algorithm 2), since this edge must get into one of the solution components. Let us call the component \(z\) the one that contains the edge \((i, j)\).

It should also be noted that although the procedure Chain_Edge_Fixing_Directed (lines 1–9, Algorithm 2) at each step calls up to two of its subroutines, the total complexity is linear \((O(V))\) since each edge can be fixed at most once, and \(|E| = 2|V|\).

### 4.2. Undirected Multigraphs

The pseudo-code of the algorithm for undirected multigraphs is presented in Algorithm 3.

**Algorithm 3.** Backtracking based on chained edge fixing for undirected graphs

```plaintext
1: procedure Chain_Edge_Fixing_Undirected \((i, j)\) in \(z\)
2:     Fix the edge \((i, j)\) in \(z\)
3: if vertex \(i\) in \(z\) is incident with 2 fixed edges then
4:     Chain_Edge_Fixing_Undirected \((i, k)\) in \(w\) \(\triangleright\) Fix two other edges in \(w\)
5:     Chain_Edge_Fixing_Undirected \((i, h)\) in \(w\)
6: end if
7: if vertex \(j\) in \(z\) is incident with 2 fixed edges then
8:     Chain_Edge_Fixing_Undirected \((j, k)\) in \(w\) \(\triangleright\) Fix two other edges in \(w\)
9:     Chain_Edge_Fixing_Undirected \((j, h)\) in \(w\)
10: end if
11: end procedure
12: procedure Backtracking_Chain_Edge_Fixing_Undirected \((z, w, (i, j), x \cup y)\)
13:     Chain_Edge_Fixing_Undirected \((i, j)\) in \(z\)
14: if \(z\) and \(w\) are Hamiltonian cycles (different from \(x\) and \(y\)) then
15:     return \(z\) and \(w\) \(\triangleright\) Solution is found
16: end if
17: if \(z\) or \(w\) contains a non-Hamiltonian cycle then \(\triangleright\) Partial solution is incorrect
18:     return \(\triangleright\) Backtrack to the previous step
19: end if
20: Choose a vertex \(i\) with the minimum degree \(d\) of free edges
21: Sort edges \((i, j_k)\) incident to \(i\) in ascending order of free powers \(j_k\)
22: for \(k \leftarrow 1\) to \(d\) do
23:     \(z, w \leftarrow\) Backtracking_Chain_Edge_Fixing_Undirected \((z, w, (i, j_k), x \cup y)\)
24:     if Hamiltonian decomposition is found then
25:         return Hamiltonian decomposition \(z\) and \(w\)
26:     end if
27: end for
28: end procedure
```

The chain edge fixing procedure is triggered as soon as 2 edges are incident to a vertex in one of the partial solutions, then the other two edges are sent to another partial solution (lines 3–10, Algorithm 3).

A partial solution is considered correct if the components \(z\) and \(w\) are forests with vertex degrees at most two or Hamiltonian cycles. By construction, we add to \(z\) and \(w\) one edge at a time. Moreover, as soon as the degree of some vertex in one of the components becomes equal to two, then the two remaining edges
are sent to another component. Thus, to check the correctness of a partial solution, it suffices to verify that \( z \) and \( w \) do not contain non-Hamiltonian cycles.

Note that at each step, to expand the partial solution, we choose a vertex \( i \) with the minimum degree over free edges in order to reduce the branching factor of the recursion. Edges incident to \( i \) are considered in ascending order of degrees of adjacent vertices along free edges (lines 20–27, Algorithm 3).

### 4.3. Vertex Adjacency and Data Preprocessing

Let us note that the algorithms considered in this paper were developed for the problem of constructing a Hamiltonian decomposition of a 4-regular multigraph \( x \cup y \). The algorithms refer directly to the cycles \( x \) and \( y \) only when checking that the second decomposition \( z \) and \( w \) is different from the original one. If we omit this check, then we get algorithms for constructing a Hamiltonian decomposition without any reference to polyhedral combinatorics. Otherwise, if we are interested in exactly the adjacency of vertices in the 1-skeleton of the traveling salesperson polytope, then we can strengthen the considered algorithms by adding a data preprocessing step and checking the known polynomial sufficient conditions for non-adjacency: pyramidal tours [10], pyramidal tours with step-backs [24], pedigrees [3, 4], etc.

