Computing Sharp 2-factors in Claw-free Graphs* **

Hajo Broersma and Daniël Paulusma

Department of Computer Science, Durham University, DH1 3LE Durham, United Kingdom, {hajo.broersma,daniel.paulusma}@durham.ac.uk

Abstract. In a previous paper we obtained an upper bound for the minimum number of components of a 2-factor in a claw-free graph. This bound is sharp in the sense that there exist infinitely many claw-free graphs for which the bound is tight. In this paper we extend these results by presenting a polynomial algorithm that constructs a 2-factor of a claw-free graph with minimum degree at least four whose number of components meets this bound. As a byproduct we show that the problem of obtaining a minimum 2-factor (if it exists) is polynomially solvable for a subclass of claw-free graphs. As another byproduct we give a short constructive proof for a result of Ryjáček, Saito & Schelp.

1 Introduction

In this paper we consider 2-factors of claw-free graphs. Graph factors are well-studied. See [16] for a survey. Our motivation to study 2-factors goes back to the well-known NP-complete decision problem H-CYCLE (cf. [9]) in which the problem is to decide whether a given graph has a hamiltonian cycle, i.e., a connected 2-regular spanning subgraph. In the related problem 2-FACTOR the connectivity condition is dropped, hence the problem is to decide whether a given graph admits a 2-factor, i.e., a 2-regular spanning subgraph. This makes the problem considerably easier in the algorithmic sense: it is well-known that 2-FACTOR can be solved in polynomial time by matching techniques, and a 2-factor can be constructed in polynomial time if the answer is YES (cf [14]). Clearly, a hamiltonian cycle is a 2-factor consisting of one component, and the minimum number of components of a 2-factor can be seen as a measure for how far a graph is from being hamiltonian. So, from an algorithmic viewpoint a natural question is to consider the problem of determining a 2-factor of a given graph with a minimum number of components. Obviously, this is an NP-hard problem. Hence it makes sense to search for 2-factors with a reasonably small number of components if we aim for polynomial time algorithms. For this research we have restricted ourselves to the class of claw-free graphs. This is a rich class containing, e.g., the class of line graphs and the class of complements of triangle-free graphs. It is also a very well-studied graph class, both within structural graph theory and within algorithmic graph theory; see [7] for a survey. Furthermore, computing a 2-factor with a minimum number of components remains NP-hard for the class of claw-free graphs.

* This work has been supported by EPSRC (EP/D053633/1).

** A preliminary and shortened version of this paper appeared in the Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2008).

In [1] we already obtained an upper bound for the minimum number of components of a 2-factor in a claw-free graph. This bound is sharp in the sense that there exist infinitely many claw-free graphs for which the bound is tight; we will specify this later. When considering the related complexity problems, we soon realized that the proof methods used in [1] need to be extended in order to obtain a polynomial algorithm that constructs a corresponding 2-factor, e.g., a 2-factor whose number of components is at most our upper bound. In the present paper we present this polynomial time algorithm.

2 Terminology and Background

We consider graphs that are finite, undirected and simple, i.e., without multiple edges and loops. For notation and terminology not defined in this paper we refer to [4].

Let $G = (V(G), E(G))$ be a graph of order $|G| = |V(G)| = n$ and of size $e(G) = |E(G)|$. The neighbor set of a vertex x in G is denoted by $N_G(x) = \{y \in V(G) \mid xy \in E(G)\}$, and its cardinality by $d_G(x)$. We denote the minimum (vertex) degree of G by $\delta_G = \min\{d_G(x) \mid x \in V(G)\}$. If no confusion can arise we often omit the subscripts.

Let K_n denote the complete graph on n vertices. A graph F is called a *2-factor* of a graph G if F is a 2-regular spanning subgraph of G, i.e., if F is a subgraph of G with $V(F) = V(G)$ and $d_F(x) = 2$ for all $x \in V(F)$. A *claw-free* graph is a graph that does not contain an induced subgraph isomorphic to the four-vertex star $K_{1,3} = (\{u, a, b, c\}, \{ua, ub, uc\})$.

2.1 Known Results

Several interesting problems are still open for claw-free graphs such as the conjecture of Matthews and Sumner [15] that every 4-connected claw-free graph is hamiltonian. However, there is quite a lot known on 2-factors in claw-free graphs, including some very recent results. Results of both Choudum & Paulraj [3] and Egawa & Ota [5] imply that every claw-free graph with $\delta \geq 4$ contains a 2-factor.

