Finding Uniquely Hamiltonian Graphs of Minimum Degree Three with Small Crossing Numbers

Benedikt Klocker(✉), Herbert Fleischner, and Günther R. Raidl

Institute of Computer Graphics and Algorithms, TU Wien,
Favoritenstraße 9–11/1861, 1040 Vienna, Austria
{klocker,fleischner,raidl}@ac.tuwien.ac.at

Abstract. In graph theory, a prominent conjecture of Bondy and Jackson states that every uniquely hamiltonian planar graph must have a vertex of degree two. In this work we try to find uniquely hamiltonian graphs with minimum degree three and a small crossing number by minimizing the number of crossings in an embedding and the number of degree-two vertices. We formalize an optimization problem for this purpose and propose a general variable neighborhood search (GVNS) for solving it heuristically. The several different types of used neighborhoods also include an exponentially large neighborhood that is effectively searched by means of branch and bound. To check feasibility of neighbors we need to solve hamiltonian cycle problems, which is done in a delayed manner to minimize the computation effort. We compare three different configurations of the GVNS. Although our implementation could not find a uniquely hamiltonian planar graph with minimum degree three disproving Bondy and Jackson's conjecture, we were able to find uniquely hamiltonian graphs of minimum degree three with crossing number four for all number of vertices from 10 to 100.

Keywords: Variable neighborhood search · Uniquely Hamiltonian graphs · Combinatorial optimization

1 Introduction

A lot of research in graph theory focuses on hamiltonian cycles. The problem of finding hamiltonian cycles is well studied in theoretical aspects [12] and in practical aspects as a special case of the traveling salesman problem [2]. An interesting topic in graph theory is the question of how many hamiltonian cycles a given graph has. A simpler version of this question only asks if there is exactly one hamiltonian cycle in a given graph.

In this paper we will only be concerned with undirected simple graphs and just write graph for this type of graphs. A graph $G = (V, E)$ is uniquely hamiltonian if and only if it contains exactly one hamiltonian cycle, i.e., a cycle visiting

This work is supported by the Austrian Science Fund (FWF) under grant P27615.

M.J. Blesa et al. (Eds.): HM 2016, LNCS 9668, pp. 1–16, 2016.
DOI: 10.1007/978-3-319-39636-1_1

each node exactly once. A graph G is planar if and only if it has a planar embedding, i.e., it can be drawn in the Euclidean plane without any crossing edges. The crossing number $\mathrm{cr}(G)$ of a graph G is the smallest possible number of crossings in an embedding of G into the plane. Last but not least, a graph G has minimum degree $\delta(G) = k$ if each node has at least k incident edges.

One type of problem in the area of uniquely hamiltonian graphs is to determine for a given class of graphs if it contains a uniquely hamiltonian graph. Bondy and Jackson [4] showed that every uniquely hamiltonian graph with n vertices has a vertex with degree at most $c \log(8n) + 3$ for a small constant c. This limits the minimum degree of a uniquely hamiltonian graph (note that a better lower bound for $\delta(G)$ has been established in [1]). Bondy and Jackson proved in the same paper also a simpler statement: Every planar uniquely hamiltonian graph contains at least two vertices of degree two or three. Furthermore, they stated an interesting still unsolved conjecture that every planar uniquely hamiltonian graph contains a vertex of degree two.

In the case of non-planar graphs a question by Sheehan [21] asks whether or not a uniquely hamiltonian 4-regular graph exists. Note that a graph which only contains vertices with odd degree cannot be uniquely hamiltonian [22]. Fleischner [9] showed that in the case of multigraphs there exist 4-regular uniquely hamiltonian graphs. In more recent work Fleischner [10] constructed an infinite family of uniquely hamiltonian (simple) graphs with minimum degree four. This surprising result leads to the guess that there may also be a uniquely hamiltonian planar graph with minimum degree three which would disprove the conjecture of Bondy and Jackson. Entringer and Swart [8] constructed already in 1980 uniquely hamiltonian graphs with minimal degree three, but they are not planar.

In this paper we transform the problem of finding a uniquely hamiltonian planar graph with minimum degree three into a bi-objective optimization problem which minimizes the crossing number and the number of vertices with degree two. To solve this problem nearly optimal we propose a general variable neighborhood search (GVNS) heuristic [14]. The GVNS framework was already successfully applied to other graph theoretical problems, see, e.g., [5].

As the search space increases exponentially with increasing number of vertices a heuristic could be beneficial compared to an exact approach. In fact, before we implemented the general variable neighborhood search, we tried to apply an exact approach, which enumerates graphs of a given vertex degree in a clever way and tests if they are uniquely hamiltonian [19]. With that approach we were able to check that no uniquely hamiltonian planar graph with minimum degree three and with 22 or less vertices exists. Unfortunately, for graphs with more than 22 vertices the running time explodes because of the vast number of graphs to check.