### 5. COMPUTATIONAL EXPERIMENTS

The algorithms were tested on random directed and undirected 4-regular multigraphs. For each graph size, 100 pairs of random permutations with a uniform probability distribution were generated using the Fisher–Yates shuffle algorithm [20].

For comparison, we chose 3 algorithms:
- backtracking based on simple path extension (BSP);
- backtracking based on chain edge fixing (BCEF);
- general variable neighborhood search (GVNS) heuristic algorithm from [25], which is a modification of the simulated annealing algorithm [22] and is based on constructing a vertex-disjoint cycle covers by reduction to perfect matching and several cycle merging operations.

Backtracking algorithms are implemented in Python, for the general variable neighborhood search heuristics, a ready-made implementation in Node.js [25] is taken. Computational experiments were carried out on an Intel(R) Core(TM) i5-4460 machine with a 3.20GHz CPU and 16GB RAM.

**Table 1.** Computational results for 100 random undirected Hamiltonian cycles

| \( |V| \) | \( |V| \) time, s | \( |V| \) times, s | \( |V| \) time, s | \( |V| \) times, s | \( |V| \) time, s | \( |V| \) time, s |
|---|---|---|---|---|---|---|
| 32 | 0.030 | – | – | 0.015 | – | – | 0.001 | – | – |
| 48 | 0.151 | – | – | 0.032 | – | – | 0.002 | – | – |
| 64 | 0.154 | – | – | 0.052 | – | – | 0.003 | – | – |
| 96 | 0.908 | – | – | 0.09 | – | – | 0.007 | – | – |
| 128 | 3.045 | – | – | 0.157 | – | – | 0.012 | – | – |
| 192 | 4.257 | – | – | 0.22 | – | – | 0.023 | – | – |
| 256 | 19.625 | – | – | 0.487 | – | – | 0.033 | – | – |
| 384 | 80.037 | – | – | 1.278 | – | – | 0.079 | – | – |
| 512 | 9 | – | – | 2.447 | – | – | 0.121 | – | – |
| 768 | 3.391 | – | – | 10.0000 | – | – | 0.272 | – | – |
| 1024 | 6.305 | – | – | 8.799 | – | – | 1.011 | – | – |
| 1536 | 18.499 | – | – | 18.499 | – | – | 1.821 | – | – |
| 2048 | 18.499 | – | – | 45.542 | – | – | 3.852 | – | – |
| 3072 | 0 | – | – | 82.973 | – | – | 7.768 | – | – |
| 4096 | 0 | – | – | 82.973 | – | – | – | – | – |
The results of computational experiments for undirected graphs are presented in Table 1 and in Fig. 5. Results for directed graphs are presented in Table 2 and in Figs. 6 and 7.

For each set of 100 tests, the average running time of the algorithms in seconds is given separately for feasible and infeasible instances. For two backtracking algorithms (BSP and BCEF), a time limit of 2 h was set for each set of 100 test problems. Accordingly, the tables indicate how many instances out of 100 the algorithm managed to solve in 2 h. The general variable neighborhood search heuristics (GVNS) was set to a limit of 1 minute per instance. This is because the heuristic algorithm has a one-sided error. If the

![Average runtime vs. Graph order](image_url)

Fig. 5. Computational results for undirected graphs.