Theorem 1 ([3, 5]). *A claw-free graph with $\delta \geq 4$ has a 2-factor.*

We observe that every 4-connected claw-free graph has minimum degree at least four, and hence has a 2-factor. A 2-connected claw-free graph already has a 2-factor if $\delta = 3$ [20]. However, in general a claw-free graph with $\delta \leq 3$ does not have to contain a 2-factor. Examples are easily obtained.

Faudree et al. [6] showed that every claw-free graph with $\delta \geq 4$ has a 2-factor with at most $6n/(\delta + 2) - 1$ components. Gould & Jacobson [11] proved that, for every integer $k \geq 2$, every claw-free graph of order $n \geq 16k^3$ with $\delta \geq n/k$ has a 2-factor with at most k components. Fronček, Ryjáček & Skupień [8] showed that, for every integer $k \geq 4$, every claw-free graph G of order $n \geq 3k^2 - 3$ with $\delta \geq 3k - 4$ and $\sigma_k > n + k^2 - 4k + 7$ has a 2-factor with at most $k - 1$ components. Here σ_k denotes the minimum degree sum of any k mutually nonadjacent vertices.

If a graph G is claw-free, 2-connected and has $\delta \geq 4$, then G has a 2-factor with at most $(n+1)/4$ components [13]. If a graph G is claw-free, 3-connected and has $\delta \geq 4$, then G has a 2-factor with at most $2n/15$ components [13].

In [1] we considered claw-free graphs with $\delta \geq 4$. Our motivation for this is as follows. We first note that the number of components of a 2-factor in any graph on n vertices is obviously at most $n/3$. For claw-free graphs with $\delta = 2$ that have a 2-factor we cannot do better than this trivial upper bound. This is clear from considering a disjoint set of triangles (cycles on three vertices). For claw-free graphs with $\delta = 3$ that have a 2-factor, the upper bound $n/3$ on its number of components is also tight, as shown in [1]. Hence, in order to get a nontrivial result it is natural to consider claw-free graphs with $\delta \geq 4$.

Our two main results in [1] provide answers to two open questions posed in [20].

Theorem 2 ([1]). *A claw-free graph G on n vertices with $\delta \geq 5$ has a 2-factor with at most $(n-3)/(\delta-1)$ components unless G is isomorphic to K_n.*

Theorem 3 ([1]). *A claw-free graph G on n vertices with $\delta = 4$ has a 2-factor with at most $(5n-14)/18$ components, unless G belongs to a finite class of exceptional graphs.*

Both results are tight in the following sense. Let $f_2(G)$ denote the minimum number of components in a 2-factor of G. Then in [20], for every integer $d \geq 4$, an infinite family $\{F_i^d\}$ of claw-free graphs with $\delta(F_i^d) \geq d$ is given such that $f_2(F_i^d) > |F_i^d|/d \geq |F_i^d|/\delta(F_i^d)$. This shows we cannot replace $\delta - 1$ by δ in Theorem 2. The bound in Theorem 3 is tight in the following sense. There exists an infinite family $\{H_i\}$ of claw-free graphs with $\delta(H_i) = 4$ such that

$$\lim_{|H_i|\to\infty} \frac{f_2(H_i)}{|H_i|} = \frac{5}{18}.$$

This family can be found in [20] as well.

The exceptional graphs of Theorem 3 have at most seventeen vertices. They are described in [1], and we will not specify them here. In [1] we also explain that Theorem 2 and 3 together improve the previously mentioned result of Faudree et al. [6] and that Theorem 2 also improves the previously mentioned result of Gould & Jacobson [11].

2.2 Results of This Paper

The proofs in [1] do not yield algorithms for constructing 2-factors that satisfy the upper bounds in Theorems 2 and 3. In the remainder of this paper we will develop a new approach to these problems in order to establish polynomial algorithms that construct 2-factors of claw-free graphs with minimum degree at least four. Using our results in [1] we show that the number of components in these 2-factors are guaranteed to satisfy the upper bounds of Theorems 2 and 3. We will illustrate our approach by concentrating on Theorem 2. It will be immediately clear that the same approach works for Theorem 3 in exactly the same way. As a byproduct we show that the problem of obtaining a minimum 2-factor (if it exists) is polynomially solvable for a subclass of claw-free graphs which we describe later on. As another byproduct we give a short constructive proof for a result of Ryjáček, Saito & Schelp [19].

