Reduced-by-matching graphs: toward simplifying Hamiltonian circuit problem

Jacek Blazewicz Marta Kasprzak*

Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60-965 Poznan, Poland and Institute of Bioorganic Chemistry, Polish Academy of Sciences, Poznan, Poland

Abstract

The results presented in the paper are threefold. Firstly, a new class of reduced-by-matching directed graphs is defined and its properties studied. The graphs are output from the algorithm which, for a given 1-graph, removes arcs which are unnecessary from the point of view of searching for a Hamiltonian circuit. In the best case, the graph is reduced to a quasi-adjoint graph, what results in polynomial-time solution of the Hamiltonian circuit problem. Secondly, the systematization of several classes of digraphs, known from the literature and referring to directed line graphs, is provided together with the proof of its correctness. Finally, computational experiments are presented in order to verify the effectiveness of the reduction algorithm.

Keywords: Hamiltonian circuit problem, directed line graphs, quasi-adjoint graphs, graph reduction, digraphs

1 Introduction

The Hamiltonian circuit problem (HCP) is frequently studied in the scientific literature for its significance in both theoretical and practical branches of computer science. It is one of the most well known combinatorial problems, commonly used for modeling more complex and real-life problems. In light of this, its strong NP-completeness is an ongoing motivation for developing better and faster algorithms. Researchers have put a lot of effort in defining wider and wider classes of graphs being "easy" for the problem, together with dedicated polynomial-time exact algorithms. This research is often done autonomously for directed and undirected graphs, since polynomial-time solvability of HCP for one kind of graph does not imply that it is suitable for the other one. As an example we can give directed line graphs and line graphs: while the former

*The corresponding author: marta@cs.put.poznan.pl.

have polynomial-time exact algorithm solving HCP [8], HCP for the latter is strongly NP-complete [6]. Our work is focused on directed graphs and their classes containing or intersecting directed line graphs.

While many papers are concerned with classes of undirected graphs which are easy from the point of view of searching for HCP (see e.g. [21] for interval graphs, [15] for co-comparability graphs, [20] for distance-hereditary graphs, or [1] for some partly claw-free graphs), analogous classes of directed graphs have not been so widely studied. An example is the class of adjoints of other directed graphs, which strictly contains directed line graphs (e.g. see [5, 24]). HCP is easily solved in an adjoint by transforming the adjoint into its original graph and then by searching for an Eulerian circuit within it. The existence of an Eulerian circuit in the original directed graph is a necessary and sufficient condition of the existence of a Hamiltonian circuit in its adjoint [8]. Other papers on the topic define HCP as an easy problem for the classes of strongly connected tournaments [12], semicomplete multipartite digraphs [4] and other related graphs [3].

The paper by Blais and Laporte [7] deals with a transformation for directed, undirected, and mixed graphs, related to the generalized routing problem. The transformation replaces vertices, arcs and edges of the initial graph by vertices in a new complete weighted digraph, and then the (generalized) traveling salesman problem is solved. However, this approach preserves computational hardness of the problems before and after the transformation. This transformation is carried out inversely to the transformation for adjoints, where (in the latter) vertices are replaced by arcs.

In a recent paper by Blazewicz et al. [11] a new class of *quasi-adjoint graphs* was defined, which extends the set of known graph classes, for which HCP is polynomially solvable. The class of quasi-adjoint graphs is a generalization of, among others, adjoints and the graphs modeling the problem of isothermic DNA sequencing by hybridization without errors in experimental data [9]. The algorithm proposed in [11] tends to reduce a digraph into a quasi-adjoint graph by removing some arcs unnecessary from the point of view of searching for a Hamiltonian circuit. Even if the graph does not become a quasi-adjoint graph, it gets simpler and looking for a solution takes less time. None of feasible solutions of the Hamiltonian circuit problem is lost after this reduction.

In this paper, this theoretical algorithm is studied and the contribution is threefold. Firstly, the algorithm is slightly modified, the class of graphs on the output of the algorithm is defined and its properties studied. Secondly, the systematization of several classes of digraphs, known from the literature and referring to directed line graphs, is provided together with the proof of its correctness. Finally, the algorithm has been implemented and verified on a wide set of instances. The graphs reduced by the algorithm have been compared with their initial forms, for demonstrating the effectiveness of the algorithm. Two algorithms solving HCP known from the literature [23, 13] and two other simple algorithms have been used in the experiment in order to compare their performance for the graphs before and after the reduction.

The organization of the paper is as follows. Section 2 contains the description of the algorithm for graph reduction that simplifies the Hamiltonian circuit problem, and presents the resulting class of graphs. In Section 3 the systematization of digraph classes is presented. In Section 4 computational results are discussed. We conclude the paper in Section 5.

2 Reduction of graphs toward simplifying HCP

Throughout the paper we use a standard terminology from graph theory, see e.g. [5, 14]. *We are dealing with directed graphs* (digraphs), for which notions of interest are recalled below.

The *Hamiltonian circuit* is a circuit in a graph including every vertex exactly once. The *Eulerian circuit* is a circuit including every arc of a graph exactly once. The *1-graph* is a graph having, for all ordered pairs of vertices (x, y), at most one arc from vertex x to vertex y. A 1-graph can be represented as 0-1 *adjacency matrix* M, where $M[x, y] = 1$ means the existence of the arc from vertex x to vertex y. The *indegree* of vertex x (being the number of arcs entering x) is denoted by $d^-(x)$, the *outdegree* of x (being the number of arcs leaving x) by $d^+(x)$. In the following, using the term "successor" or "predecessor" we always mean the immediate one.

Definition 1. [5] The *adjoint* $G = (V, A)$ of a graph $H = (U, V)$ is a 1-graph whose vertices represent arcs of H, and which has an arc from x to y if the terminal endpoint of the arc in H corresponding to x is the initial endpoint of the arc corresponding to y.

The *directed line graph* is defined as an adjoint G of a 1-graph H.

Theorem 1. [5] *A 1-graph $G = (V, A)$ is an adjoint if and only if the following property is satisfied for all pairs $x, y \in V$:*

$$N^+(x) \cap N^+(y) \neq \emptyset \Rightarrow N^+(x) = N^+(y),$$

where $N^+(x)$ is the set of successors of vertex x.

Theorem 2. [8] *Let G be the adjoint of graph H. Then, there is an Eulerian path/circuit in H if and only if there is a Hamiltonian path/circuit in G.*

From Theorem 2 it follows that the problem of searching for a Hamiltonian circuit in a digraph, which is in general strongly NP-hard, becomes easy for adjoints. In [11] a wider class of graphs, for which the solution of HCP is polynomially solvable, was defined.

Theorem 3. [11] *A graph is a* quasi-adjoint graph *if, for any two vertices x and y, the following property holds:*

$$\begin{array}{lll} N^+(x) \cap N^+(y) \neq \emptyset & \Rightarrow & N^+(x) = N^+(y) \quad \vee \\ & & N^+(x) \subset N^+(y) \quad \vee \\ & & N^+(y) \subset N^+(x), \end{array}$$

where $N^+(x)$ is the set of successors of vertex x.

