Hindawi
Journal of Optimization
Volume 2018, Article ID 9328103, 10 pages
https://doi.org/10.1155/2018/9328103

Research Article

HybridHAM: A Novel Hybrid Heuristic for Finding Hamiltonian Cycle

K. R. Seeja

Department of Computer Science & Engineering, Indira Gandhi Delhi Technical University for Women, Kashmere Gate, Delhi 110006, India

Correspondence should be addressed to K. R. Seeja; krseeja@gmail.com

Received 9 May 2018; Revised 7 September 2018; Accepted 25 September 2018; Published 16 October 2018

Academic Editor: Jun He

Hamiltonian Cycle Problem is one of the most explored combinatorial problems. Being an NP-complete problem, heuristic approaches are found to be more powerful than exponential time exact algorithms. This paper presents an efficient hybrid heuristic that sits in between the complex reliable approaches and simple faster approaches. The proposed algorithm is a combination of greedy, rotational transformation and unreachable vertex heuristics that works in three phases. In the first phase, an initial path is created by using greedy depth first search. This initial path is then extended to a Hamiltonian path in second phase by using rotational transformation and greedy depth first search. Third phase converts the Hamiltonian path into a Hamiltonian cycle by using rotational transformation. The proposed approach could find Hamiltonian cycles from a set of hard graphs collected from the literature, all the Hamiltonian instances (1000 to 5000 vertices) given in TSPLIB, and some instances of FHCP Challenge Set. Moreover, the algorithm has $O(n^3)$ worst case time complexity. The performance of the algorithm has been compared with the state-of-the-art algorithms and it was found that HybridHAM outperforms others in terms of running time.

1. Introduction

The Hamiltonian Cycle Problem (HCP) is to identify a cycle in an undirected graph connecting all the vertices in the graph. It is considered as a subproblem of the most popular NP-complete problem, the Travelling Salesman Problem (TSP), where the problem is to find the minimum weighted Hamiltonian cycle. Hamiltonian cycles have many applications like reconstructing genome sequences, solving games like Icosian game, finding a knight's tour on a chessboard, and finding circular embeddings for regular graphs. There is no single efficient algorithm for this problem till date. The state-of-the-art algorithms are mainly classified into two: exponential time exhaustive search algorithms and polynomial time heuristic algorithms. While the first category guarantees giving solution, the latter does not. The latter gives solution in sufficiently less time in most of the cases compared to the first. The algorithms in the first category generally find some efficient pruning rules for reducing the search space and thus improving the running time, while those in the second category find some general thump rules for finding the solution in as many problem instances as possible, in less time. The objective of this research is to design a heuristic which is quicker than the established sophisticated heuristics, but which is more reliable than the very fast techniques.

Many theorems can be found in literature, giving the necessary and sufficient conditions [1–4] for Hamiltonian cycle. These conditions are used for checking whether the graph is Hamiltonian or not. A good study [5] on these theorems and algorithms for solving HCP is given by Vandegriend & Basil. Rubin & Frank [6] proposed an exhaustive search method for finding all Hamiltonian paths or cycles in a directed or undirected graph. Christophides [7] proposed a multipath algorithm that was again an intelligent search algorithm with exponential complexity. Christophides algorithm has been improved by Kocay & William [8] by proposing two operations for pruning the search space. Martello [9] proposed a backtrack search algorithm that uses low degree first heuristic for selecting the next vertex. Ejov et al [10] solved HCP by solving equations drawn from the adjacent matrix of the graph. Even though they could find long cycles in case of non-Hamiltonian graphs, they could

not reduce the exponential time complexity. Posa's algorithm [11] is considered as the base algorithm for the heuristic algorithms for HCP. Posa's idea of rotational transformation and its variants formed the basis of almost all heuristic algorithms proposed later. Angluin and Valiant [12] proposed a much more complicated transformation for directed graph, as the rotational transformation is not suitable for directed graph. Bollobas et al. [13] proposed a Hamiltonian cycle algorithm called HAM that uses rotational transformation and cycle extension. Various versions of HAM algorithm like SparseHam [14] and HideHam [15] are also proposed for different kinds of graphs. Brunacci [16] proposed two algorithms DB2 and DB2A, which consider the HCP as a version of TSP and the nonedges as highly weighted edges. DB2A algorithm is a modification of DBA algorithm for directed graphs where directed graph is transformed into undirected graph. Many TSP heuristics like the famous Lin-Kernighan heuristic [17, 18] use a technique called "k-opt" transformation [19, 20], which is an exchange of k edges. Baniasadi et al. [21] proposed a "snakes and ladders" heuristic for solving HCP inspired from k-opt transformations. There are very few published HCP heuristics that sit in the "middle" area between sophisticated reliable heuristics and very simplistic (usually linear or quadratic time) approaches.