One of our proposed neighborhood structures is exponentially large and we use a branch-and-bound procedure to find the best neighbor in its neighborhoods. This embeds the idea of a large neighborhood search [20] in our GVNS. We will see that the bottleneck in our algorithm, which consumes most of the running time, are the expensive unique hamiltonicity checks. To reduce the number of such checks we keep infeasible, not uniquely hamiltonian, solutions

formally in our neighborhood. Only after finding the best neighbor we apply a Lin-Kerninghan heuristic [16] and in a later step we use Concorde [7] to check for feasibility.

In the next section we describe some transformations of the problem and formally state the resulting optimization problem. In Sect. 3 we describe our GVNS framework and the neighborhood structures in detail. The comparison of three different configurations and the experimental results are presented in Sect. 4. Finally, we conclude with Sect. 5 and propose some further work ideas.

2 Problem Description

In graph theory an important and challenging open question is whether or not a *uniquely hamiltonian planar graph with minimum degree three* (UHPG3) exists [4]. Bondy and Jackson [4] conjectured that every planar uniquely hamiltonian graph contains a vertex of degree two, i.e., that no UHPG3 exists. So far, however, neither could a UHPG3 be found nor could it be proven that none exists.

To disprove Bondy and Jackson's conjecture, it would be enough to find a planar graph with minimum degree three that contains an arbitrarily selected fixed edge $e \in E$ and exactly one hamiltonian cycle containing this edge e. This means the graph may contain also other hamiltonian cycles which do not contain the edge e. From such a graph we can remove the edge $e = (v, w)$, duplicate the remaining graph and connect the two copies by adding edges (v, v') and (w, w'), where v' and w' are the duplicates of nodes v and w, respectively. The resulting graph is then uniquely hamiltonian and still planar with minimum degree three. We call a uniquely hamiltonian graph whose Hamiltonian cycle contains the known edge e *fixed edge uniquely hamiltonian graph* (FEUHG).

In this work we concentrate on the optimization problem variant of finding a graph G with a given number of nodes $n = |V|$ that *as far as possible* corresponds to a UHPG3. To this end, we relax the conditions that the graph must be planar and must have minimum degree three and instead minimize the deviations from these properties.

To our knowledge, the problem of finding graphs that as far as possible correspond to a UHPG3 has so far not been considered in a more systematical, and in particular computational way.

More specifically, we consider the bi-objective optimization problem to find a FEUHG G together with an embedding into the plane and minimize

- the number of edge crossings and
- the number of vertices with degree two.

To obtain a single-objective optimization problem we linearly combine these two objectives with corresponding weights α and β.

Since the problem of computing the crossing number of a graph is NP-hard (see [11]), it makes sense to approximate this value. We do this by allowing only crossings between edges which are not part of the uniquely hamiltonian cycle

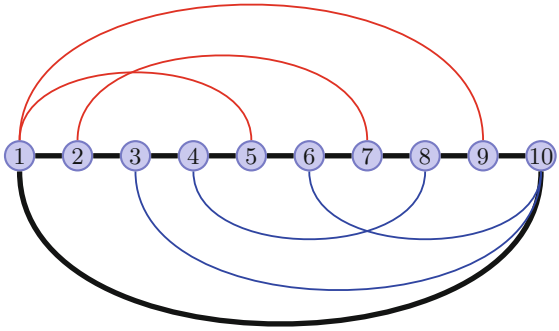

Fig. 1. Example solution with two crossings and no nodes of degree two. Edges of the hamiltonian cycles are bold and chords are colored red and blue. (Color figure online)

of the graph. This relaxation allows us to fix a hamiltonian cycle in advance and then only concentrate on the problem of adding additional edges (*chords*) between vertices that are no neighbors on the cycle. Since the added chords are not allowed to cross with an edge of the hamiltonian cycle, there are only two possibilities how to draw a chord: either outside the cycle or inside the cycle. We encode this two states, inside or outside, of a chord by two colors.

Let us assume without loss of generality that $V = \{1, \ldots, n\}$ and our predefined cycle C visits the nodes in the natural order from 1 to n before getting back to 1. If we fix the chords and their colors it is easy to derive the minimal number of crossings. To see this, we draw the nodes from 1 to n on one line, such that the hamiltonian cycle consists of this line together with a half cycle connecting the vertices n and 1. We draw now every chord as a half cycle, either above the line or below the line depending on the color of the chord. Figure 1 illustrates this construction with an example. For this construction we get that two chords (i, j) and (k, ℓ) with $1 \leq i < j \leq n$ and $1 \leq k < \ell \leq n$ cross if and only if their corresponding half cycles cross. This is the case if and only if the two chords have the same color and

$$i < k < j < \ell \text{ or } k < i < \ell < j \tag{1}$$

holds. Notice that in any drawing of a graph, two chords, which are on the same side of the graph and satisfy (1), cross at least once. This implies the optimality of our construction.