For the class of quasi-adjoint graphs a polynomial-time exact algorithm solving HCP was proposed in [11], which also used the rule of transforming an adjoint into its original graph.

The motivation of our work is to apply this rule to an enlarged class of graphs, which could be reduced into quasi-adjoint graphs by removing some superfluous arcs. The proposed algorithm (Algorithm 1) accepts as an input any 1-graph and attempts to reduce it through a series of arc removals, which are guaranteed not to belong to any feasible solution of the Hamiltonian circuit problem. It uses the concept of a cluster in an adjacency matrix of a 1-graph, defined below.

Definition 2. A *cluster* in an adjacency matrix M is a collection of rows and columns of M tied by 1s, both rows and columns of non-zero cardinality. In other words, a cluster can be identified in M by the successive addition to a selected non-zero row, alternately, these columns and rows of M which have 1s on the intersection with the rows/columns already belonging to the cluster.

A cluster can be interpreted as a bipartite graph $C = (V_{C1}, V_{C2}, A_C)$ with V_{C1} corresponding to the cluster's rows, V_{C2} corresponding to the cluster's columns, and A_C corresponding to the set of 1s at their intersection. One vertex may be present both in V_{C1} (as the tail of an arc) and V_{C2} (as the head of the same or another arc).

Algorithm 1

```
Input: An adjacency matrix $M$ of 1-graph $G$.
Output: A modified $M$ corresponding to the reduced 1-graph $G$.
(1) $\forall x$, $M[x,x] \leftarrow 0$;
(2) if there is any row or column of $M$ containing only 0s then exit;
(3) for every cluster identified in $M$ do
    {
        solve the problem of perfect matching in the bipartite graph $C$
            corresponding to the cluster;
        if there is no solution to this problem then exit;
        else
        {
            mark all arcs of $C$ composing the solution as "N"
                (Necessary);
            for every not yet marked arc $(x,y)$ of $C$ do
            {
                solve the problem of perfect matching in the bipartite
                    graph $C$ minus $x$, $y$, and minus all arcs incident
                    to $x$ or $y$;
                if there is no solution to this problem then $M[x,y] \leftarrow 0$
                    (the corresponding arc of $C$ disappears);
                else mark $(x,y)$ and all not yet marked arcs composing
                the solution as "N";
            }
        }
    }
```

Later, when we refer to "a perfect matching in a cluster" we mean a perfect matching in the bipartite graph corresponding to the cluster. Similarly, we denote cluster C like a bipartite graph, i.e. $C = (V_{C1}, V_{C2}, A_C)$.

Proposition 1. *If the instruction "exit" is performed in the algorithm, then no solution of HCP exists in the graph.*

Proof. The instruction "exit" in step (2) of Algorithm 1 is executed for graphs having at least one vertex without incoming/outgoing arcs. In step (3) it is executed for graphs having a cluster without any perfect matching. The latter means that there exists at least one vertex without a predecessor/successor in a circuit. From the definition of the Hamiltonian circuit it follows that such graphs have no feasible solution to this problem. □

Proposition 2. *Algorithm 1 reduces a 1-graph G without loss of any feasible solutions of the Hamiltonian circuit problem in this graph.*

Proof. The correctness of the previous version of this algorithm was proven in [11]. The current version, despite distinct notation, is logically the same except step (2) which does not remove any arcs. Recalling the arguments, the reduced arcs are either self-loops (in step (1)) or do not belong to any perfect matching in clusters (step (3)), thus they are not parts of any feasible solution of the Hamiltonian circuit problem. □

At the end of the algorithm if graph G becomes a quasi-adjoint graph, the solution for HCP can be found in polynomial time. Otherwise, the reduced graph becomes an easier instance for some (exact or heuristic) algorithm. The following example visualizes the former case when the reduced graph becomes an adjoint.

Example 1. Let graph G be defined as in Fig. 1A (it is not a quasi-adjoint graph). In the graph two clusters can be distinguished: the one composed only of arc (b, g) and the second one involving the remaining arcs. After applying Algorithm 1, the arcs unnecessary from the point of view of searching for a Hamiltonian circuit in the graph are removed (here 8 of the 20 arcs are removed). The algorithm will output the reduced graph shown in Fig. 1B (it is an adjoint). Now, HCP can be solved for this graph in polynomial time.

Let us now define the class of non-trivial graphs reduced by Algorithm 1. "Non-trivial" means here that it is not obvious whether the graph has a Hamiltonian circuit, i.e. instruction "exit" in Algorithm 1 has not been used for the graph.

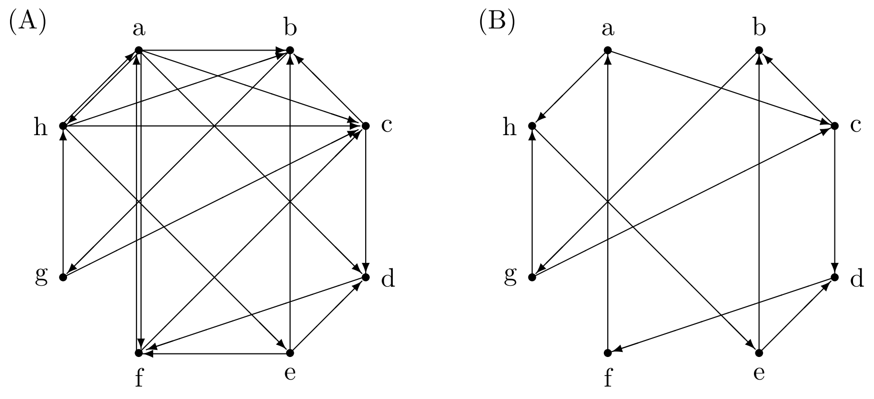

Figure 1: Graph G from Example 1. (A) The initial graph. (B) The graph after applying Algorithm 1.

Definition 3. The *reduced-by-matching graph* (RBM graph) is a 1-graph after the reduction done by Algorithm 1, i.e. a simple digraph with every vertex of non-zero indegree and outdegree, and with every arc belonging to a perfect matching in a cluster.

The reduced-by-matching graphs have the following properties.