3 The Algorithm for Constructing 2-factors of Claw-free Graphs

We split our polynomial time algorithm into six different parts. For the first two parts we do not have to develop any new theory or algorithms, but can rely on the beautiful existing machinery from the literature. The first part of this says that claw-free graphs behave the same with respect to our problem as line graphs obtained from them by performing some closure operation which will be explained shortly. The second part then describes the known equivalence of our problem with an analogous problem based on concepts and results in the preimage graphs of line graphs. Our new contributions are described and explained in the third, fourth, fifth and sixth part. In the third part we consider preimage graphs that are trees and in the fourth part we consider preimage graphs that are triangle-free. Finally, in the fifth and sixth part we translate the results back to the original domain of claw-free graphs and mention some special class for which our algorithm finds a 2-factor with a minimum number of components.

Step 1: restrict to line graphs of triangle-free graphs

The *line graph* of a graph H with edges $e_1, \ldots, e_p$ is the graph $L(H)$ with vertices $u_1, \ldots, u_p$ such that there is an edge between any two vertices u_i and u_j if and only if i and e_j share one end vertex in H. It is easy to verify and well-known (see e.g. [12]) that line graphs are claw-free graphs, but that the class of claw-free graphs is much richer (in fact, line graphs have been characterized by a set of nine forbidden induced subgraphs). We show that we can restrict ourselves to an even smaller subclass of claw-free graphs, namely the class of line graphs of triangle-free graphs. For this purpose we use the *closure* concept as defined in [18].

The closure of a claw-free graph is defined as follows. Let G be a claw-free graph. Then, for each vertex x of G, the set of neighbors of x in G induces a subgraph with at most two components. If this subgraph has two components, both of them must be cliques. If the subgraph induced by $N(x)$ is connected, we add edges joining all pairs of nonadjacent vertices in $N(x)$. This operation is called the *local completion of G at x*. The *closure* $cl(G)$ of G is a graph we can obtain by recursively repeating the local completion operation, as long as this is possible. Ryjáček [18] showed that the closure of G is uniquely determined, i.e., that the ordering in which one performs the local completions does not matter. Ryjáček [18] also showed that G is hamiltonian if and only if $cl(G)$ is hamiltonian. This result was later extended to 2-factors [19].

Theorem 4 ([19]). *Let G be a claw-free graph. Then G has a 2-factor with at most k components if and only if $cl(G)$ has a 2-factor with at most k components.*

The following relationship between claw-free graphs and triangle-free graphs exists.

Theorem 5 ([18]). *If G is a claw-free graph, then there is a triangle-free graph H such that $L(H) = cl(G)$.*

It is well-known that apart from K_3 which is $L(K_3)$ and $L(K_{1,3})$, every connected line graph F has a unique H with $F = L(H)$ (see e.g. [12]). We call H the *preimage graph* of F. For K_3 we let $K_{1,3}$ be its preimage graph. For disconnected graphs we define the preimage graphs according to their components.

Recall that $f_2(G)$ denotes the minimum number of components in a 2-factor of a graph G. By Theorem 4 and Theorem 5, we deduce that for a claw-free graph G, $f_2(G) = f_2(cl(G)) = f_2(L(H))$, where H is the (triangle-free) preimage graph of $cl(G)$. Recall that the closure of a claw-free graph can be obtained in polynomial time. Since it is known that the preimage graph of a line graph can be obtained in polynomial (linear) time (see e.g. [17]) we can efficiently compute H.

Step 2: translate the problem into finding dominating systems

An *even* graph is a graph in which every vertex has a nonzero even degree. A connected even graph is called a *circuit*. For $q \geq 2$, a *star* $K_{1,q}$ is a complete bipartite graph with sets $A = \{c\}$ and B with $|B| = q$; the vertex c is called the *center* and the vertices in B are called the *leaves* of $K_{1,q}$.

Let H be a graph that contains a set $\mathcal{S}$ consisting of stars with at least three edges and circuits, all (stars and circuits) mutually edge-disjoint. We call $\mathcal{S}$ a *system that dominates* H or simply a *dominating system* if for every edge e of H the following holds:

- e is contained in one of the stars of $\mathcal{S}$, or
- e is contained in one of the circuits of $\mathcal{S}$, or
- e shares an end vertex with an edge of at least one of the circuits in $\mathcal{S}$.

Gould & Hynds [10] proved the following result.

Theorem 6 ([10]). *The line graph $L(H)$ of a graph H has a 2-factor with k components if and only if H has a dominating system with k elements.*

Combining Theorem 4 and Theorem 5 with Theorem 6 yields the following result.