The proposed heuristic is a combination of greedy depth first search, unreachable vertex, and rotational transformation heuristics. The greedy depth first search reduces the running time considerably. The search is greedy since it always selects the unvisited low degree vertex for extending the path. While the unreachable vertex heuristic reduces the chances of reaching dead end, the rotational transformation helps to come out of the dead end condition. Thus the proposed method is faster than the complex exact algorithms and more reliable than the faster heuristics.

2. Materials and Methods

2.1. Hamiltonian Cycle Problem. Hamiltonian cycle is a cycle connecting all the vertices in a given graph only once. A graph containing at least one Hamiltonian cycle is called Hamiltonian graph. This optimization problem can be formally defined as follows:

Given a graph G=(E,V), where E is the set of edges of the graph, V is the set of vertices of the graph and |V| = n.

The problem is to find a cycle, HC=(v_1,v_2,.....,v_n), where all (v_i, v_{i+1}) and (v_1, v_n) are elements of E.

It is a hard problem that attracted both mathematicians and computer scientists. It is considered as one of the strong representatives of the NP-complete problem.

2.2. Greedy Depth First Search. The large search space of the HCP can be explored either breadth wise or depth wise. In depth first search, search is performed in depth wise manner starting from a given vertex till the point where the search cannot proceed further or a dead end is reached. The proposed heuristic uses greedy depth first search to create a Hamiltonian path. The path construction starts from the highest degree vertex because it increases the probability of returning to the starting vertex. Degree of a vertex is the number of edges connected to that vertex. The other vertices of the path are selected greedily by selecting the smallest degree neighbour among the unvisited neighbours. The lowest degree vertices are added first into the path since they may become unreachable with the selection of its neighbours. For example, if the degree of a vertex is two then both the edges of that vertex must be present in the Hamiltonian path/cycle and these edges are added to the path first. The vertices that are part of the path constructed so far are considered as "visited." Since in Hamiltonian cycle a vertex appears only once, only the unvisited neighbours of the current vertex need to be considered. In order to guide the greedy depth first search further to longer paths, an unreachable vertex heuristic is proposed. According to this heuristic, before adding a vertex to the path, an unreachable condition is checked for each of its neighbours. This heuristic is explained in Section 2.3.

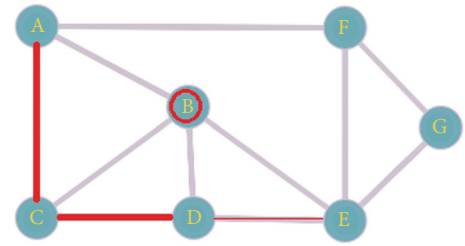

Figure 1: Unreachable vertex.

2.3. Unreachable Vertex Heuristic. This research proposes a new heuristic, namely, unreachable vertex heuristic, to reduce the chance of reaching a dead end while constructing the Hamiltonian path.

Definition 1. Let P be the partial path and x be the end vertex on partial path. Select the next vertex y∈Adj(x) in the path such that the number of unvisited adjacent vertices of all Adj(y) vertices is greater than one.

A vertex is said to be unreachable if all its neighbours are already a part of the path constructed so far. In this case, there is no way to reach the vertex and it becomes unreachable. In this proposed heuristic, if the selection of a vertex is making any of its unvisited neighbours unreachable then that vertex will not be added to path.

For example, consider the graph shown in Figure 1.

Let the partial path identified so far be A⟶C⟶D. In the given graph Adj(D) = {B, E}. Let the next vertex selected be E. Adj(E) = {B, G, F}. Number of unvisited neighbours of B, G and F are 1, 2 and 2 respectively. According to the unreachable vertex heuristic, E will not be selected as next vertex in the path since the number of unvisited neighbours of B is one.

2.4. Rotational Transformation. Rotational transformation [11] and its variations are found to be powerful heuristics for finding Hamiltonian cycle. It is used to change a path to get a new end vertex as shown in Figure 2. In the figure, e is the end

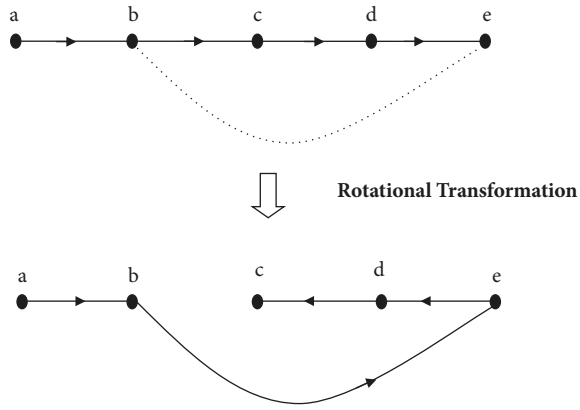

Figure 2: Rotational transformation.

on which rotational transformation is applied to get a new end vertex.