Theorem 4. *If graph G is reduced-by-matching, then the following properties hold:*

- **(A)** *For every cluster $C = (V_{C1}, V_{C2}, A_C)$ of G, $|V_{C1}| = |V_{C2}|$.*
- **(B)** *For every cluster $C = (V_{C1}, V_{C2}, A_C)$ with $|V_{C1}| > 1$, $x \in V_{C1} \Rightarrow d^+(x) > 1$ and $x \in V_{C2} \Rightarrow d^-(x) > 1$.*
- **(C)** *If, for every cluster $C = (V_{C1}, V_{C2}, A_C)$ of G, $|V_{C1}| \leq 2$, then G is an adjoint.*
- **(D)** *Cluster $C = (V_{C1}, V_{C2}, A_C)$ has exactly two perfect matchings if and only if $\forall\, x \in V_{C1},\ d^+(x) = 2$ or $\forall\, x \in V_{C2},\ d^-(x) = 2$.*

Proof. (A) The property follows from the definition of RBM graphs, because every cluster of G possesses a perfect matching.
(B) If cluster C with $|V_{C1}| > 1$ had a vertex with only one outgoing (incoming) arc a, then a would be a part of every possible perfect matching in C. Then, the second vertex incident to a would have no other incident arc (no other matching of this vertex would be possible). Therefore, arc a together with the two incident vertices would compose a separate cluster, which is a contradiction.
(C) All clusters with $|V_{C1}| \leq 2$ are complete bipartite graphs (see (B)), thus G satisfies the condition for being an adjoint, i.e. the sets of successors for each pair of its vertices are either disjoint or the same.

(D) One should note that if one of the components in the "or" condition is true then the second one is also true. This is because neither in V_{C1} nor in V_{C2} a vertex with one outgoing/incoming arc can appear (see (B)). If every vertex has degree equal to 2, then the choice of one arc in a matching determines the choice of all other arcs as the ones not incident to vertices matched till now. The choice of the other arc at some vertex at the beginning similarly implies the choice of all other arcs. Thus, we obtain exactly two perfect matchings. Considering the "only if" part of the statement, having an RBM graph possessing exactly two perfect matchings we know from (B) that vertices must have degrees at least 2. Also none of the vertices can have degree equal to 3, because it would imply three possible matchings of this vertex to other ones and then at least three perfect matchings since in RBM graphs there are only arcs participating in feasible solutions. $\square$

Part (D) of Theorem 4 allows for some restriction on the estimation of a number of possible HCP solutions in a graph. For RBM graphs with outdegrees equal to 2 the upper bound is equal to 2^c, where c is the number of clusters in the graph.

3 Systematization of graph classes

In [2] numerous results concerning *partial directed line graphs* (PDLG) were presented, the graphs being subgraphs of directed line graphs. A recognition algorithm as well as an algorithm for a minimum completion of a PDLG to a directed line graph were proposed, both of polynomial-time complexity. Also detailed characterization of partial directed line graphs was provided. The graphs are partitioned into structures named *kernels*, which are similar to our clusters. Both kernels and clusters are bipartite graphs, but kernels have more restricted structure: for every pair K_i, K_j of kernels (with $i = j$ possible) $|V_{C1}(K_i) \cap V_{C2}(K_j)| \leq 1$ is satisfied. Our clusters are not restricted in this manner.

The minimum completion of a PDLG to a directed line graph does not preserve the property of the existence of a Hamiltonian circuit in the graph before and after the completion. Although the new PDLG class does not have a special meaning from the point of view of the solvability of HCP, we mention the class as an interesting piece of work in this area and place it in the following classification scheme.

In Fig. 2 the relationship between graph classes, which were mentioned in the paper, is shown. For most of the classes there exist algorithms solving HCP in polynomial time.

Theorem 5. *The relationships among the digraph classes: directed line graphs (DLG), partial directed line graphs (PDLG), adjoints, quasi-adjoint graphs, reduced-by-matching graphs (RBM graphs), and 1-graphs, together with their reference to HCP solvability, are characterized by the following properties (presented together in Fig. 2).*

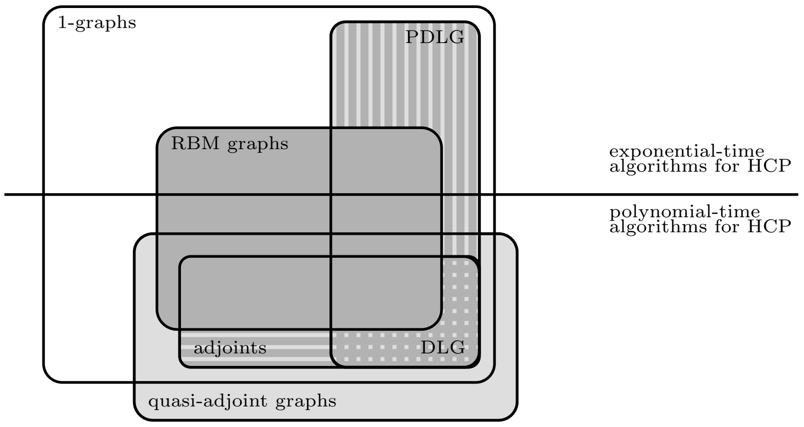
1-graphs
PDLG
RBM graphs
exponential-time algorithms for HCP
polynomial-time algorithms for HCP
adjoints
DLG
quasi-adjoint graphs

Figure 2: The relationship between digraph classes mentioned in the paper, with reference to HCP solvability. DLG stands for directed line graphs, PDLG for partial directed line graphs, and RBM graphs for reduced-by-matching graphs.

(A) *1-graphs strictly include all the classes with the exception of quasi-adjoint graphs.*

(B) *Both PDLG and adjoints strictly include DLG, and quasi-adjoint graphs stricly include adjoints.*

(C) *Subclasses PDLG$\setminus$DLG and adjoints$\setminus$DLG are disjoint.*

(D) *The class of RBM graphs has non-empty intersections with PDLG, DLG, and adjoints.*

(E) *The subclass of quasi-adjoint graphs not being adjoints has non-empty intersections with PDLG and RBM graphs.*

(F) *HCP involving quasi-adjoint graphs is solvable in polynomial time; the classes outside quasi-adjoint graphs contain non-empty subclasses with polynomial-time solution of HCP.*

Proof. **(A)** The relations between the class of 1-graphs and the other classes follow from the respective definitions.
(B) The relations between PDLG and DLG, adjoints and DLG, adjoints and quasi-adjoint graphs, follow from the respective definitions.
(C) Every adjoint not being DLG contains at least one structure, which results in parallel arcs in its original graph H [8]. Therefore, it does not belong to PDLG class, since it cannot be extended to DLG.
(D) See Fig. 3 for example elements of every subset produced by the intersection of RBM graphs with PDLG, DLG, and adjoints.
(E) See Fig. 4 for example elements of every subset produced by the intersection

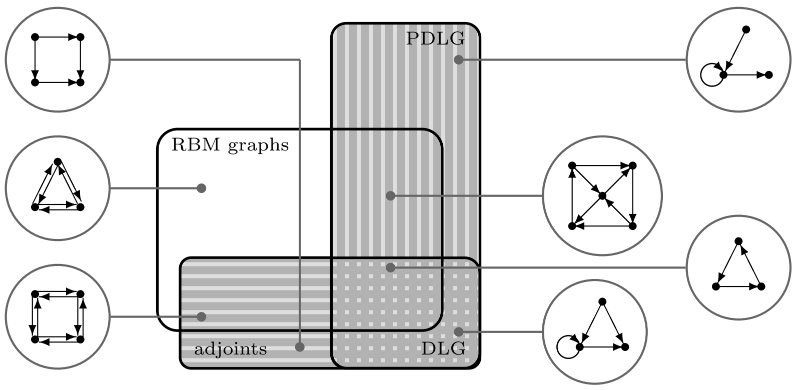

Figure 3: The relationship between DLG, PDLG, adjoints, and RBM graphs, with example elements of every subset produced by the intersection.