Theorem 7. *Let G be a claw-free graph. Then G has a 2-factor with k components if and only if the (triangle-free) preimage graph of G has a dominating system with k elements.*

The *edge degree* of an edge xy in a graph H is defined as $d_H(x) + d_H(y) - 2$. We denote the minimum edge degree of H by $\delta_e = \delta_e(H)$. Due to the previous discussions it is clear that Theorem 2 is equivalent to the following theorem.

Theorem 8. *A triangle-free graph H with $\delta_e(H) \geq 5$ has a dominating system with at most $(e(H) - 3)/(\delta_e(H) - 1)$ elements unless H is isomorphic to $K_{1,e(H)}$.*

We will now concentrate on determining (in polynomial time) a *sharp dominating system*, i.e., one that satisfies the upper bound of Theorem 8. We first deal with the case that H is a tree. In this case we can even determine a minimum dominating system in polynomial time.

Step 3: compute minimum dominating systems for trees

Here we present a polynomial time algorithm for computing the number of elements in a minimum dominating system of any given tree. We use the following new terminology. A *minimum dominating system*, or shortly, an m*-system* of a graph H is a dominating system of H with the smallest number of elements. We denote such a system by $\mathcal{M}(H)$, and its number of elements by $m(H)$. If H does not allow a dominating system we write $m(H) = \infty$.

A vertex with degree 1 in a graph F is called an *end vertex* or *leaf* of F. An edge which is incident with a leaf is called a *pendant* edge. We say that we *add a pendant edge* to F if we add a new vertex to F and join it to precisely one of the vertices of F. Two edges are called *independent* if they do not share any end vertices. A *matching* is a set of mutually independent edges.

We write $H^q(w)$ to denote a tree H that contains a vertex w to which we added q new pendant edges. Note that $H^0(w) = H$. Let $H_1, \ldots, H_p$ be a set of p mutually vertex-disjoint trees, where each H_i contains a vertex w_i. We say that we have *joined* trees $H_1, \ldots, H_p$ *in* $w_1, \ldots, w_p$ *by* u if we add a new vertex u with edges uw_i for $i = 1, \ldots, p$. If $p = 0$, then the resulting tree $H(u)$ is the single vertex u, which has a dominating system of 0 elements by definition. Before we present our algorithm we first deduce a number of equations. Note that $m(H^1(w)) = \infty$ if $H = (\{w\}, \emptyset)$.

Lemma 1. *Let $w_1, \ldots, w_p$ be a set of p vertices belonging to mutually disjoint trees $H_1, \ldots, H_p$, respectively. Let $H(u)$ be the tree obtained after joining $H_1, \ldots, H_p$ in $w_1, \ldots, w_p$ by u. Then* $m(H(u)) =$

$$\begin{cases} 0 & if\ p = 0 \\ \sum_{i=1}^{p} m(H^1(w_i)) & if\ p \in \{1, 2\} \\ \min\Big\{ \sum_{i=1}^{p} m(H^1(w_i)), & \\ 1 + \min\limits_{i_1 < i_2 < i_3} \big\{ \sum_{j=1}^{3} m(H_{i_j}) + \sum\limits_{i \notin \{i_1, i_2, i_3\}} \min\{m(H_i), m(H^1(w_i))\} \big\} \Big\} & if\ p \geq 3. \end{cases}$$

Proof. We prove each case separately.

• Let $p = 0$. Then $H(u) = (\{u\}, \emptyset)$. So, $m(H(u)) = 0$ by definition of a dominating system.

• Let $1 \leq p \leq 2$. Then, in any dominating system of $H(u)$, u is not a star center, and consequently, each w_i is the center of a star containing the edge uw_i. Note that in each tree $H^1(w_i)$, w_i is a star center (because the new pendant edge to w_i needs to be covered by a star). Hence, we can combine any m-systems $\mathcal{M}^1(w_i)$ of each $H^1(w_i)$ to obtain an m-system $\mathcal{M}(H(u))$ with $\sum_{i=1}^{p} m(H^1(w_i))$ elements.

• Let $p \geq 3$. First consider the set of dominating systems of $H(u)$ in which u is not a star center. In all these dominating systems, each w_i is the center of a star containing the edge uw_i. Similar to the previous case, we can combine any m-systems of each

$H^1(w_i)$ to obtain a dominating system $\mathcal{S}$ of $H(u)$ with $\sum_{i=1}^{p} m(H^1(w_i))$ elements. We note that $\mathcal{S}$ has the minimum number of elements over all dominating systems of $H(u)$ in which u is not a star center.