The procedure of rotational transformation is summarized below.

(1) Find a vertex adjacent to e in the input graph, in path P. Let it be vertex b.

(2) Create a new path P',

  (a) by connecting b to e
  (b) by reversing the path from c to e

In the proposed algorithm, rotational transformation is used for two purposes.

(1) To come out of the dead end during the construction of Hamiltonian path.

(2) To convert the Hamiltonian path into Hamiltonian cycle.

In the first case highest degree end of the path is selected for rotational transformation since it increases the probability of getting a new end. Similarly for converting path into cycle, the lowest degree end vertex of the path is selected for rotational transformation since it increases the probability of returning to the higher degree end to make the cycle.

3. Proposed Hybrid Heuristic

3.1. HybridHAM Algorithm. The proposed HybridHAM algorithm works in three steps:

(1) Create an initial path

(2) Convert the initial path to Hamiltonian path.

(3) Convert the Hamiltonian path to Hamiltonian cycle.

3.1.1. Initial Path Creation. The path creation starts from the highest degree vertex and then adds the smallest degree vertices greedily in order to construct an initial path as long as possible. The selection of highest degree initial vertex and then the smallest degree vertices is meant for constructing a longer initial path and is given in Table 1. The algorithm checks the unreachable condition before adding a vertex in the path. This is to reduce the probability of reaching a dead end during the path construction. If there are more than one highest degree vertices then repeat this procedure of creating initial path by selecting each one of them as the starting vertex and select the path with maximum number of vertices as the initial path. By following this procedure, we may get an initial path which is Hamiltonian or near Hamiltonian.

3.1.2. Conversion of Initial Path into Hamiltonian Path. If the initial path created is not Hamiltonian (less number of vertices than the total number of vertices), then select the highest degree end of the initial path for rotational transformation. This is to increase the probability for getting a new end vertex for extending the path further to create a Hamiltonian path. Extend the path from the new end vertex by following the greedy procedure in Section 3.1.1. The rotational transformation and greedy path extension are continued until we are getting a Hamiltonian path. At any time during this process, rotational transformation could not be applied means that either the graph is not having Hamiltonian path or the algorithm fails to identify the Hamiltonian path.

3.1.3. Conversion of Hamiltonian Path into Hamiltonian Cycle. If there is an edge connecting the two ends of the Hamiltonian path in the graph then add that edge to the path to get the Hamiltonian cycle. Otherwise, apply rotational transformation to the smallest degree vertex repeatedly until getting a new end vertex which can be connected to the other end vertex to form the Hamiltonian cycle. At any time during this process, rotational transformation could not be applied, means that either the graph is not having Hamiltonian cycle or the algorithm fails to identify the Hamiltonian cycle.

The vertex selection criteria used in various steps are summarized in Table 1.

3.2. Example. Consider the undirected graph in Figure 3.

3.2.1. Initial Path Identified in Step 1

0 1 2 3 4 5 6 7 8 12 13 14 22 21 20 19 18 15

Here the length of the initial path is 18, which is less than the total number of vertices in the graph. Since the identified path is not Hamiltonian, go to step 2.

3.2.2. Hamiltonian Path Identified in Step 2

0 7 8 12 14 13 15 17 16 9 1 2 3 4 5 6 11 23 22 21 20 18 19 10

Here the length of the path is 24 and hence it is Hamiltonian path. Since there is no edge in Figure 3, connecting the end vertices of the path (i.e., edge connecting vertex 1 and vertex 11), go to step 3 and apply rotational transformation to find the new end vertices that are connected in the graph in Figure 3

Table 1: Vertex selection criteria.

Vertex Selection Rule	Step	Reason
Selection of Highest degree initial vertex.	Step 1	It increases the probability of getting a path to reach back to this vertex to form the cycle.
Greedy selection of smallest degree vertices during path construction.	Step 1	Giving preference to smaller degree vertices increases the probability of longer paths. For example, if the degree of a vertex adjacent to the current end vertex of the path is two, then it should be included first in the path, as it is the only possible position of the vertex in the Hamiltonian path.
Selection of the highest degree end of the initial path for rotational transformation	Step 2	It increases the probability for getting a new end vertex for extending the path further to create a Hamiltonian path
Selection of the smallest degree end vertex for rotational transformation	Step 3	This is done to keep the highest degree end vertex, since the probability of getting an edge connecting a higher degree vertex is more compared to a smaller degree vertex.