of quasi-adjoint graphs with PDLG and RBM graphs.
(F) In all quasi-adjoint graphs HCP can be solved in polynomial time [11]. The membership in PDLG or RBM graphs does not have such implication. It remains to show that subsets under the line in Fig. 2 and outside quasi-adjoint graphs are not empty. Assume that the subsets are composed, among others, of graphs constructed in the following manner. Take any quasi-adjoint graph from the intersection with the considered subset (PDLG, RBM graphs, both, or none of them, respectively). Replace one (or any number) of its vertices by a subgraph such as that shown in Fig. 5. Now the graph is no longer quasi-adjoint graph, but still PDLG, or RBM graph, etc. In such graphs a solution of HCP can be always found in polynomial time. Namely, all appearances of the inserted subgraphs change back to vertices, solve HCP in the quasi-adjoint graph using the algorithm from [11], and every of the vertices replace in the solution by the corresponding subpath (see the caption of Fig. 5 for an example). If no solution of HCP exists, also no solution in its extended counterpart exists. □

4 Computational experiments

Since the Hamiltonian circuit problem is both widely used for modeling real-life problems and computationally hard, there is the need to look for approaches reducing its complexity. (For our previous approach to simplify HCP with the application to DNA sequence assembly see [10].) One of these approaches may be the use of Algorithm 1, which can either reduce a directed 1-graph into a quasi-adjoint graph (making HCP polynomially solvable) or reduce the set of arcs of the graph (making the instance easier for an exact or heuristic algorithm).

In this section the usefulness of Algorithm 1 in simplifying the problem of searching for Hamiltonian circuits is checked and the results are presented. The

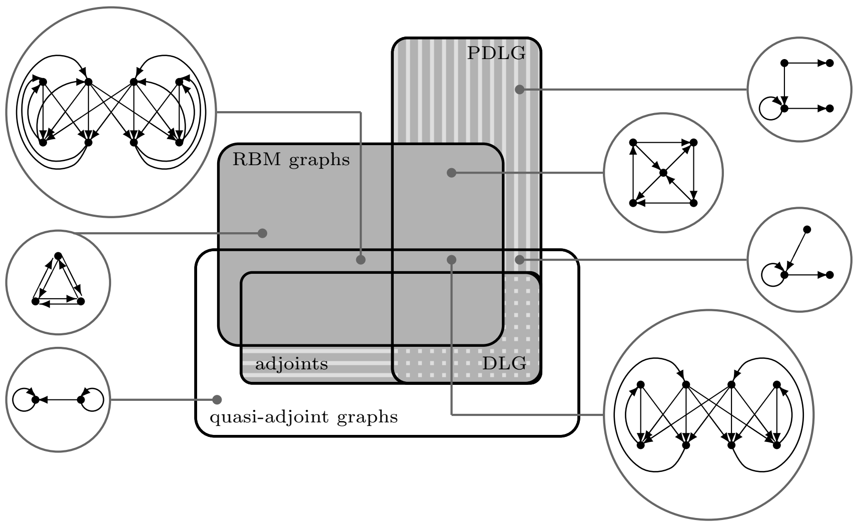

Figure 4: The relationship between DLG, PDLG, adjoints, RBM graphs, and quasi-adjoint graphs.

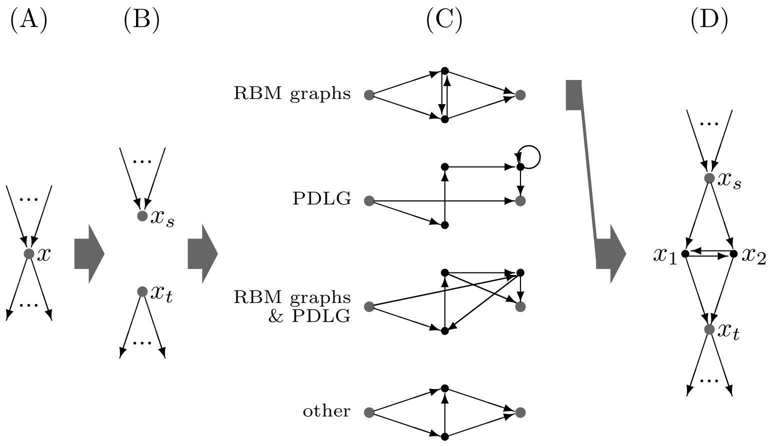

Figure 5: The transformation of a quasi-adjoint graph into other graph with polynomial-time solution of HCP (see the proof of Theorem 5). (A) A part of a quasi-adjoint graph. (B) Vertex x is replaced by two vertices x_s and x_t, all arcs entering x now enter x_s, all arcs leaving x now leave x_t. (C) The inserted subgraphs, respectively, for the RBM graphs, PDLG, the intersection of PDLG and RBM graphs, and for none of them. (D) After the insertion the initial graph is no longer the quasi-adjoint graph, but the HCP solution can be still found in polynomial time. Vertex x in a circuit found for the quasi-adjoint graph should be replaced in the example by subpath $\langle x_s, x_1, x_2, x_t \rangle$.

aim of these tests is to measure, for random and benchmark instances, both the effectiveness of the reduction (i.e. the number of the reduced arcs) and the effectiveness of algorithms gained by the reduction (i.e. the computational time and the quality of the solutions).

The benchmark instances for the problem are of two kinds. Instances of the first kind come from the TSPLIB page [25], the section "HCP" (the same instances are pointed to on the Hamiltonian page [19]). The graphs are placed in files "alb$X$.hcp", where $X \in \langle 1000, 5000 \rangle$ stands for the number of vertices. For every instance its solution is provided (files "alb$X$.opt.tour"). Because the graphs originally were undirected, we have adapted them by simple modification: all edges have been replaced by arcs with their direction based on the order of vertices within pairs in the file, and next the directions of the arcs constituting the given solution have been corrected to guarantee the Hamiltonicity. Instances of the second kind were derived from largest asymmetric TSP instances from the TSPLIB page (they are also available via the page of the 8th DIMACS Implementation Challenge for TSP [16]). Files "rbg$X$.atsp" with $X \in \langle 323, 443 \rangle$ encode complete graphs with weights on arcs. The derived HCP instances have taken from them only arcs with weights less than a given bound, and have been supplemented by arcs guaranteeing their Hamiltonicity.