Secondly, consider the set of dominating systems of $H(u)$ in which u is a star center. In all these dominating systems, the star with center u contains at least three edges, say uw_{i_1}, uw_{i_2}, and uw_{i_3}, by definition of a dominating system. For the remaining edges uw_i we act as follows. In each dominating system of $H(u)$ that has a star with center u, such an edge uw_i either belongs to the star with center u, or else to the star with center w_i. We compute an m-system $\mathcal{M}(H_i)$ and an m-system $\mathcal{M}(H^1(w_i))$. Then we choose the one with the smallest number of elements, which we denote by $\mathcal{M}_i^*$. So, $|\mathcal{M}_i^*| = \min\{m(H_i), m(H^1(w_1)\}$. We now combine the m-systems of H_{i_j} for $j = 1, 2, 3$, together with the dominating systems $\mathcal{M}_i^*$ and a star that contains the edges uw_{i_j} for $j = 1, 2, 3$ plus possibly some more edges depending on our choice for each $\mathcal{M}_i^*$. We try all possible triples (i_1, i_2, i_3), and choose the combination with the smallest total number of elements. This way we obtain a dominating system $\mathcal{S}'$ of $H(u)$ that has

$$1 + \min_{i_1<i_2<i_3} \Big\{ \sum_{j=1}^{3} m(H_{i_j}) + \sum_{i \notin \{i_1,i_2,i_3\}} \min\{m(H_i), m(H^1(w_i)\}\Big\}$$

elements. We note that $\mathcal{S}'$ has the minimum number of elements over all dominating systems of $H(u)$ in which u is a star center.

Finally, we compare the numbers of elements of $\mathcal{S}$ and $\mathcal{S}'$, and we choose (the) one with the smallest number of elements. This yields an m-system $\mathcal{M}(H(u))$. □

Using Lemma 1 we can prove the following theorem.

Theorem 9. *The problem of finding a minimum dominating system is polynomially solvable for the class of trees.*

Proof. Let H be a tree with a designated vertex v^0. We partition $V(H)$ into $L_0 \cup L_1 \cup \ldots \cup L_r$ such that for $i = 0, \ldots, r$, L_i is the set of vertices at distance i from v^0. Note that $L_0 = \{v^0\}$. For $v \in V(H)\backslash\{v^0\}$, we let $v^+ \in N(v)$ be the first vertex on the (unique) path from v to v^0 in H, and we let the subtree H_v be the component of $H - vv^+$ that contains v.

Now let $v \in V(H)$. Suppose v has neighbors $w_1, \ldots, w_p$ in H_v. Then H_v is obtained after joining the p mutually disjoint trees $H_{w_1}, \ldots, H_{w_p}$ in $w_1, \ldots, w_p$ by v. Suppose we have already computed the values $m(H_{w_i})$ and $m(H^1_{w_i}(w_i))$. Then, using Lemma 1, we can easily compute $m(H_v)$. We observe that the tree $H^1_v(v)$ is obtained after joining the trees $H_{w_i}, \ldots, H_{w_p}$ together with a new single vertex tree $(\{w_{p+1}\}, \emptyset)$ in $w_1, \ldots, w_{p+1}$ by v. Hence, we can use Lemma 1 to compute $m(H^1_v(v))$ as well. So, our strategy is to recursively compute the values $m(H_v)$ and $m(H^1_v(v))$: for $i = 1, \ldots, r$, we first compute the values $m(H_{v^i})$ and $m(H^1_{v^i}(v^i))$ for all $v^i \in L_i$, and use them to compute $m(H_{v^{i-1}})$ and $m(H^1_{v^{i-1}}(v^{i-1}))$ for all $v^{i-1} \in L_{i-1}$ according to Lemma 1. Clearly, computing $m(H)$ this way can be done in polynomial time.

In order to find an m-system $\mathcal{M}(H)$, we keep track of the stars as follows. Firstly, for each $v \in V(H)$, we remember whether v is a star center in an m-system of H_v

when we compute $m(H_v)$. In case v is the center of a star S_v, we keep track of the edges in S_v as well. Secondly, we check whether v becomes the center of a star S_v (and which edges belong to S_v if S_v exists) both when we compute $m(H_{v^+})$ and when we compute $m(H^1_{v^+}(v^+))$. Note that we can do this in polynomial time when we use the formula in Lemma 1. With the above information we can efficiently compute an m-system $\mathcal{M}(H)$, as the following claim shows.