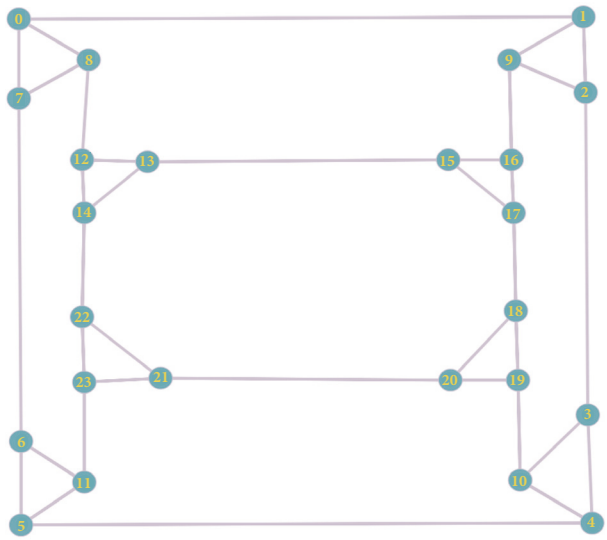

Figure 3: Input undirected graph.

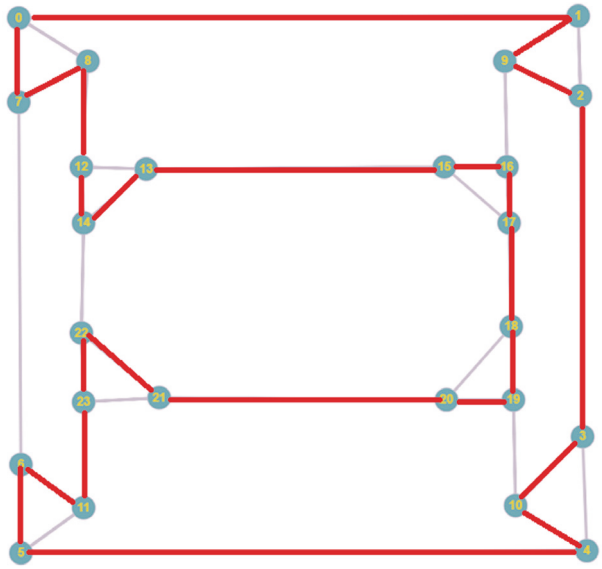

Figure 4: Hamiltonian cycle.

3.2.3. Hamiltonian Cycle Identified in Step 3

0 7 8 12 14 13 15 16 17 18 19 20 21 22 23 11 6 5 4 10 3 2 9 1

Here the end vertices 1 and 2 are connected in the initial graph shown in Figure 3 and hence form the Hamiltonian cycle. It is shown in Figure 4.

3.3. Pseudocode. The input to the algorithm is the adjacency matrix representation of the graph and the output is the set of vertices corresponding to the Hamiltonian cycle. Let n be the number of vertices in the graph. The proposed algorithm contains three phases and is outlined as follows:

HybridHAM()
{

From the input adjacency matrix, create two arrays V_a and V_d of vertices sorted in the increasing and decreasing order of their degrees, respectively.

//Phase 1
//Create an initial path

(1) Start from one of the highest degree vertex (first vertex in the array V_d). Let it be v_s.

(2) Add it to the initial path P_i.

(3) Repeat

(a) Select the next smallest degree neighbour of v_s (from the array V_a). Let it be v_i.

(b) If the selection of v_i is not making any of its unvisited neighbours unreachable then

(i) add it to the initial path P_i

(ii) make v_i as v_s

Until v_s becomes a dead end.

//End of Phase 1

(4) If $|P_i| = n$ then go to Phase 3.

Else

Repeat Phase 1 for each of the highest degree vertex of the graph and select the

longest P_i as initial path.

End if

//Phase 2
//Convert the initial path into Hamiltonian path

(5) Repeat

(a) Select the highest degree end of the path P_i for rotational transformation

(b) Reverse P_i if the first vertex in P_i is having degree higher than the last vertex to make the highest degree vertex as the end vertex of the path. Let it be v_x.

(c) Apply rotational transformation to P_i using v_x to get a new path.

(d) If rotational transformation could not be applied then either the graph is not having Hamiltonian path or the algorithm fails to identify the Hamiltonian path and so exit.

(e) Extend this new path by using the greedy depth first search as in Phase 1.

Until $|P_i| = n$.

(6) Now P_i is the Hamiltonian Path P_h. Assign $P_h = P_i$.

//End of Phase 2

(7) If there is an edge connecting the first and last vertices of P_h in the graph then

P_h is the Hamiltonian cycle and return P_h

else

go to Phase 3

End if

//Phase 3
//Convert Hamiltonian path into Hamiltonian cycle

(8) Repeat

(a) Select the smallest degree end of the path P_h for rotational transformation

(b) Reverse P_h if the first vertex in P_h is having degree higher than the last vertex to make the smallest degree vertex as the end vertex of the path. Let it be v_y.

(c) Apply rotational transformation to P_h using v_y to get a new path.

(d) If rotational transformation could not be applied then either the graph is not having Hamiltonian Cycle or the algorithm fails to identify the Hamiltonian path and so exit.