Four algorithms solving HCP have been used in the computational experiments, two heuristics and two exact algorithms. Two methods are known from the literature: the heuristic randomized algorithm of Gopal Pandurangan [23] and the exact backtracking algorithm invented by William Kocay and improved by Andrew Chalaturnyk [22, 13]. The Pandurangan's algorithm was implemented by us while the implementation of the Chalaturnyk's algorithm comes from the Groups & Graphs page [18]. Although Pandurangan's algorithm is guaranteed to find only a 2-factor, in sparse graphs (which are of main interest here) it often finds long cycles and even Hamiltonian ones. The Chalaturnyk's algorithm was designed for undirected graphs, but has appeared to be suitable for our purposes. In addition, two simple algorithms were implemented by us: Algorithm 2 is an exact, enumerative method and Algorithm 3 is a heuristic. Both algorithms are not sophisticated, but it is of little significance here because mainly the relative improvement of their outcomes is of interest. As they are rather intuitive, their results can somehow predict how other algorithms incorporating similar rules could respond for the reduction.

Algorithm 2

```
Input: A digraph $G$ with $n$ vertices.

Output: A Hamiltonian circuit in $G$ saved in table $solution$, if exists.

(1) Initialize variables:  $index \leftarrow 1$, $solution[index] \leftarrow 1$.
      Execute the following recurrent procedure.

(2) Procedure($index$,$solution$)
      {
          if $index = n$ then
              if arc($solution[index]$,1) exists then
```

```
        {
            print $solution$;
            exit;
        }
    else for $i = 1..n$ do
        if $i$ is not in current path and arc($solution\,[index]$,$i$) exists then
        {
            $solution\,[index+1] \leftarrow i$;
            Procedure($index+1$,$solution$);
        }
}
```

Algorithm 3

```
Input: A digraph $G$ with $n$ vertices.
Output: A simple circuit in $G$ (optimally being the Hamiltonian circuit)
    saved in table $solution$.
(1) Initialize variables:  $index \leftarrow 1$,  $solution\,[index] \leftarrow 1$.
(2) for $index = 2..n$ do
    {
        find $i \notin solution$ such that arc($solution\,[index-1]$,$i$) exists and
            number of immediate successors of $i$ not being in $solution$
            is minimal;
        if there is no such $i$ then
        {
            $index \leftarrow index - 1$;
            go to step (3);
        }
        $solution\,[index] \leftarrow i$;
    }
(3) while arc($solution\,[index]$,1) does not exists and $index > 0$ do
        $index \leftarrow index - 1$;
(4) print $solution$ of length $index$.
```

We started the experiment with random instances. The computations were done on a PC with an Intel T2300 1.7 GHz processor with 1 GB RAM. The instances were generated randomly with a uniform distribution. For a given number n of vertices of the graph and a given average outdegree d, arcs were added according to the following rule. Firstly, a random Hamiltonian circuit was built on the vertices to ensure at least one feasible solution. Next, the remaining $n(d-1)$ arcs were added randomly provided that the graph must be a 1-graph.

Table 1 contains results of reduction of random initial graphs from Algorithm 1. The graphs were generated with the use of two parameters: the number of vertices n and the initial average outdegree d, and for every pair of their values 100 instances were produced. The initial numbers of arcs in the graphs are

Table 1: The initial numbers of arcs in the graphs (the left half of the table) and the average numbers of arcs after the reduction done by Algorithm 1 (the right half) as functions of changing parameters n (the number of vertices in the graphs) and d (the initial average outdegree).

		before the reduction				**after the reduction**			
		d				d			
		2	3	4	5	2	3	4	5
n	10	20	30	40	50	11.5	21.3	33.1	44.7
	20	40	60	80	100	22.5	41.9	68.7	93.5
	30	60	90	120	150	33.4	62.9	106.1	140.9
	40	80	120	160	200	43.4	86.0	143.5	189.6
	50	100	150	200	250	54.6	110.2	179.9	237.3
	60	120	180	240	300	64.9	132.4	216.3	287.3
	70	140	210	280	350	75.2	153.8	252.4	335.7
	80	160	240	320	400	85.5	179.0	287.5	384.0
	90	180	270	360	450	94.8	199.9	324.2	431.2
	100	200	300	400	500	105.9	226.8	362.5	479.6

present in the left half of the table. The right half shows the average numbers of arcs after the reduction.

We see in Table 1 that the number of reduced arcs in the graphs reaches almost 50% of their initial cardinality in column $d = 2$. In other words, after the reduction vertices have usually only one outgoing arc in the graphs. It means that thanks to Algorithm 1 the detection of the Hamiltonian circuit in these graphs has become easy (on average).

Figures 6 and 7 visualize the results from Table 1 in the form of two functions: the average number of reduced arcs and the effectiveness of the reduction, respectively. The first function is defined as the difference between the numbers of arcs in graphs before and after the reduction done by Algorithm 1. The second function is the first one taken as the percentage of the maximum number of reduced arcs, i.e. it is the ratio of the average number of reduced arcs to the difference between the initial number of arcs and the number of vertices.

Figure 7 shows that the fewer the initial outdegree is, the greater the effectiveness of the reduction of arcs is. It is almost independent of the instance size. The value of 100% means that all arcs not composing a solution were removed by Algorithm 1. The graphs with vertices having two outgoing arcs on average result in the mean effectiveness equal to 91%. The graphs with vertices having the average number of outgoing arcs equal to 3 result in the mean effectiveness equal to 41%.

Even if the reduction from Algorithm 1 does not transform a graph into a quasi-adjoint, the solution can be found much more effectively in the reduced graph by some exact or heuristic algorithms. In Tables 2 and 3 the gain is shown to be significant — the computation time in the case of exact Algorithm 2 (see Table 2) and the quality of the generated solutions in the case of heuristic

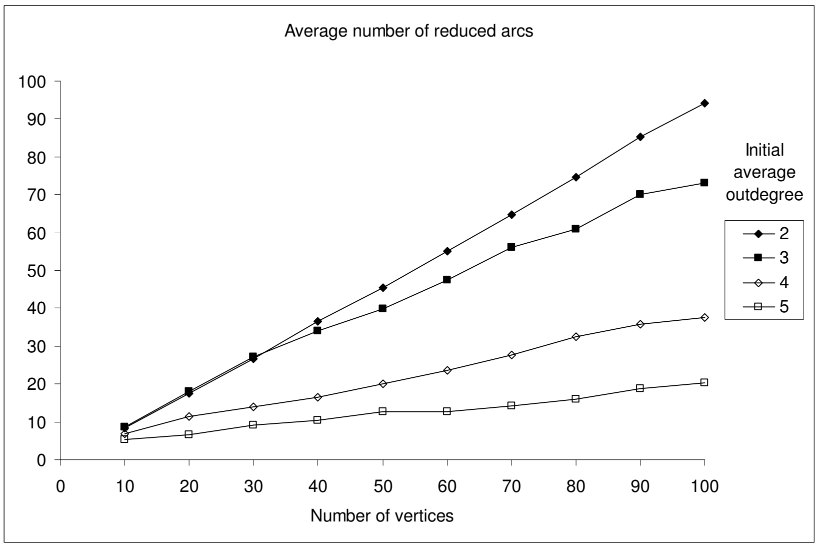

Figure 6: The average number of reduced arcs as the function of the number of vertices (X-axis) and the initial average outdegree in the graphs (the key on the right). The function is defined as the difference between the numbers of arcs in the graphs before and after the reduction done by Algorithm 1.