All together we get the following optimization problem. Given the cycle C over nodes $V = \{1, \ldots, n\}$, let

$$\mathcal{H} = \{(i, j) \mid i = 1, \ldots, n-2,\ j = i+2, \ldots, n\} \setminus \{(1, n)\}$$

be the set of all possible chords. A candidate solution is represented by (H, c), where $H \subseteq \mathcal{H}$ is the subset of selected chords and $c : H \to \{0, 1\}$ specifies their coloring by assigning either 0 or 1.

$$\underset{\substack{H\subseteq\mathcal{H}\\ c:H\to\{0,1\}}}{\text{minimize}} \quad \alpha\cdot\sum_{(i,j)\in H}\left|\{(k,\ell)\in H : c(i,j)=c(k,\ell),\, i<k<j<\ell\}\right| + \beta\cdot\left(n-\left|\bigcup_{(i,j)\in H}\{i,j\}\right|\right) \tag{2}$$

$$\text{subject to: there is no hamiltonian cycle in } C\cup H \text{ containing edge } (1,2) \text{ and at least one chord.} \tag{3}$$

In this formulation, there is always the same edge, namely the edge $(1,2)$, fixed. This eliminates some symmetries. Be aware that a vertex has degree three or more if and only if it is incident to at least one used chord. Therefore, the union in the second part of the objective expression (2) contains exactly all vertices of degree three or more. In the following we refer to constraint (3) also simply as *unique hamiltonicity*.

The problem of checking if a given (planar) graph contains a hamiltonian cycle is NP-complete. Therefore, the problem of checking if a given (planar) graph contains no hamiltonian cycle is in coNP. If coNP $\neq$ NP this would also imply that the latter problem is not in NP. Thus, checking only feasibility in our model is already a hard problem.

3 General Variable Neighborhood Search with Delayed Feasibility Checking

In this section we will present our algorithmic approach to tackle the optimization problem described in Sect. 2. As mentioned checking only feasibility of an instance in our model is already a hard problem. Therefore, the use of heuristics, instead of an exact algorithm, appears appropriate. We propose a *general variable neighborhood search* (GVNS) as framework to solve the problem heuristically [13,14] and combine it with Concorde [7] for checking feasibility w.r.t. unique hamiltonicity. Remember that a solution is represented by the set of chords H and the associated coloring c for each chord in H. We start with the empty initial solution $H=\emptyset$, i.e., just the cycle C without any chords, as this is always a feasible solution.

The GVNS contains a *variable neighborhood descent* (VND) for locally improving candidate solutions in systematic ways according to five different types of neighborhood structures, and a parameterized shaking neighborhood structure for diversifying the search. In the following we describe these neighborhood structures and the corresponding search algorithms.

3.1 VND Neighborhoods

The following types of neighborhoods and respective algorithms to search them are considered within the VND. Different specific configurations will be considered, which will be described in Sect. 3.2. In general, if an improved solution

is achieved within one neighborhood, the VND restarts with its first neighborhood structure; otherwise it continues with the next as long as one is available. When the VND terminates, a solution is obtained that is locally optimal w.r.t. all used neighborhood structures. All these neighborhoods are searched in a best-improvement fashion, and ties are broken randomly.

Changing Color of k Chords [cchol(k)]. This neighborhood consists of all solutions where the colors of k chords are flipped for some parameter $k \geq 1$. The size of the neighborhood is therefore m^k where m is the number of chords in the current solution. We apply this neighborhood structure for $k = 1$ and $k = 2$ since the neighborhood is relatively small and easy to search. Since only the colors of chords are changed, the structure of the solution graph stays the same, and therefore the unique hamiltonicity is still valid for each neighbor of a feasible incumbent solution. To calculate the objective gain of a neighbor incrementally we simply count the number of crossings with the old color and with the new color and take the difference.

Removing k Chords and Adding ℓ New Chords [remadd(k, ℓ)]. This neighborhood consists of all solutions where exactly k chords are removed from the current solution and ℓ new chords are added. The size of the neighborhood is therefore $m^k(M-m)^\ell$ where m is the number of chords in the current solution and M is the number of possible chords ($M = (n-2)(n-1)/2-1$). For $\ell > 0$ the neighborhood may contain solutions which are infeasible as they are not uniquely hamiltonian. Instead of checking feasibility immediately for each neighbor, which can be very time-expensive, we first evaluate all neighboring solutions according to our objective function, filter out any solutions that are not better than our incumbent, and sort all better solutions according to their objective function gain. Only then we consider the these solutions according to decreasing gain, check the feasibility of each w.r.t. unique hamiltonicity, and immediately return with the first, and thus best, feasible solution. To calculate the objective gain of a neighbor incrementally we count the crossings of the removed edges and the crossings of the added edges and take the difference. Additionally we have to check all vertices incident to removed and added edges if their degree decreased to two or increased to three or higher. In case there are multiple, i.e., equally good, best neighbors, all of these are checked for feasibility and one is chosen at random. This random tie breaking turned out to be crucial for the performance. The procedure of checking unique hamiltonicity will be described in Sect. 3.4. In case of $\ell = 0$ we do not need this check as a solution obtained from a feasible solution by just removing chords cannot become infeasible. The values used for k and ℓ will be described later.