Until there is an edge connecting the first and last vertices of the path P_h.

(9) Now P_h is the Hamiltonian cycle and return P_h.

//End of Phase 3
}// End of HybridHAM algorithm

3.4. Worst Case Complexity

(1) *Creation of sorted arrays*

$T(n) = O(n^2)$

(2) *Phase 1*

Greedy depth first search goes through the adjacency matrix once in the worst case and hence its complexity is $O(n^2)$.

This search is repeated for each of the highest degree vertex. The worst case is all the n vertices are of the same degree.

Therefore, $T(n) = O(n^3)$

(3) *Phase 2*

Complexity of rotational transformation at the most is $O(n)$

Complexity of greedy depth first search is $O(n^2)$

Total of these two = $O(n^2)$

These two operations are repeated at most n times.

Therefore, $T(n) = O(n^3)$

(4) Phase 3

Complexity of rotational transformation at the most is $O(n)$

This operation is repeated at most n times.

Therefore, $T(n) = O(n^2)$

Total worst case complexity of HybridHam, $T(n) = O(n^2) + O(n^3) + O(n^3) + O(n^2) = \boldsymbol{O(n^3)}$.

4. Experiments

The performance evaluation experiments of the proposed algorithm are conducted in a system with 4GB RAM and Intel Core i5 processor. The algorithm is implemented in MATLAB 13RA. The experiments were carried out on a set of graphs collected from the literature as well as from the TSPLIB.

4.1. Sample Hamiltonian Cycle Graphs. The Hamiltonian cycles returned by the proposed algorithm on the collected set of sample graphs are shown in Figure 5.

The running time of the algorithm to solve these sample instances is given in Table 2.