Claim. For all v in each L_i we can compute in polynomial time whether v is the center of a star S_v of an m-system $\mathcal{M}(H)$ and, if so, which edges of H belong to S_v.

We prove this claim by induction on i. Let $i = 0$. When we computed the value for $m(H_{v^0}) = m(H)$ by using Lemma 1, we checked whether v^0 is the center of a star in an m-system of H. In case v^0 is the center of such a star S_{v^0}, we also remembered which edges belong to S_{v^0}.

Now suppose $i \geq 1$. Let $v \in L_i$. By the induction hypothesis, we know whether v^+ is the center of a star in an m-system $\mathcal{M}(H)$ or not. First suppose v^+ is not the center of a star in an m-system $\mathcal{M}(H)$. Then v is the center of a star S_v in $\mathcal{M}(H)$, and S_v is a star in an m-system $\mathcal{M}(H^1_v(v))$ as well. So, we kept track of S_v. Now suppose v^+ is the center of a star S_{v^+} in an m-system $\mathcal{M}(H)$. By the induction hypothesis, we know which edges S_{v^+} has. Then there are two cases: either vv^+ belongs to S_{v^+}, or it does not. If vv^+ belongs to S_{v^+}, then v is the center of a star S_v in $\mathcal{M}(H)$ if and only if S_v is a star in $\mathcal{M}(H_v)$. If vv^+ does not belong to S_{v^+}, then v is the center of a star S_v in $\mathcal{M}(H)$, and S_v is a star in an m-system $\mathcal{M}(H^1_v(v))$. In both cases we kept track of all the edges of S_v. □

Step 4: compute sharp dominating systems for general triangle-free graphs

Suppose G is a claw-free graph. Let H be the preimage of $cl(G)$, i.e., the triangle-free graph with $L(H) = cl(G)$. We now assume that H is not a tree. If H is a forest we apply Theorem 9 on each of its components, which are trees. Below we discuss the case in which H is not a forest but contains one or more circuits.

The key idea behind our approach in this case is to start with an even subgraph X of H, then to "break" the circuits in X by removing a number of edges, such that we obtain a new graph H^* that is a forest. Then we can apply our approach from the previous section to each component of H^* if we first add sufficiently many pendant edges to ensure that each component has minimum edge-degree at least $\delta_e(H)$. In this procedure we have to add more edges than we remove. However, we will have the following advantage. The added pendant edges have to be dominated by (extra) stars in any dominating system of H^*, and these stars can be merged together into fewer elements of a dominating system in the original graph H. In other words, the larger number of stars we get by applying the upper bounds to H^* will provide the necessary compensation for the larger number of edges that we created. This way we are able to establish our upper bound for H. In [1] we gave a nonconstructive proof to show that this approach works. This proof in [1] was based on a number of assumptions on the choice of the even subgraph X of H. Here we follow an alternative approach which enables a constructive proof.

Let X be an even subgraph of H with set of components $\mathcal{C}$. For each $C \in \mathcal{C}$ we do as follows. First suppose C is isomorphic to a complete bipartite graph $K_{2,2k}$ for some $k \geq 1$. Let $A(C) = \{s, t\}$ and $B(C) = \{s_1, s_2, \ldots, s_{2k}\}$ be the partition classes of C. If $k = 1$, we choose edges ss_1 and ts_2. If $k \geq 2$, we choose the $2k$ edges ss_i ($i = 1, \ldots, 2k$). If C is not isomorphic to some $K_{2,2k}$, we choose at most one (arbitrary) edge from C. We call the set of all chosen edges an *X-set* and denote it by M. Let $H^* = (H - E(X)) \cup M$. We call H^* an *X-graph* of H.

Lemma 2. *Let H be a triangle-free graph that is not a forest. We can compute in polynomial time an X-graph of H that is a forest.*

Proof. We present the following polynomial time algorithm that has H as input and that outputs an X-graph of H that is a forest.

CREATE-A-FOREST

Phase 1. Construct an even subgraph X' of H by adding mutually edge-disjoint cycles to X' until this is not possible anymore.

Phase 2. Choose an X'-set M' and check whether its X'-graph H' is a forest. If H' is a forest, output H'. If not go to Phase 3.