| $|V|$ | BSP feasible | BSP infeasible | BCEF feasible | BCEF infeasible | GVNS feasible | GVNS infeasible |
|-----|--------------|----------------|---------------|----------------|---------------|----------------|
|     | $N$ time, s  | $N$ times, s  | $N$ time, s   | $N$ time, s   | $N$ time, s  | $N$ time, s   |
| 32  | 30 0.069     | 70 0.592       | 30 0.006      | 70 0.011       | 30 0.001      | 70 0.248       |
| 48  | 20 1.838     | 80 17.005      | 20 0.011      | 80 0.023       | 20 0.002      | 80 0.440       |
| 64  | 8 36.850     | 17 403.016     | 20 0.026      | 80 0.044       | 20 0.002      | 80 0.714       |
| 96  | 0 –          | 0 –            | 19 0.032      | 81 0.086       | 19 0.005      | 81 1.330       |
| 128 | – –          | – –            | 18 0.055      | 82 0.141       | 18 0.009      | 82 2.312       |
| 192 | – –          | – –            | 21 0.059      | 79 0.262       | 21 0.014      | 79 4.569       |
| 256 | – –          | – –            | 25 0.134      | 75 0.628       | 25 0.093      | 75 9.120       |
| 384 | – –          | – –            | 20 0.083      | 80 1.646       | 20 0.104      | 80 19.215      |
| 512 | – –          | – –            | 22 0.143      | 78 2.131       | 22 0.112      | 78 29.048      |
| 768 | – –          | – –            | 19 0.426      | 81 2.241       | 19 0.404      | 81 55.991      |
| 1024| – –          | – –            | 17 0.847      | 83 10.545      | 17 1.679      | 83 60.000      |
| 1536| – –          | – –            | 16 0.369      | 84 7.844       | 16 0.943      | 84 60.000      |
| 2048| – –          | – –            | 15 1.651      | 85 16.179      | 14 1.987      | 86 60.000      |
| 3072| – –          | – –            | 21 1.059      | 79 46.589      | 21 2.999      | 79 60.000      |
| 4096| – –          | – –            | 17 1.551      | 78 91.963      | 18 4.846      | 82 60.000      |
algorithm finds a solution, then the instance is feasible. However, the heuristic algorithm cannot guarantee that the problem is infeasible, only that the solution is not found in a given time or number of iterations.

It is known that random undirected regular graphs have a Hamiltonian decomposition with a very high probability [19]. Indeed, for all test problems on undirected multigraphs (Table 1), there was a second Hamiltonian decomposition into cycles different from the original ones, and the vertices of the traveling salesperson polytope were not adjacent. From a geometric point of view, this means that the degrees of vertices in a 1-skeleton are much less than the total number of vertices, so two random vertices are not adjacent with a high probability.

According to the computational results for undirected Hamiltonian cycles, both backtracking algorithms lost out to the general variable neighborhood search heuristics (GVNS). In the allotted time, the
algorithm based on simple path extension (BSP) solved 709 instances, the algorithm based on chain edge
fixing (BCEF) solved 1486 instances, and the general variable neighborhood search (GVNS) solved all
1500 out of 1500 instances. GVNS was on average 13 times faster than BCEF and 640 times faster than BSP.
Among the two backtracking algorithms, the chain edge fixing (BCEF) algorithm proved to be signifi-
cantly more efficient, solving all test instances up to 3072 vertices and showing the running time on the
solved problems on average 48 times faster than the simple path construction (BSP).

The computational results for directed multigraphs (Table 2, Figs. 6 and 7) turned out to be fundamen-
tally different. Only about 20% of random instances were feasible. Note that this does not mean that the
corresponding vertices of the traveling salesperson polytope are adjacent since the second Hamiltonian
decomposition problem only verifies a sufficient condition for the non-adjacency in a 1-skeleton.

The worst results of the three considered algorithms were shown by backtracking based on simple path
extension (BSP). In the allotted time, the algorithm solved only 225 instances out of 1500, without solving a
single problem for graphs with more than 64 vertices. While chain edge fixing (BCEF) solved 1495 instances,
and the general variable neighborhood search (GVNS) solved all instances. In terms of running time on fea-
sible problems, the GVNS heuristic algorithm showed an advantage over BCEF on small graphs up to
512 vertices by an average of 6 times. However, on graphs with more than 512 vertices, BCEF turned out
to be on average 2 times faster than GVNS. In general, for feasible problems, the BCEF and GVNS algo-
rithms showed comparable results. Some variations in performance may be due to different implementa-
tions of the algorithms. However, we note that with an increase in the graph order, the advantage of BCEF
over GVNS also increased. On the other hand, the general variable neighborhood search heuristics (GVNS)
has encountered significant difficulties in infeasible instances. The algorithm cannot determine
this scenario and exits only when the limit on running time or number of iterations is reached. On such
problems, the algorithm based on chain fixing of edges (BCEF) turned out to be on average 16 times faster
than GVNS. In part, this means that the iteration threshold for the heuristic algorithm could be lowered.
However, in this case, there would be a danger of losing existing solutions.