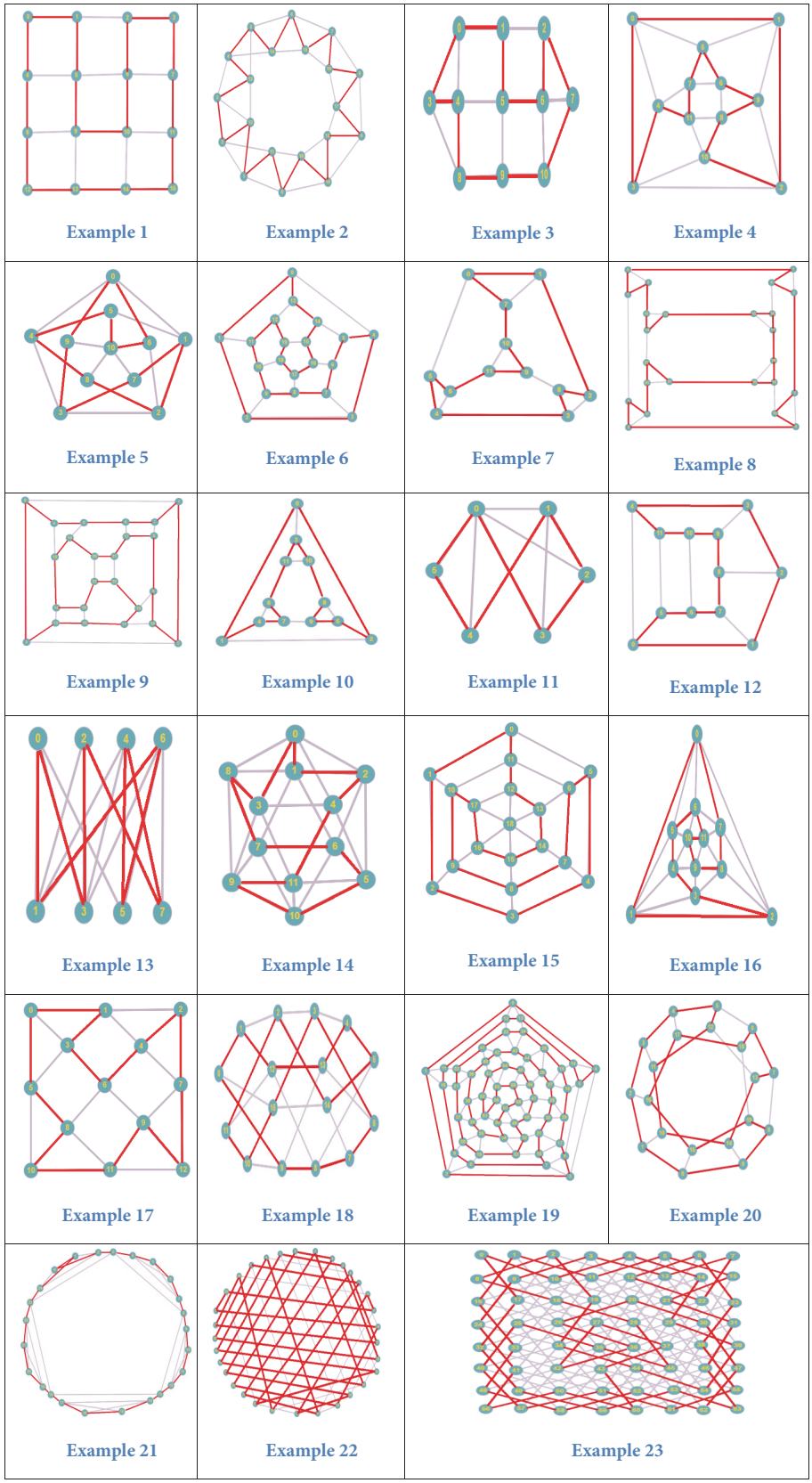

Figure 5: Sample Hamiltonian graphs.

Table 2: Running time on sample instances.

Sl.No	Name	No. of Vertices	Running time in Sec.
1.	Example 1	16	0.051
2.	Example 2	20	0.056
3.	Example 3	11	0.051
4.	Example 4	12	0.054
5.	Example 5	11	0.051
6.	Example 6	20	0.049
7.	Example 7	12	0.052
8.	Example 8	24	0.083
9.	Example 9	24	0.063
10.	Example 10	12	0.051
11.	Example 11	6	0.049
12.	Example 12	12	0.031
13.	Example 13	8	0.029
14.	Example 14	12	0.034
15.	Example 15	19	0.030
16.	Example 16	12	0.032
17.	Example 17	13	0.030
18.	Example 18	16	0.038
19.	Example 19	60	0.043
20.	Example 20	20	0.042
21.	Example 21	25	0.031
22.	Example 22	32	0.032
23.	Example 23	64	0.078

Table 3: Running time on TSPLIB instances.

Sl.No	Name	No. of vertices	Running time in sec.			
			Concorde	HCP Solver	Snake and Ladder Heuristic	Hybrid Ham
1.	Alb1000	1000	4.95	0.72	0.1	0.2656
2.	Alb2000	2000	7.30	3.29	0.8	1.4375
3.	Alb3000a	3000	9.56	8.36	3.44	2.7656
4.	Alb3000b	3000	9.94	8.02	3.64	1.5781
5.	Alb3000c	3000	9.95	8.43	4.31	1.8438
6.	Alb3000d	3000	10.14	8.48	4.03	1.6406
7.	Alb3000e	3000	10.44	8.03	4.29	1.6719
8.	Alb4000	4000	13.45	17.84	13.89	3.0625
9.	Alb5000	5000	17.24	30.85	14.12	8.9844

4.2. TSPLIB Instances. The TSPLIB library [22] contains seven Hamiltonian cycle instances of 1000, 2000, 3000, and 5000 vertices. The proposed algorithm could solve all these problem instances in a few seconds. The running time of the proposed algorithm is compared with that of *HCP Solver [23]*, *Concorde TSP Solver [24]*, and the latest *Snake and Ladder Heuristic [21]*. Table 3 gives the time taken by these algorithms to solve each of the HCP instances.

4.3. FHCP Challenge Set. The FHCP Challenge Set [25] is a collection of 1001 Hamiltonian Cycle Problem instances. This dataset is specifically designed to resist the heuristic approaches. HybridHAM is tested on 250 instances of the FHCP Challenge Set. Out of these 250 instances, the proposed heuristic could find 75 Hamiltonian paths and 6 Hamiltonian cycles. The complete results are shown in Table 4. The graphs are too sparse so that the rotation transformation could not convert the Hamiltonian path into Hamiltonian cycle in Phase 3 of the algorithm. Phase 1 and Phase 2 of the algorithm perform comparatively well with the highly difficult instances of the Challenge Set and hence the Hamiltonian path. It is also found that the proposed highest degree

Table 4: Result on FHCP Challenge Set.