Computing Optimal Crossings [compcross]. This is an exponentially large neighborhood consisting of all solutions that have the same underlying graph as the current solution but different colors on the chords. The size of this neighborhood is thus $2^m - 1$, where m is the number of chords in the current solution. Instead of a naive enumeration we search this neighborhood in an efficient way by

a branch-and-bound procedure to obtain a best possible coloring of the chords. The crossings of the current solution can be used as a good initial upper bound, which is in many cases already tight. In every level of the search tree we assign one color to one chord. The sequence of chords is predefined randomly and used to determine which chord is colored next. As a branching strategy we use depth-first search. To compute a local lower bound we use the crossings of the currently assigned chords and add for every not assigned chord the minimum of crossings to the assigned chords for the two colors. As already mentioned finding the optimal crossings is an NP-hard problem, but with the described branch-and-bound procedure it can be computed relatively fast compared to the effort which is needed to check for unique hamiltonicity.

Split Crossings [splitcross]. This neighborhood is a special subset of the neighborhood remadd$(2, 2)$ which tries to resolve a crossing by splitting. More precisely for every crossing pair of chords (i, j) and (k, ℓ) with $i < k < j < \ell$ we construct a new solution by removing these two edges and adding the chords (i, ℓ) and (k, j) instead. There are two exceptions: if $j = k + 1$ the edge (k, j) would be no chord and therefore this case is skipped. Similarly, case $i = 1$ and $\ell = n$ is excluded since these two vertices are already connected in the original hamiltonian cycle. If the chords (i, ℓ) or (k, j) already exist in the current solution, this neighbor is skipped. If the newly added edges do not generate other crossings and the new graph is still uniquely hamiltonian we get a solution with one crossing less. The size of this neighborhood is equal to the number of crossings in the current solution and is therefore typically a small subset of remadd$(2, 2)$.

Merge Crossings [mergecross]. This neighborhood is a special subset of the neighborhood remadd$(1, 2)$ which tries to resolve a crossing by merging the crossing point into one of the neighboring vertices. We generate up to four new solutions for every pair of crossing chords (i, j) and (k, ℓ) with $i < k < j < \ell$ by applying the following operations:

1. Remove (i, j) and add (i, k) and (k, j).
2. Remove (i, j) and add (i, ℓ) and (j, ℓ).
3. Remove (k, ℓ) and add (i, k) and (i, ℓ).
4. Remove (k, ℓ) and add (k, j) and (j, ℓ).

Cases where the edges to be added do not correspond to valid chords or where they already exist in the solution are skipped again.

3.2 VND Neighborhood Selection

In our experiments in Sect. 4, we will consider three different configurations of VND neighborhood, which are shown in Table 1. The VND considers the stated specific neighborhoods in the listed order.

Table 1. Three different neighborhood structure sets used to compare with each other

Slim set	Medium set	Thick set
1. cchol(1)	1. cchol(1)	1. cchol(1)
2. cchol(2)	2. cchol(2)	2. cchol(2)
3. remadd(1, 0)	3. remadd(1, 0)	3. remadd(1, 0)
4. remadd(0, 1)	4. remadd(0, 1)	4. remadd(0, 1)
	5. splitcross	5. splitcross
	6. mergecross	6. mergecross
	7. compcross	7. compcross
	8. remadd(1, 1)	8. remadd(1, 1)
	9. remadd(2, 1)	9. remadd(2, 1)
		10. remadd(2, 2)

It is important to notice that all neighborhoods choose the candidates randomly if their improvements are the same. This implies that the slim neighborhood set is still capable of reaching all feasible solutions. The neighborhoods which do not require rechecking unique hamiltonicity are listed before the expensive neighborhoods which require rechecking. The only exception of this rule is the compcross neighborhood which may also need significant time for larger graphs as it solves an NP-problem by branch-and-bound.

3.3 Shaking

To diversify the search, the GVNS applies the following shaking operation parameterized by $k \in \{3, \ldots, \lfloor \frac{n}{2} \rfloor\}$. Considering the chords in an order of non-increasing number of crossings, k chords are deleted from the current solution. If there are less than k chords in the current solution we delete them all. Be aware that there are in general multiple chords with the same number of crossings (most of them will have no crossings in a good solution), and they are then considered in a random order. Thus, there also is a significant randomization involved. Since we do not know in advance how many chords a solution will have, it is so far not guaranteed that in principle our GVNS can reach every possible solution from an incumbent solution. Therefore, we add as last shaking operation the removal of all chords. In other words this last shaking operation corresponds to a complete restart of the GVNS, and we trust on the VND to add chords again.