CONCLUSIONS

In this paper, two backtracking algorithms were considered for the problem of constructing a second
Hamiltonian decomposition of a 4-regular multigraph. According to the results of computational experi-
ments on random directed and undirected multigraphs, the backtracking algorithm based on chain edge
fixing turned out to be much more efficient than backtracking based on simple path extension. In addi-
tion, on directed multigraphs, the chain edge fixing showed comparable results with the previously known
general variable neighborhood search heuristics on feasible instances and significantly outperformed the
heuristics on infeasible problems.

The considered backtracking algorithms were developed for the problem of verifying the vertex non-
adjacency in a 1-skeleton of a traveling salesperson polytope. However, they can also be applied directly
to the problem of constructing the Hamiltonian decomposition of a regular multigraph and many of its
applications.

FUNDING

The research is supported by the Demidov Yaroslavl State University, project VIP-016.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. Aguilera, N.E., Katz, R.D., and Tolomei, P.B., Vertex adjacencies in the set covering polyhedron, Discrete Appl.
Math., 2017, vol. 218, pp. 40–56.
https://doi.org/10.1016/j.dam.2016.10.024
2. Applegate, D.L., Bixby, R.E., Chvátal, V., and Cook, W.J., The Traveling Salesman Problem: A Computational
Study, Princeton Univ. Press, 2006.
3. Arthanari, T.S., On pedigree polytopes and Hamiltonian cycles, Discrete Math., 2006, vol. 306, no. 14,
pp. 1474–1492.
https://doi.org/10.1016/j.disc.2005.11.030
4. Arthanari, T.S., Study of the pedigree polytope and a sufficiency condition for nonadjacency in the tour polytope, *Discrete Optim.*, 2013, vol. 10, no. 3, pp. 224–232. https://doi.org/10.1016/j.disopt.2013.07.001

5. Bae, M.M. and Bose, M., Edge disjoint Hamiltonian cycles in $k$-ary $n$-cubes and hypercubes, *IEEE Trans. Comput.*, 2003, vol. 52, no. 10, pp. 1271–1284. https://doi.org/10.1109/TC.2003.1234525

6. Bailey, R.F., Error-correcting codes from permutation groups, *Discrete Math.*, 2009, vol. 309, no. 13, pp. 4253–4265. https://doi.org/10.1016/j.disc.2008.12.027

7. Balinski, M.L., Signature methods for the assignment problem, *Oper. Res.*, 1985, vol. 33, no. 3, pp. 527–536. https://doi.org/10.1287/opre.33.3.527

8. Bondarenko, V. and Nikolaev, A., On graphs of the cone decompositions for the min-cut and max-cut problems, *Int. J. Math. Math. Sci.*, 2016, vol. 2016, p. 786350. https://doi.org/10.1155/2016/786350

9. Bondarenko, V.A., Nonpolynomial lower bounds for the complexity of the traveling salesman problem in a class of algorithms, *Autom. Remote Control*, 1983, vol. 44, no. 9, pp. 1137–1142.

10. Bondarenko, V.A. and Nikolaev, A.V., On the skeleton of the polytope of pyramidal tours, *J. Appl. Ind. Math.*, 2018, vol. 12, no. 1, pp. 9–18. https://doi.org/10.1134/S1990478918010027

11. Chegireddy, C.R. and Hamacher, H.W., Algorithms for finding $K$-best perfect matchings, *Discrete Appl. Math.*, 1987, vol. 18, no. 2, pp. 155–165. https://doi.org/10.1016/0166-218X(87)90017-5

12. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., and Zhu, M.Y., Tools for privacy preserving distributed data mining, *ACM SIGKDD Explor. Newsl.*, 2002, vol. 4, no. 2, pp. 28–34. https://doi.org/10.1145/772862.772867

13. Combarro, E.F. and Miranda, P., Adjacency on the order polytope with applications to the theory of fuzzy measures, *Fuzzy Sets Syst.*, 2010, vol. 161, no. 5, pp. 619–641. https://doi.org/10.1016/j.fss.2009.05.004