Phase 3. Let D be a cycle in H'. Let $\mathcal{C}^*$ be the set of circuits of X' that share an edge with D. Consider each $C \in \mathcal{C}^*$.

Case 1. C shares exactly one edge e with D.
Remove e from $X' \cup D$.

Case 2. C is isomorphic to some $K_{2,2k}$ and shares exactly two edges e, e' with D.
Let $A(C) = \{s, t\}$ and $B(C) = \{s_1, s_2, \ldots, s_{2k}\}$ be the partition classes of C.

If $k = 1$, then we may without loss of generality assume $e = ss_1$ and $e' = ts_2$. Remove ss_2 and ts_1 from $X' \cup D$.

If $k = 2$, then we may without loss of generality assume $e = ss_1$ and $e' = ss_2$. Remove ss_1 and ss_2 from $X' \cup D$.

After dealing with all circuits in $\mathcal{C}^*$, we have obtained a subgraph Y' of $X' \cup D$. Go to Phase 2 with Y' instead of X'.

In order to show that this algorithm is correct and runs in polynomial time, we start with making four observations. First, a set $\mathcal{C}^*$ in Phase 3 is nonempty, as otherwise $X' \cup D$ would be an even subgraph of H with more edges than X'. Second, by definition of an X-set, Case 1 and Case 2 are the only cases we have to consider in Phase 3. Third, by the construction, a subgraph Y' obtained in Phase 3 is indeed an even subgraph of H, and fourth, $|V(Y')| \geq |V(X')|$ as we did not remove any vertices from $X' \cup D$. We claim that either Y' contains fewer components than X' or else $|V(Y')| > |V(X')|$. As each phase is performed in polynomial time, the algorithm will then terminate in polynomial time with as output some X-graph H^* that is a forest.

The above claim can be verified as follows. Suppose Y' does not contain fewer components. We observe that all vertices of D belong to the same circuit of Y'. As $\mathcal{C}^* \neq \emptyset$, we then find that Y' cannot contain more components than X'. Hence Y' must have the same number of components as X'. This means that $|\mathcal{C}^*| = 1$, say $\mathcal{C}^* = \{C\}$

and that there are no components of X' that only share vertices with D. If C shares only one edge with D, then $|V(Y')| > |V(X')|$ as $|D| \geq 4$. In the other case, C is isomorphic to some $K_{2,2k}$ and shares (exactly) two edges with D. Suppose $k = 1$. If D contains exactly four edges, then D is a four-cycle on the same four vertices as C. Then H contains an induced K_4. As H is triangle-free, this is not possible. Hence, D contains more than four edges. This implies that $|V(Y')| > |V(X')|$. Finally, suppose $k \geq 2$. In that case, C shares two vertices with D that have a common end-vertex. Hence, also here we find that $|V(Y')| > |V(X')|$ as $|D| \geq 4$. □

Note that the above result implicitly implies that we can not only compute the desired X-graph H^* of H in polynomial time but also the corresponding even subgraph X.

Theorem 10. *Let H be a triangle-free graph not isomorphic to $K_{1,e(H)}$ such that $\delta_e(H) \geq 5$. Then a dominating system of H with at most $(e(H) - 3)/(\delta_e(H) - 1)$ elements can be found in polynomial time.*

Proof. Let H be a triangle-free graph that is not isomorphic to $K_{1,e(H)}$ and that has $d = \delta_e(H) \geq 5$. Recall that we can apply Theorem 9 on each component of H if H is a forest. This way we even get a minimum dominating system of H, which satisfies the desired upper bound due to Theorem 8.

Suppose H is not a forest. By Lemma 2, we compute in polynomial time an X-graph H^* of H that is a forest. It can happen that H^* does not have minimum edge degree at least d. In the proof of Theorem 8 of [1], we therefore modified H^* into a new forest H' by adding some new pendant edges and removing some vertices. This proof can be rewritten in constructive form. In order to show the polynomial upper bound on the running time we include this algorithm, called COMPUTE-A-DOMINATING-SYSTEM, below. It has as input H together with X as computed by Lemma 2. From its description it will be directly clear that it runs in polynomial time indeed. For its correctness we refer to [1].