Thus, our shaking neighborhoods do not contain the complete solution space but rather a cone of all solutions we can get by removing chords from the current solution. Just removing chords in the shaking has the advantage that we will never violate unique hamiltonicity, and thus we always get a feasible solution efficiently. It would be difficult to generate a random solution in the complete feasible solution space since we would have to check for unique hamiltonicity

until we find a solution satisfying it. This would cost a lot of time which is not our intention behind shaking. However, our overall approach guarantees that every feasible solution can in principle be found by descending from one of the solutions generated by shaking. This can easily be seen by the fact that every solution can be constructed by adding chords from the empty solution.

3.4 Checking Unique Hamiltonicity

In this section we describe the procedure we apply to check if a given solution is uniquely hamiltonian, i.e., satisfies (3). The running time of this procedure is crucial for the success of the algorithm since it is the bottleneck of the GVNS as we will see in the experimental results in Sect. 4.

As we only check unique hamiltonicity for neighbors which would improve the current solution we can descend to the neighbor whenever the condition is satisfied. This means that the number of procedure calls where the condition is satisfied corresponds to the number of local improvements. However, it is possible that any number of neighbors get checked before one is found which satisfies condition (3). Therefore, we do not have a better bound on the number of negative procedure calls than the current neighborhood size.

Since the hamiltonian cycle problem is a special case of the well studied travelling salesman problem, there already exist a lot of practically efficient algorithms to approach the hamiltonian cycle problem. To model the hamiltonian cycle problem as a traveling salesman problem one simply assigns all pairs of nodes corresponding to edges in the graph, i.e., E, unit costs and all other pairs of nodes larger costs. The question whether or not a hamiltonian cycle exists is then equivalent to the question whether or not a tour with costs $|V|$ exists. If we want to fix a subset of edges $E' \subseteq E$, then we can give them zero costs. The question if a hamiltonian cycle containing all edges in E' exists is then equivalent to the question if a tour with costs $|V| - |E'|$ exists. Thus, we can also model a fixed edge hamiltonian cycle problem.

To check condition (3), more specifically we need to check if a hamiltonian cycle containing edge $(1, 2)$ and edge e for any chord $e \in H$ exists. This means, we have to solve $|H|$ traveling salesman problems before we know for sure that condition (3) is satisfied. If at some point we find a hamiltonian cycle we can stop and know that the condition is not satisfied.

Clearly, solving $|H|$ traveling salesman problems would be very time-expensive. Fortunately, we can apply an improvement in our case that takes into account that the considered candidate solution graph is a neighbor of our current solution for which we already know that it satisfies condition (3). Remember that the only situation where we have to recheck the condition is for neighbors where we removed k chords and added $\ell > 0$ new chords. In this situation it is sufficient to check if there exists a hamiltonian cycle containing the edge $(1, 2)$ and at least one of the *newly added* chords. Therefore, we only have to solve ℓ traveling salesman problems to perform this check.

As we already mentioned there may be many more negative procedure calls than positive ones and therefore we need a solver which is able to

find hamiltonian cycles fast. Thus, we decided to use a heuristic to find hamiltonian cycles. We use Helsgaun's version of the Lin-Kerninghan heuristic, which is faster than exact approaches but can still solve many problems to optimality [16]. If one run of the LKH heuristic does not find a hamiltonian cycle we apply ten further runs of the heuristic. If it still does not find a hamiltonian cycle we assume that the graph does not contain anyone. To avoid that the algorithm returns an infeasible solution as an optimal solution at the end we use Concorde [7] to check for hamiltonian cycles. As Concorde needs much more time than the Lin-Kerninghan heuristic, we only call Concorde whenever a neighbor would lead to a new best solution and the Lin-Kerninghan heuristic did not find a cycle in any run.

This means it may happen that an infeasible neighbor gets visited if it is no new best solution. In this case Lin-Kerninghan calls, where we assume that the current solution is feasible, are not correct anymore. This is no problem since as soon as the search visits a neighbor which would be a new best solution, Concorde gets applied and shows that the neighbor is infeasible. As we will see in Sect. 4 this situation will almost never happen for small vertex degrees.

Further Improvements. To further improve the running time of checking unique hamiltonicity we exploit the fact that only small parts of the graph change when doing local improvements. Obviously the same hamiltonian cycles may frequently appear in the investigated graphs having only small differences. The idea is now to store found hamiltonian cycles in an appropriate data structure which allows us to check for a new graph if it contains a cycle from the data structure efficiently. If searching through this data structure can be done reasonably fast, this will improve the overall runtime. We can represent a cycle or a whole graph by the set of its edges. Thus, we need a data structure which stores sets and can compute subset queries quickly. This problem is known as the containment query problem [6]. It is the complementary problem of the better known subset query problem, which furthermore corresponds to the well known partial match problem [18].

For our purposes we used a trie data structure presented in [3]. Note that we only store the set of chords used in the hamiltonian cycle. Checking if a subset exists in such a data structure needs exponential time in the worst case. Nevertheless, it is still much faster to search in this data structure than searching a hamiltonian cycle in practice, as we will see in Sect. 4. There would also be more sophisticated data structures for storing sets (see for example [15]). In our case we do not need such complex data structures since our practical tests indicated that the simple data structure's time consumption is almost neglectable in comparison to the effort for finding hamiltonian cycles in the remaining cases.