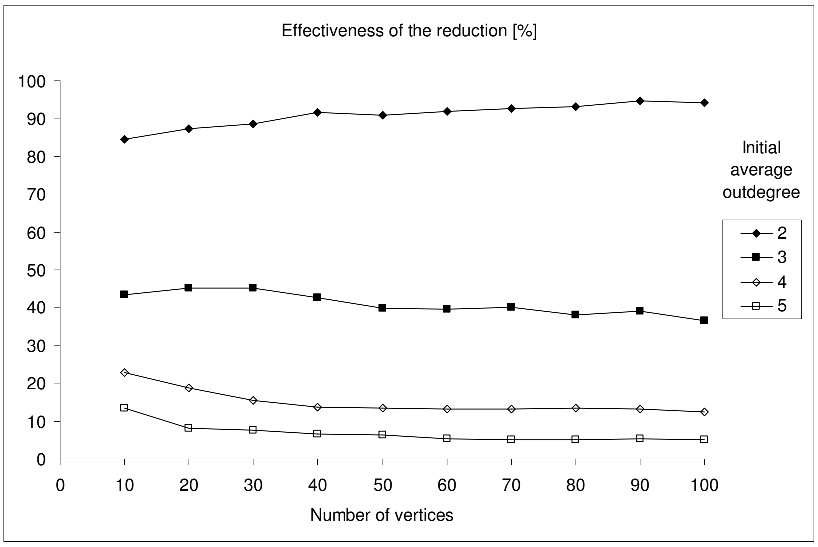

Figure 7: The effectiveness of the reduction as the function of the number of vertices (X-axis) and the initial average outdegree in the graphs (the key on the right). The effectiveness is defined as the ratio of the average number of reduced arcs to the difference between the initial number of arcs and the number of vertices.

Table 2: Average computation times of Algorithm 2 for the initial graphs (the left half of the table) and for the reduced graphs (the right half) as functions of changing parameters n (the number of vertices in the graphs) and d (the initial average outdegree). The times given in seconds.

		before the reduction				**after the reduction**			
		d				d			
		2	3	4	5	2	3	4	5
n	10	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	20	<0.01	<0.01	0.01	<0.01	<0.01	<0.01	<0.01	0.02
	30	<0.01	0.08	3.12	16.35	<0.01	<0.01	0.11	0.79
	40	0.01	5.86	—	—	<0.01	0.04	13.79	—
	50	0.09	619.61	—	—	<0.01	0.83	—	—
	60	1.12	—	—	—	<0.01	16.56	—	—
	70	9.87	—	—	—	<0.01	—	—	—
	80	124.99	—	—	—	<0.01	—	—	—
	90	—	—	—	—	<0.01	—	—	—
	100	—	—	—	—	<0.01	—	—	—

Algorithm 3 (Table 3).

The missing results in Table 2 represent cases when acceptable computation time was exceeded. The improvement in computational time after the graph reduction is significant. For example, for $n = 50$ and $d = 3$ the speedup is 746-fold, and more than 12499-fold for $n = 80$ and $d = 2$. For $n = 90$ or $n = 100$ and $d = 2$ the speedup is even greater — the empty entries versus less than 10 milliseconds are meaningful.

Table 3 shows the improvement in quality of solutions generated by Algorithm 3. It can be observed that the data series in the right of Table 3 have their local minima. After very good results for sparse graphs, Algorithm 3 was returning worse circuits, reaching the minima on average at 3 outgoing arcs per a vertex. After that, solution quality gradually increases. The average quality for graphs with $d = 2$ is equal to 15.04 for the initial graphs and to 43.92 for the reduced graphs. For $d = 3$ the average quality is equal to 16.92 for the initial graphs and to 21.03 for the reduced graphs.

Figure 8 presents the improvement of the quality of the results produced by Algorithm 3. This function is defined as the ratio of the difference of the qualities for the graphs before and after the reduction, to the maximum possible difference (the latter being the number of vertices in the graph minus the quality in the initial graph). Thus, the quality improvement equal to 100% means that the quality in the reduced graphs could not be improved. The quality improvement for the graphs with vertices having two outgoing arcs on average is 73%, and 16% for three outgoing arcs.

It is worth noting how many initial graphs were transformed into quasi-adjoint graphs by applying Algorithm 1. Table 4 presents this information. The results for graphs with $d = 2$ are very good, on average 51% of instances

Table 3: Average quality of solutions of Algorithm 3 for the initial graphs (the left half of the table) and for the reduced graphs (the right half) as functions of changing parameters n (the number of vertices in the graphs) and d (the initial average outdegree). The quality is defined as the number of vertices in a simple circuit returned by Algorithm 3.

		before the reduction				**after the reduction**			
		d				d			
		2	3	4	5	2	3	4	5
n	10	7.75	7.78	7.71	8.91	9.24	8.69	8.07	9.04
	20	10.93	10.55	12.26	14.65	18.27	12.83	13.71	15.84
	30	14.84	13.09	16.48	20.13	27.33	15.31	17.95	21.09
	40	13.56	13.26	20.64	22.63	31.54	20.33	21.97	23.99
	50	14.16	18.56	21.76	29.57	37.87	21.84	25.23	30.46
	60	18.66	18.28	22.85	31.75	46.96	23.26	26.22	34.90
	70	17.43	17.45	24.27	35.73	56.05	23.09	28.07	38.75
	80	17.40	23.14	28.17	36.41	63.96	26.15	34.59	41.67
	90	19.05	24.33	31.08	38.22	71.34	28.73	37.02	45.35
	100	16.57	22.75	29.51	42.14	76.65	30.05	40.88	48.37

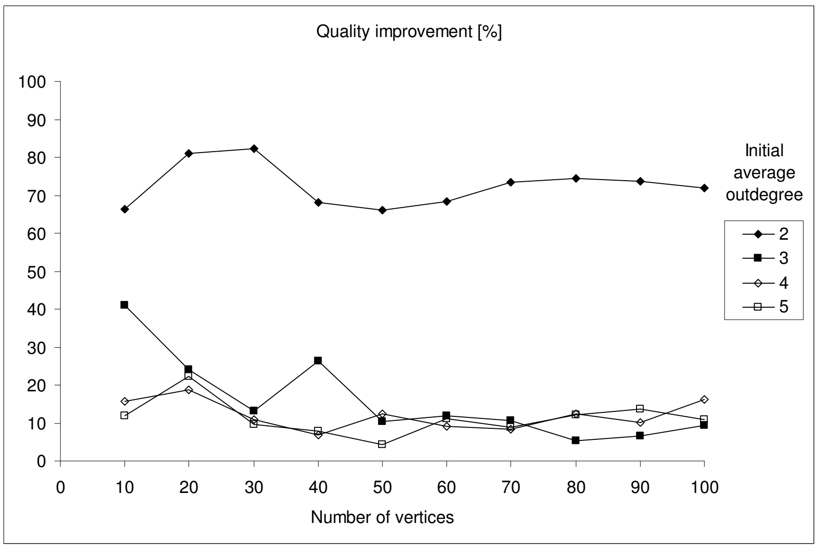

Figure 8: The improvement of the quality as the function of the number of vertices (X-axis) and the initial average outdegree in the graphs (the key on the right). The improvement is defined as the ratio of the difference between the qualities for the graphs after and before the reduction, to the maximum possible difference.