Sl.No	Name	No. of Vertices	No. of Edges	Output	Running Time in Sec.
1.	Graph 1	66	99	HP	0.0625
2.	Graph 2	70	106	HC	0.0469
3.	Graph 3	78	117	HP	0.0625
4.	Graph 4	84	127	HP	0.0625
5.	Graph5	90	135	HP	0.0625
6.	Graph 6	94	142	HC	0.0625
7.	Graph 7	102	153	HP	0.0781
8.	Graph8	108	162	HP	0.0625
9.	Graph 9	114	171	HP	0.0938
10.	Graph 11	126	189	HP	0.1094
11.	Graph 12	132	199	HP	0.1094
12.	Graph 14	142	214	HP	0.0938
13.	Graph 15	150	225	HP	0.1250
14.	Graph 16	156	235	HP	0.0938
15.	Graph 17	162	243	HP	0.1875
16.	Graph 18	166	250	HP	0.1094
17.	Graph 20	174	261	HP	0.2344
18.	Graph 21	180	271	HP	0.1406
19.	Graph 22	186	279	HP	0.2188
20.	Graph 23	190	286	HP	0.1563
21.	Graph 25	204	307	HP	0.1719
22.	Graph 26	210	315	HP	0.3750
23.	Graph 27	214	322	HP	0.2031
24.	Graph 29	228	343	HP	0.2500
25.	Graph32	246	369	HP	0.4063
26.	Graph 33	252	379	HP	0.3281
27.	Graph34	258	387	HP	0.4688
28.	Graph 35	262	394	HP	0.5781
29.	Graph 36	270	405	HP	0.7031
30.	Graph 37	276	415	HP	0.3750
31.	Graph 40	294	441	HP	1.0469
32.	Graph 41	300	451	HP	0.4844
33.	Graph 43	310	466	HP	0.7031
34.	Graph 44	312	477	HP	0.9844
35.	Graph 45	324	487	HP	0.6094
36.	Graph 50	348	523	HP	0.9063
37.	Graph 53	366	549	HP	1.6875
38.	Graph 54	372	559	HP	0.9219
39.	Graph 58	396	595	HP	1.7031
40.	Graph 59	400	40001	HC	0.2656
41.	Graph 64	416	625	HP	1.0938
42.	Graph 65	419	631	HP	1.4375
43.	Graph 68	438	657	HP	2.9219
44.	Graph69	444	667	HP	2.9063
45.	Graph 72	460	52901	HC	0.4375
46.	Graph 79	480	57601	HC	0.4844
47.	Graph 82	496	745	HP	1.7031

Table 4: Continued.

Sl.No	Name	No. of Vertices	No. of Edges	Output	Running Time in Sec.
48.	Graph 84	500	62501	HC	0.5313
49.	Graph 90	510	65026	HC	0.5625
50.	Graph 91	516	775	HP	3.5156
51.	Graph 95	540	811	HP	3.0469
52.	Graph 96	540	72901	HC	0.7031
53.	Graph 99	550	826	HP	5.6719
54.	Graph 104	576	865	HP	2.8594
55.	Graph 118	636	955	HP	6.8594
56.	Graph 122	656	985	HP	4.1563
57.	Graph 124	660	991	HP	8.2813
58.	Graph 128	677	114583	HC	1.3594
59.	Graph 134	724	131045	HC	1.5313
60.	Graph 137	736	1105	HP	12.9531
61.	Graph 148	816	1225	HP	7.9531
62.	Graph 150	823	169333	HC	2.7500
63.	Graph 151	828	1243	HP	11.5625
64.	Graph 160	896	1345	HP	14
65.	Graph 162	909	206571	HC	4.0625
66.	Graph 168	972	1459	HP	33.0938
67.	Graph 169	976	1465	HP	28.6406
68.	Graph 176	1020	1531	HP	29.4375
69.	Graph 182	1056	1585	HP	22.6719
70.	Graph 188	1123	315283	HC	9.6406
71.	Graph 190	1136	1705	HP	14.4844
72.	Graph 203	1216	1825	HP	40.2813
73.	Graph 211	1296	1945	HP	40.1406
74.	Graph 233	1456	2185	HP	85.4688
75.	Graph 246	1536	2305	HP	111.8438

and smallest degree vertex selection criteria suit more graphs having vertices of different degrees. However, the algorithm could find Hamiltonian cycle from medium sparse graphs in very less time (e.g., Graphs 72, 79, 84, 90, etc. in Table 4). Even the winners [26] of the challenge could not solve the complete set and they used different algorithms for different types of graphs as their objective was to find solutions rather than developing algorithms.

It would be worth pointing out that the difficult instances of FHCP challenge dataset are very rare and difficult to construct, let alone encounter "naturally." Also even discovering a Hamiltonian path is a difficult (NP-complete) problem and the proposed heuristic succeeded in doing so for many instances of the FHCP Challenge Set.

5. Conclusion

This paper proposes a hybrid heuristic for finding Hamiltonian cycle from undirected graphs. The proposed three-stage algorithm uses a combination of three heuristics: greedy depth first search, unreachable vertex, and rotational transformation. From the various experiments conducted on different graphs of variable sizes and complexity, it is found that the highest degree heuristic used for selecting the initial vertex for the creation of longest initial path as well as Hamiltonian path and the smallest degree heuristic used for the conversion of Hamiltonian path to Hamiltonian cycle are the reasons for getting the solution in a single run. The experimental evaluation also shows that the proposed heuristic is much faster and succeeds in obtaining good results in the majority of cases. It should be worth noting that the increase in speed comes without sacrificing the heuristic's reliability on "easy" instances of various sizes (including the quite large TSPLIB instances) and that it still performs reasonably well on more difficult instances.

Data Availability

The data supporting this research are from previously reported studies and datasets, which have been cited. The processed data are available in [22, 25].