4 Computational Results

Our VNS-approach is implemented in C++ and compiled with g++ 4.8.4. We used the LKH-heuristic implementation provided in [17]. The Concorde implementation from [7] uses CPLEX 12.6.2 for solving. All tests were performed on

a single core of an Intel Xeon E5540 processor with 2.53 GHz and 10 GB RAM. The input of our algorithm is simply the number of vertices $n \in \mathbb{N}$. For all tests we used the weights $\alpha = 0.25$ and $\beta = 1$ (see (2)). This implies that all solution graphs with an objective value smaller than 1 have minimum degree three.

As the instances we used different n values between 10 and 100. We ran all three configurations (see Table 1) for every instance 20 times with different seed values and a maximal execution time of 3600 seconds per run. In Table 2 we see the results for the three different configurations for different instances n. The columns *best* contain the best value found in all 20 runs for one configuration. The columns *avg.* contain the averages of the results over the 20 runs and the columns $t[s]$ *med.* contain the medians of the times until the best solution was found in each run in seconds. For every instance and every of the three column types we marked the best value of the three configurations by displaying it bold.

Table 2. Results for the three different configurations

	Slim set			Medium set			Thick set		
n	*best*	*avg.*	$t[s]$ *med.*	*best*	*avg.*	$t[s]$ *med.*	*best*	*avg.*	$t[s]$ *med.*
10	**0.5**	**0.5**	56.83	**0.5**	**0.5**	37.77	**0.5**	**0.5**	**21.33**
15	**0.5**	**0.5**	5.45	**0.5**	**0.5**	**4.92**	**0.5**	**0.5**	15.77
20	**0.5**	**0.5**	13.23	**0.5**	**0.5**	**12.29**	**0.5**	**0.5**	21.71
25	**0.5**	**0.5**	50.25	**0.5**	**0.5**	**48.92**	**0.5**	**0.5**	82.42
30	**0.5**	**0.5**	234.6	**0.5**	**0.5**	**75.03**	**0.5**	**0.5**	301.92
35	**0.5**	**0.5**	**139.81**	**0.5**	**0.5**	209.68	**0.5**	**0.5**	442.44
40	**0.5**	**0.5**	369.86	**0.5**	**0.5**	**277.22**	**0.5**	0.6	426.32
45	**0.5**	0.55	1,020.1	**0.5**	**0.53**	714.29	**0.5**	0.7	1,510.86
50	**0.5**	0.74	144.2	**0.5**	**0.56**	1,007.63	**0.5**	0.81	321.41
60	**0.5**	0.93	19.52	**0.5**	**0.83**	43.22	**0.5**	0.94	99.78
70	**0.5**	0.98	35.76	**0.5**	**0.76**	238.53	**0.5**	1	306.56
80	**0.5**	0.96	71.46	**0.5**	**0.9**	68.08	1	1.2	564.71
90	0.75	1.04	128.52	**0.5**	**0.95**	130.56	1	1.46	1,260.15
100	1	1	183.33	**0.5**	**0.96**	173.95	**0.5**	1.66	570.27

As we can see, on average, the medium set found better solutions than the other two configurations. Note that the running times until the best solution was found are only comparable if the corresponding solutions are equally good. Therefore, it makes no sense to compare the time medians for the instances with $n \geq 45$. To verify that the medium solution performs better than the other two solutions we applied a Wilcoxon signed-rank test. We use a p-value of 5 % for the significance value. As a result we get that the medium set computed for the instances $n = 50, 60, 70$ significantly better results than the slim set and for all instances with $n \geq 40$ significantly better results than the thick set. For all

other instances the difference was not significant. We also compared the slim set with the thick set and interestingly the slim set computed for the instances $n = 40, 45, 80, 90, 100$ significantly better solutions than the thick set.

To compare the running times until the best solution was found we also applied the test for the running times, but only for the instances with $n \leq 40$. If we compare the medium set and the slim set there is only for the instance $n = 30$ a significant difference, where the medium set is significantly faster than the slim set. If we compare the slim or the medium set with the thick set, we get that both are for the instances with $15 \leq n \leq 35$ significantly faster than the thick set. For $n = 10$ the thick set is the fastest on average and compared to the slim set it is also significantly faster.

From these tests we get the intuition that the neighborhood remadd(2,2) is too large and not beneficial for graphs of medium size or larger sizes. Only for graphs with size around $n = 10$ the neighborhood is small enough to be beneficial. We need, however, more neighborhoods than only adding and removing chords, as in the slim set, to find good solutions for larger instances. It is also interesting that the slim set has no significant speed gain compared to the medium set. From the *best* column of the medium set we see that we found a solution with an objective of 0.5 for all given instances. This means that these solution graphs have minimum degree three and exactly two crossings in the planar embedding. We also applied some test runs for all other n values between 10 and 100 and found a solution with an objective of 0.5 for every $n \in \{10, \ldots, 100\}$.