Table 4: The number of quasi-adjoint graphs among 100 random instances, for the initial graphs (the left half of the table) and for the reduced graphs (the right half) as functions of changing parameters n (the number of vertices in the graphs) and d (the initial average outdegree).

		before the reduction				**after the reduction**			
		d				d			
		2	3	4	5	2	3	4	5
n	10	0	0	0	0	76	9	0	0
	20	0	0	0	0	63	2	0	0
	30	0	0	0	0	58	0	0	0
	40	0	0	0	0	57	0	0	0
	50	0	0	0	0	46	0	0	0
	60	0	0	0	0	42	0	0	0
	70	0	0	0	0	41	0	0	0
	80	0	0	0	0	41	0	0	0
	90	0	0	0	0	51	0	0	0
	100	0	0	0	0	38	0	0	0

Table 5: The results of Algorithm 1 for HCP-descended instances from TSPLIB.

	before the reduction			**after the reduction**		
	#arcs	avg d^+	q-adj.	#arcs	avg d^+	q-adj.
alb1000.hcp	1998	2.00	N	1000	1.00	Y
alb2000.hcp	3996	2.00	N	2013	1.01	N
alb3000a.hcp	5999	2.00	N	3000	1.00	Y
alb3000b.hcp	5997	2.00	N	3015	1.00	N
alb3000c.hcp	5996	2.00	N	3005	1.00	N
alb3000d.hcp	5993	2.00	N	3007	1.00	N
alb3000e.hcp	5996	2.00	N	3002	1.00	Y
alb4000.hcp	7997	2.00	N	4000	1.00	Y
alb5000.hcp	9999	2.00	N	5033	1.01	N

passed to the class of graphs of polynomial-time solvability of HCP. Although more dense graphs did not become quasi-adjoint graphs, yet they lost a number of arcs and they became simpler from the point of view of solvability of HCP.

In order to verify the usefulness of Algorithm 1 on different kind of instances, we selected to further tests much greater benchmark graphs (described at the beginning of this section). Tables 5 and 6 show the results of the reduction made for them by Algorithm 1. In the tables, "#arcs" is the number of arcs in the graph, "avg d^+" means the average outdegree, "q-adj." means the quasi-adjoint ("Yes" or "No"), and b stands for the bound for the ATSP graphs (i.e. the arcs with weights less than b were accepted).

As we see, Algorithm 1 is still very useful, not only for sparse graphs with average outdegree equal to 2, but also for instances with much greater average

Table 6: The results of Algorithm 1 for ATSP-descended instances from TSPLIB.

		before the reduction			after the reduction		
	b	#arcs	avg d^+	q-adj.	#arcs	avg d^+	q-adj.
rbg323.atsp	1	4923	15.24	N	1910	5.91	N
rbg323.atsp	2	5020	15.54	N	2746	8.50	N
rbg323.atsp	3	5089	15.76	N	3077	9.53	N
rbg323.atsp	4	5187	16.06	N	3286	10.17	N
rbg323.atsp	5	5299	16.41	N	3925	12.15	N
rbg358.atsp	1	8096	22.61	N	5170	14.44	N
rbg358.atsp	2	8176	22.84	N	6258	17.48	N
rbg358.atsp	3	8256	23.06	N	6897	19.27	N
rbg358.atsp	4	8368	23.37	N	7554	21.10	N
rbg358.atsp	5	8534	23.84	N	8041	22.46	N
rbg403.atsp	1	12087	29.99	N	8077	20.04	N
rbg403.atsp	2	12215	30.31	N	9344	23.19	N
rbg403.atsp	3	12325	30.58	N	9954	24.70	N
rbg403.atsp	4	12450	30.89	N	10365	25.72	N
rbg403.atsp	5	12695	31.50	N	11653	28.92	N
rbg443.atsp	1	13857	31.28	N	10236	23.11	N
rbg443.atsp	2	14137	31.91	N	11490	25.94	N
rbg443.atsp	3	14390	32.48	N	12534	28.29	N
rbg443.atsp	4	14636	33.04	N	12807	28.91	N
rbg443.atsp	5	14980	33.81	N	13500	30.47	N

outdegree, up to 30. The reduction applied to the HCP-descended graphs resulted in graphs close to simple cycles. Four instances have been turned into Hamiltonian circuits, while the remaining ones have become much easier, even for the time-consuming simple exact Algorithm 2 (see Table 7). The same algorithm could not be used for these instances before the reduction, its computations for the smallest instance with 1000 vertices had to be broken after several hours. In the face of the time of its reduction consumed by Algorithm 1 (97 seconds), the gain is visible.

In Table 7, the values in columns "PA" are mean values for 100 runs of this randomized algorithm. The results for "CA" have been obtained for undirected graphs derived from their directed counterparts by replacing arcs with edges. The difference between the performance of the algorithms for the initial and reduced graphs is significant. Let us focus on algorithm CA. It immediately checked for Hamiltonicity of graphs before the reduction alb1000.hcp and alb2000.hcp, however, for alb3000a.hcp we decided to stop its computations after 5 hours. The same algorithm had no problems with the reduced instances.

The similar comparison for ATSP-descended instances has appeared to be not such spectacular, especially because of very long computational time for even the smallest graph. Both algorithms A2 and CA could not handle the smallest graph (rbg323.atsp with $b = 1$) within reasonable time (5 hours). A gain on the quality was also not impressive, for PA it was 203.7 (after the

Table 7: The results for HCP-descended instances achieved by Algorithm 2 (A2), Algorithm 3 (A3), the Pandurangan's algorithm (PA) and the Chalaturnyk's algorithm (CA). The results of exact algorithms given as computational time, the results of heuristics represented by the number of vertices in longest found simple cycles.

	before the reduction				**after the reduction**			
	A2	A3	PA	CA	A2	A3	PA	CA
alb1000.hcp	—	84	879.5	<1 sec.	<1 sec.	1000	1000.0	<1 sec.
alb2000.hcp	—	117	1422.6	<1 sec.	<1 sec.	1615	1574.1	<1 sec.
alb3000a.hcp	—	564	2090.5	—	<1 sec.	3000	3000.0	<1 sec.
alb3000b.hcp	—	211	2521.4	—	1 sec.	2200	2617.5	<1 sec.
alb3000c.hcp	—	24	2903.0	—	<1 sec.	2841	2934.8	1 sec.
alb3000d.hcp	—	0	2467.1	—	1 sec.	1117	2547.8	<1 sec.
alb3000e.hcp	—	177	2018.7	—	<1 sec.	1997	2538.6	<1 sec.
alb4000.hcp	—	36	4000.0	—	<1 sec.	4000	4000.0	<1 sec.
alb5000.hcp	—	0	2628.4	—	139 sec.	1588	3027.2	<1 sec.

reduction) vs. 193.1 (before the reduction). As we observed, even after the reduction many vertices remained with a large number of incident arcs, what made these instances intractable.