14. Dantzig, G., Fulkerson, R., and Johnson, S., Solution of a large-scale traveling-salesman problem, *J. Oper. Res. Soc. Am.*, 1954, vol. 2, no. 4, pp. 393–410. https://doi.org/10.2307/166695

15. Gabow, H.N., Two algorithms for generating weighted spanning trees in order, *SIAM J. Comput.*, 1977, vol. 6, no. 1, pp. 139–150. https://doi.org/10.1137/1006011

16. Glebov, R., Luria, Z., and Sudakov, B., The number of Hamiltonian decompositions of regular graphs, *Israel J. Math.*, 2017, vol. 222, no. 1, pp. 91–108. https://doi.org/10.1007/s11856-017-1583-y

17. Grötschel, M. and Padberg, M., Polyhedral theory, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B., Eds., Chichester: John Wiley, 1985, pp. 251–305.

18. Hung, R.W., Embedding two edge-disjoint Hamiltonian cycles into locally twisted cubes, *Theor. Comput. Sci.*, 2011, vol. 412, no. 35, pp. 4747–4753. https://doi.org/10.1016/j.tcs.2011.05.004

19. Kim, J.H. and Wormald, N.C., Random matchings which induce Hamilton cycles and hamiltonian decompositions of random regular graphs, *J. Combinatorial Theory, Ser. B*, 2001, vol. 81, no. 1, pp. 20–44. https://doi.org/10.1006/jctb.2000.1991

20. Knuth, D.E., *The Art of Computer Programming*, vol. 2: Seminumerical Algorithms, Addison-Wesley, 1997, 3rd ed. https://doi.org/10.5555/270146

21. Korostil, A.V. and Nikolaev, A.V., Backtracking algorithm to construct the Hamiltonian decomposition of a 4-regular multigraph, *Zam. Inf. Mat.*, 2020, vol. 12, pp. 91–97.

22. Kozlova, A. and Nikolaev, A., Simulated annealing approach to verify vertex adjacencies in the traveling salesperson polytope, *Mathematical Optimization Theory and Operations Research. MOTOR 2019*, Khachay, M., Kochetov, Y., and Pardalos, P., Lecture Notes in Computer Science, vol. 11548, Cham: Springer, 2019, pp. 374–389. https://doi.org/10.1007/978-3-030-22629-9_26

23. Krarup, J., The peripatetic salesman and some related unsolved problems, *Combinatorial Programming: Methods and Applications*, Roy, B., Eds., NATO Advanced Study Institutes Series, vol. 19, Dordrecht: Springer, 1995, pp. 173–178. https://doi.org/10.1007/978-94-011-7557-9_8
24. Nikolaev, A., On vertex adjacencies in the polytope of pyramidal tours with step-backs, Mathematical Optimization Theory and Operations Research. MOTOR 2019, Khachay, M., Kochetov, Y., and Pardalos, P., Lecture Notes in Computer Science, vol. 11548, Cham: Springer, 2019, pp. 247–263. https://doi.org/10.1007/978-3-030-22629-9_18

25. Nikolaev, A. and Kozlova, A., Hamiltonian decomposition and verifying vertex adjacency in 1-skeleton of the traveling salesperson polytope by variable neighborhood search, J. Combinatorial Optim., 2021, vol. 42, no. 2, pp. 212–230. https://doi.org/10.1007/s10878-020-00652-7

26. Papadimitriou, C.H., The adjacency relation on the traveling salesman polytope is NP-complete, Math. Program., 1978, vol. 14, no. 1, pp. 312–324. https://doi.org/10.1007/BF01588973

27. Péroche, B., NP-completeness of some problems of partitioning and covering in graphs, Discrete Appl. Math., 1984, vol. 8, no. 2, pp. 195–208. https://doi.org/10.1016/0166-218X(84)90101-X

28. Rao, M.R., Adjacency of the traveling salesman tours and $0 - 1$ vertices, SIAM J. Appl. Math., 1976, vol. 30, no. 2, pp. 191–198. https://doi.org/10.1137/0130021

29. Skiena, S.S., The Algorithm Design Manual, London: Springer, 2008, 2nd ed. https://doi.org/10.1007/978-1-84800-070-4