Note that the solutions of our problem, as we stated it, are not uniquely hamiltonian, they are FEUHG. That means to get a uniquely hamiltonian graph we need to duplicate it as described in Sect. 2, but then the crossings get also duplicated. Therefore, all our solutions with an objective of 0.5 induce uniquely hamiltonian graphs with minimum degree three which have embeddings with four crossings. Therefore, our results impose the following question:

Does there exist a uniquely hamiltonian graph with minimal degree three and a crossing number smaller than four?

Table 3 contains the number of performed local search improvements for each neighborhood and each instance with the medium set configuration. The numbers are averaged over all 20 runs with 20 different seeds. The column name $ra(x, y)$ stands for remadd(x, y), *splitcr* for splitcrossing, *mergecr* for mergecrossing and *compcr* for compcrossing.

Since the shake operations only remove chords, the neighborhood remadd$(0, 1)$ which only adds one chord is applied after shaking several times. This explains why this neighborhood is used much more often than the other neighborhoods.

To find out which part of the algorithm consumes the most time we can identify three subroutines which have an exponential worst case running time. The first one is Concorde, which gets applied whenever the search finds a new best solution for which the LKH-heuristic did not find a second hamiltonian cycle. The second one is the solver of the containment query problem, which uses a trie data structure to check if a cycle which was already found is contained

Table 3. Number of improvements found in the different neighborhood structures for the medium set configuration

n	*cchol*(1)	*cchol*(2)	*ra*(1, 0)	*ra*(0, 1)	*splitcr*	*mergecr*	*compcr*	*ra*(1, 1)	*ra*(2, 1)
10	2,667	60	0	314,452	677	197	0	3,898	4
15	6,551	3,872	19	279,279	522	856	3,126	3,202	204
20	3,904	1,887	19	202,046	1,599	1,371	1,732	2,211	128
25	2,676	1,179	26	133,997	1,293	1,287	1,311	1,675	86
30	1,828	646	34	92,814	1,201	1,152	895	1,334	63
35	1,257	501	23	63,717	1,013	958	550	1,054	44
40	999	375	24	48,623	844	828	482	869	39
45	756	248	32	36,387	769	771	284	794	30
50	595	193	20	28,707	661	640	232	656	26
60	365	107	25	18,612	553	564	99	559	17
70	264	75	15	12,953	398	410	77	390	13
80	201	53	17	9,573	330	336	50	328	9
90	147	37	17	7,184	258	283	39	263	8
100	123	29	10	5,891	221	229	27	212	5

in the current candidate. The third one is the branch and bound procedure to calculate the optimal colors of the chords such that they have a minimal number of crossings.

Table 4 lists running time information and additional information for these three subroutines and the LKH subroutines with the medium set configuration. The column *HC-Checks* contains the number of graphs for which we had to test the unique hamiltonicity constraint and the *UHGs* column contains the number of graphs which satisfied the unique hamiltonicity constraint. The column *calls* contains the number of Concorde calls and the column *rate* contains the number of graphs which got discarded by a containment query relative to the overall number of discarded graphs. The columns $t[s]$ contain the overall time used for all corresponding calls in seconds. The time columns represent median values and the other columns represent average values over all different seeds. As we can see the three mentioned subroutines are in total extremely fast compared to the LKH subroutines which have to get applied very often. Therefore, the LKH subroutines are clearly the bottleneck of the whole VNS. Another interesting fact is that Concorde never found a hamiltonian cycle in any of the runs. This means that whenever LKH did not find a second hamiltonian cycle the graph was in fact uniquely hamiltonian.

The fact that the rates of the containment queries increase with increasing n can be explained as follows. First of all, for two solutions which are very similar it is more likely that they both contain the same hamiltonian cycle than for solutions which are completely different. This implies that as long as the search

Table 4. Hard subproblem statistics for the medium set configuration

	LKH			Concorde		Containment Queries		B&B
n	*HC-Checks*	*UHGs*	$t[s]$	*calls*	$t[s]$	*rate*	$t[s]$	$t[s]$
10	1,467,893	255,790	1,356	7	0	4.8 %	3	0
15	1,607,901	258,962	2,480	10	0	5.5 %	4	0
20	1,456,778	200,372	3,230	14	0	7.5 %	7	0
25	1,151,036	137,859	3,421	17	0	10.1 %	8	0
30	929,409	99,379	3,491	21	1	12.3 %	9	0
35	761,117	70,547	3,512	24	1	14.5 %	10	0
40	653,680	55,017	3,518	26	1	17.2 %	11	0
45	570,683	42,607	3,521	30	1	19.3 %	12	0
50	493,993	34,275	3,523	33	2	21.3 %	13	0
60	425,704	23,800	3,523	40	3	28.5 %	14	0
70	323,338	16,864	3,528	46	5	30.6 %	14	1
80	277,038	12,984	3,525	52	9	34.2 %	13	1
90	246,591	10,072	3,515	60	11	39.1 %	17	2
100	204,408	8,324	3,510	65	14	41.3 %	21	4

is concentrated in a local area the containment queries will be very effective. Only the shaking methods guide the search out of such a local area. The simple fact that for larger n the search in the local neighborhoods needs longer implies that it can do less shaking than for smaller n. Therefore, the containment queries are more effective for larger n.