5 Conclusions

In this paper both theoretical and practical aspects of simplifying the Hamiltonian circuit problem in digraphs have been studied.

We have examined our algorithm from [11], carrying out a reduction of arcs toward simplifying searching for a Hamiltonian circuit in a graph (Algorithm 1). In the previous work the theoretical correctness of the algorithm was proved, but till now we knew nothing about the practical significance of the reduction proposed there. In the computational experiments from the preceding section, the effectiveness of Algorithm 1 has been verified together with the performance of four algorithms solving HCP, executed for initial and reduced graphs, for random and benchmark instances. The tests demonstrate the effect of Algorithm 1, which reaches, for example, 91% of the effectiveness of the reduction for random graphs having two outgoing arcs per a vertex on average. For benchmark graphs with mean outdegree equal to two, its effectiveness has appeared to be very close to 100% (99.75% on average). Algorithm 2 and Algorithm 3 improved their efficiency for reduced random graphs, reaching even 12499-fold speedup or 4.6-fold better quality, respectively, with even better results for benchmark HCP-descended graphs. Also algorithms known from the literature worked much better for the reduced instances.

Concerning the more theoretical aspects of this paper, the new class of reduced-by-matching graphs has been defined and studied, the graphs being output of Algorithm 1. Optimally, the graphs are reduced to quasi-adjoint graphs, what results in polynomial-time solution of the Hamiltonian circuit problem. Although among graphs generated randomly in the experiment no one appeared to be a quasi-adjoint graph, the application of Algorithm 1 changed this state a lot for sparse graphs. For mean initial outdegree equal to 2, 51% of graphs on average became quasi-adjoint graphs and thus appeared to be easy for the Hamiltonian circuit problem (similar proportion is satisfied for benchmark graphs). From the practical point of view, Algorithm 1 identifies these graphs (on average) as computationally easy instances, what follows the reduction of the mean initial outdegree from 2 to almost 1. More dense random graphs also became simpler from the point of view of solving HCP, the loss of arcs resulted in much more effective searching for a solution by an exact or heuristic algorithm.

In addition to the above work, nevertheless being an important piece of work never before presented in the literature, the relationship between reduced-by-matching graphs, quasi-adjoint graphs and other related digraph classes has been visualized here and its correctness proved.

6 Acknowledgements

The authors thank Graham Kendall and an anonymous referee for their valuable remarks. The research has been partially supported by grant No. DEC-2011/01/B/ST6/07021 from the National Science Centre, Poland.

References

[1] M. Abbas and Z. Benmeziane, Hamiltonicity in partly claw-free graphs, *RAIRO Operations Research*, 43 (2009), 103–113.

[2] N. Apollonio and P.G. Franciosa, A characterization of partial directed line graphs, *Discrete Mathematics*, 307 (2007), 2598–2614.

[3] J. Bang-Jensen and G. Gutin, On the complexity of hamiltonian path and cycle problems in certain classes of digraphs, *Discrete Applied Mathematics*, 95 (1999), 41–60.

[4] J. Bang-Jensen, G. Gutin, and A. Yeo, A polynomial algorithm for the Hamiltonian cycle problem in semicomplete multipartite digraphs, *Journal of Graph Theory*, 29 (1998), 111–132.

[5] C. Berge, *Graphs and Hypergraphs*, North-Holland Publishing Company, London, 1973.

[6] A. A. Bertossi, The edge Hamiltonian path problem is NP-complete, *Information Processing Letters*, 13 (1981), 157–159.

[7] M. Blais and G. Laporte, Exact solution of the generalized routing problem through graph transformations, *Journal of the Operational Research Society*, 54 (2003), 906-910.

[8] J. Blazewicz, A. Hertz, D. Kobler, and D. de Werra, On some properties of DNA graphs, *Discrete Applied Mathematics*, 98 (1999), 1–19.

[9] J. Blazewicz and M. Kasprzak, Computational complexity of isothermic DNA sequencing by hybridization, *Discrete Applied Mathematics*, 154 (2006), 718–729.

[10] J. Blazewicz and M. Kasprzak, Graph reduction and its application to DNA sequence assembly, *Bulletin of the Polish Academy of Sciences. Technical Sciences*, 56 (2008), 65–70.

[11] J. Blazewicz, M. Kasprzak, B. Leroy-Beaulieu, and D. de Werra, Finding Hamiltonian circuits in quasi-adjoint graphs, *Discrete Applied Mathematics*, 156 (2008), 2573–2580.

[12] P. Camion, Chemins et circuits hamiltoniens des graphes complets, *Comptes Rendus de l'Acadmie des Sciences de Paris*, 249 (1959), 2151-2152.

[13] A. Chalaturnyk, A Fast Algorithm for Finding Hamilton Cycles, M.Sc. Thesis at the University of Manitoba (2008).

[14] G. Chartrand and L. Lesniak, *Graphs and Digraphs*, Wadsworth & Brooks/Cole, Pacific Grove, 1986.

[15] J.S. Deogun and G. Steiner, Polynomial algorithm for Hamiltonian cycle in cocomparability graphs, *SIAM Journal on Computing*, 23 (1994), 520–552.

[16] 8th DIMACS Implementation Challenge: The Traveling Salesman Problem, http://www2.research.att.com/˜dsj/chtsp/.

[17] M.R. Garey and D.S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, 1979.

[18] Groups & Graphs page, http://www.combinatorialmath.ca/G&G/.

[19] The Hamiltonian page, http://alife.ccp14.ac.uk/memetic/www.densis.fee .unicamp.br/˜moscato/Hamilton.html.

[20] R.W. Hung and M.S. Chang, Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs, *Theoretical Computer Science*, 341 (2005), 411–440.

[21] J.M. Keil, Finding Hamiltonian circuits in interval graphs, *Information Processing Letters*, 20 (1985), 201–206.

[22] W. Kocay, An extension of the multi-path algorithm for Hamilton cycles, *Discrete Mathematics*, 101 (1992), 171–188.

[23] G. Pandurangan, On a simple randomized algorithm for finding a 2-factor in sparse graphs, *Information Processing Letters*, 95 (2005), 321–327.

[24] M. Preissmann, Detection of circuits and ordering in regular networks at two dimensions, Report CRSI-IMAG-RR-004, 1985.

[25] TSPLIB page, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.