Conflicts of Interest

The author declares that there is no conflict of interest.

Acknowledgments

The research presented in this manuscript is supported by University Grants Commission Major Project Grant F.No.42-136/2013(SR).

References

[1] G. A. Dirac, "Some theorems on abstract graphs," *Proceedings of the London Mathematical Society. Third Series*, vol. 2, pp. 69–81, 1952.

[2] O. Ore, "Note on Hamilton circuits," *The American Mathematical Monthly*, vol. 67, 55 pages, 1960.

[3] G. Fan, "New sufficient conditions for cycles in graphs," *Journal of Combinatorial Theory, Series B*, vol. 37, no. 3, pp. 221–227, 1984.

[4] R. J. Gould, "Advances on the Hamiltonian problem—a survey," *Graphs and Combinatorics*, vol. 19, no. 1, pp. 7–52, 2003.

[5] Vandergriend, *Basil (1998), Finding Hamiltonian Cycles: Algorithms, graphs and performance [M.sc. Thesis]*, Faculty of Graduate Studies and Research, University of Alberta, 1998.

[6] F. Rubin, "A search procedure for Hamilton paths and circuits," *Journal of the ACM*, vol. 21, pp. 576–580, 1974.

[7] N. Christofides, *Graph Theory: An Algorithmic Approach, Computer Science and Applied Mathematics*, Academic Press, New York, NY, USA, 1975.

[8] W. Kocay, "An extension of the multi-path algorithm for finding Hamilton cycles," *Discrete Mathematics*, vol. 101, no. 1-3, pp. 171–188, 1992.

[9] S. Martello, "Algorithm 595: An Enumerative Algorithm for Finding Hamiltonian Circuits in a Directed Graph," *ACM Transactions on Mathematical Software*, vol. 9, no. 1, pp. 131–138, 1983.

[10] V. Ejov, J. A. Filar, S. K. Lucas, and J. L. Nelson, "Solving the Hamiltonian Cycle problem using symbolic determinants," *Taiwanese Journal of Mathematics*, vol. 10, no. 2, pp. 327–338, 2006.

[11] L. Pósa, "Hamiltonian circuits in random graphs," *Discrete Mathematics*, vol. 14, no. 4, pp. 359–364, 1976.

[12] D. Angluin and L. G. Valiant, "Fast probabilistic algorithms for Hamiltonian circuits and matchings," in *Proceedings of the ninth annual ACM symposium on Theory of computing*, pp. 30–41, ACM, 1977.

[13] B. Bollobás, T. I. Fenner, and A. M. Frieze, "An algorithm for finding hamilton paths and cycles in random graphs," *Combinatorics, Probability and Computing*, vol. 7, no. 4, pp. 327–341, 1987.

[14] A. M. Frieze, "An algorithm for finding Hamilton cycles in random directed graphs," *Journal of Algorithms*, vol. 9, no. 2, pp. 181–204, 1988.

[15] A. Z. Broder, A. M. Frieze, and E. Shamir, "Finding hidden hamiltonian cycles," *Random Structures & Algorithms*, vol. 5, no. 3, pp. 395–410, 1994.

[16] F. A. Brunacci, "DB2 and DB2A: Two useful tools for constructing Hamiltonian circuits," *European Journal of Operational Research*, vol. 34, no. 2, pp. 231–236, 1988.

[17] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, pp. 498–516, 1973.

[18] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[19] S. Lin, "Computer solutions of the traveling salesman problem," *Bell Labs Technical Journal*, vol. 44, pp. 2245–2269, 1965.

[20] M. M. Flood, "The traveling-salesman problem," *Operations Research*, vol. 4, pp. 61–75, 1956.

[21] P. Baniasadi, V. Ejov, J. A. Filar, M. Haythorpe, and S. Rossomakhine, "Deterministic "snakes and Ladders" Heuristic for the Hamiltonian cycle problem," *Mathematical Programming Computation*, vol. 6, no. 1, pp. 55–75, 2014.

[22] TSPLIB-Hamiltonian cycle problem (HCP), accessed on June 2013, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95.

[23] PCGRATE, accessed on September 2014, http://www.pcgrate.com/about/npcomprb/hcp.

[24] CONCORDE, accessed on September 2014, http://www3.cs.stonybrook.edu/~algorith/implement/concorde/implement.shtml.

[25] M. Haythorpe, "FHCP challenge set: the first set of structurally difficult instances of the hamiltonian cycle problem," *Bulletin of the Institute of Combinatorics and Its Applications*, vol. 83, pp. 98–107, 2018.

[26] A. Schneider, "When researchers play to solve 1001 instances of the Hamiltonian Cycle Problem," 2016, https://www.inria.fr/en/centre/sophia/news/when-researchers-play-to-solve-1001-instances-of-the-hamiltonian-cycle-problem.