We want to mention that we tested the algorithm also for $n > 100$. For these instances the running times of the subproblems exploded and the VNS could do only few iterations in reasonable time which lead to poor quality solutions.

5 Conclusions and Future Work

In this paper we presented a new optimization problem for finding uniquely hamiltonian graphs of minimum degree three with small crossing numbers. We proposed a general variable neighborhood search framework to solve the problem heuristically. We proposed different neighborhood structures including one large neighborhood and different configurations which we compared in experimental tests. The bottleneck of the proposed algorithm is checking unique hamiltonicity for every neighbor, to stay in the feasible area. With an implementation of this framework we were able to find uniquely hamiltonian graphs of minimum degree three with only four crossings for many different instances, which naturally gives rise to the question if we can do better.

Future work may be to develop an exact algorithm for the proposed problem and compare the two algorithms for small instances. One problem of the

proposed heuristic is that the constraint of unique hamiltonicity is completely independent from the objective function. If we could measure how promising a graph is according to the unique hamiltonicity constraint we could better guide the search. Furthermore, it would be interesting to test if other heuristics than LKH or other variants of LKH for checking unique hamiltonicity perform better.

References

1. Abbasi, S., Jamshed, A.: A degree constraint for uniquely Hamiltonian graphs. Graphs and Combinatorics **22**(4), 433–442 (2006)
2. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2011)
3. Bevc, S., Savnik, I.: Using tries for subset and superset queries. In: Proceedings of the ITI 2009, pp. 147–152 (2009)
4. Bondy, J.A., Jackson, B.: Vertices of small degree in uniquely Hamiltonian Graphs. J. Comb. Theory Ser. B **74**(2), 265–275 (1998)
5. Caporossi, G., Hansen, P.: Variable neighborhood search for extremal graphs: 1 The AutoGraphiX system. Discrete Math. **212**(1–2), 29–44 (2000)
6. Charikar, M., Indyk, P., Panigrahy, R.: New algorithms for subset query, partial match, orthogonal range searching, and related problems. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 451–462. Springer, Heidelberg (2002)
7. Cook, W.: Concorde TSP Solver (2011). http://www.math.uwaterloo.ca/tsp/concorde/. Accessed on 31 Jan 2016
8. Entringer, R.C., Swart, H.: Spanning cycles of nearly cubic graphs. J. Comb. Theory Ser. B **29**(3), 303–309 (1980)
9. Fleischner, H.: Uniqueness of maximal dominating cycles in 3-regular graphs and of Hamiltonian cycles in 4-regular graphs. J. Graph Theory **18**(5), 449–459 (1994)
10. Fleischner, H.: Uniquely Hamiltonian graphs of minimum degree 4. J. Graph Theory **75**(2), 167–177 (2014)
11. Garey, M., Johnson, D.: Crossing number is NP-complete. SIAM J. Algebraic Discrete Methods **4**(3), 312–316 (1983)
12. Gould, R.J.: Advances on the Hamiltonian problem-a survey. Graphs and Combinatorics **19**(1), 7–52 (2003)
13. Hansen, P., Mladenović, N.: An introduction to variable neighborhood search. In: Voss, S., et al. (eds.) Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization, pp. 433–458. Kluwer, Dordrecht (1999)
14. Hansen, P., Mladenović, N.: A tutorial on variable neighborhood search. Technical report G-2003-46, GERAD, July 2003
15. Helmer, S., Aly, R., Neumann, T., Moerkotte, G.: Indexing set-valued attributes with a multi-level extendible hashing scheme. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 98–108. Springer, Heidelberg (2007)
16. Helsgaun, K.: Effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur. J. Oper. Res. **126**(1), 106–130 (2000)
17. Helsgaun, K.: LKH (2012). http://www.akira.ruc.dk/~keld/research/LKH/. Accessed 03 Feb 2016
18. Jayram, T.S., Khot, S., Kumar, R., Rabani, Y.: Cell-probe lower bounds for the partial match problem. In: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, STOC 2003, pp. 667–672. ACM, New York, (2003)

19. Klocker, B., Raidl, G.: Finding uniquely hamiltonian graphs with minimal degree three. Technical report, Algorithms and Complexity Group, TU Wien (2016)
20. Pisinger, D., Ropke, S.: Large neighborhood search. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics, pp. 399–419. Springer US, London (2010)
21. Sheehan, J.: The multiplicity of Hamiltonian circuits in a graph. In: Recent Advances in Graph Theory, pp. 477–480 (1975)
22. Thomason, A.G.: Hamiltonian cycles and uniquely edge colourable graphs. Ann. Discrete Math. **3**, 259–268 (1978). Advances in Graph Theory