We first give some terminology. Let $V_1(H)$ be the degree one vertices of H and $\mathcal{C}$ be the set of circuits of X. For each $C \in \mathcal{C}$ we partition $V(C)$ into two sets $I(C) \cup J(C)$, where $I(C)$ denotes the set of vertices in C that are only adjacent to vertices in $V(C) \cup V_1(H)$ and $J(C)$ denotes the set $V(C) \backslash I(C)$. Note that $J(C) = \emptyset$ for some $C \in \mathcal{C}$ implies that the component of H containing C consists of vertices of $V(C) \cup V_1(H)$ only. Our algorithm makes use of a subroutine called ADD-AND-REMOVE. The latter algorithm considers each circuit in $\mathcal{C}$ and decides which vertices of C should be deleted and to which vertices new pendant edges should be added. We give its description after presenting the main algorithm.

COMPUTE-A-DOMINATING-SYSTEM

Let $\mathcal{C}' = \{C \in \mathcal{C} \mid J(C) = \emptyset\}$. If $\mathcal{C}' = \mathcal{C}$ then output $\mathcal{C}$. Otherwise, delete the components that contain the circuits in $\mathcal{C}'$ from H, and perform the algorithm ADD-AND-REMOVE on each $C \in \mathcal{C} \backslash \mathcal{C}'$. Denote the resulting forest by H'.

Apply Theorem 9 on each component of H' in order to obtain a minimum dominating system $\mathcal{S}'$ of H'.

Let $\mathcal{S}'(C)$ be the set of stars that dominate the remaining vertices of C in $\mathcal{S}'$. For each $C \in \mathcal{C}$ replace the stars in $\mathcal{S}'(C)$ by C. Keep all other elements of $\mathcal{S}'$. Add $\mathcal{C}'$. Output the resulting dominating system $\mathcal{S}$.

Below we state the algorithm ADD-AND-REMOVE that has as input a circuit $C \in \mathcal{C}$ with $J(C) \neq \emptyset$. For convenience, we write $I = I(C)$ and $J = J(C)$. Furthermore, let $d^*(u)$ denote the number of edges incident with a vertex $u \in I$ in the subgraph of H obtained from $H[V(C) \cup V_1(H)]$ after removing $E(C)$. Let J^* be the subset of J that consists of all vertices u with $d^*(u) \geq d$.

ADD-AND-REMOVE

Case 1. $J^* \neq \emptyset$.
Remove all edges of C and all vertices of I together with their possible neighbors in $V_1(H)$. To each $u \in J^*$ add one new pendant edge, and to each $u \in J \backslash J^*$ add $d - d^*(u) + 1$ new pendant edges.

Case 2. $J^* = \emptyset$ and $|J| = 1$.
Let $J = \{u\}$. Remove all vertices of C except u. Add d new pendant edges to u.

Case 3. $J^* = \emptyset$ and $|J| \geq 2$.
Let $u_1, u_2 \in J$. Let v_1, v_2 be two vertices in C such that u_1v_1 and u_2v_2 are independent.

Case 3a. $C = u_1v_1u_2v_2u_1$.
Remove u_1v_2 and v_1u_2. For each $u \in C$, add $d - d(u) + 2$ new pendant edges.

Case 3b. $|C| \geq 5$ and $v_1, v_2 \in I$.
Remove all edges of C and all vertices of I together with their possible neighbors in $V_1(H)$. Add $d - d^*(u) + 1$ new pendant edges to each $u \in J$.

Case 3c. $|C| \geq 5$ and $v_1 \in I, v_2 \in J$.
Remove all edges of C except u_1v_1. Remove all vertices of I together with their possible neighbors in $V_1(H)$. Add $d - d^*(u) + 1$ new pendant edges to each $u \in J \backslash \{u_1, v_1\}$. Add $d - d^*(u)$ new pendant edges to $u \in \{u_1, v_1\}$.

Case 3d. $|C| \geq 5$ and $v_1 \in J, v_2 \in I$.
Act as in Case 3c with the roles of v_1 and v_2 reversed.

Case 3e. $|C| \geq 5$ and $v_1, v_2 \in J$.

Case 3e-1. $I \neq \emptyset$.
Let $y \in I$ be such that there exists a path P from u_1 to y in C that besides y only uses vertices from J. Let x be the neighbor of y on P. If $x \notin \{u_2, v_2\}$ return to Case 3c with vertices x, y in the role of u_1, v_1, respectively. If $x \in \{u_2, v_2\}$ return to Case 3c with vertices x, y in the role of u_2, v_2, respectively.

Case 3e-2. $I = \emptyset$ and C has an edge u_3v_3 independent from u_1v_1 and u_2v_2.
If $d = 5$ add three and if $d = 6$ add four new pendant edges to each $u \in C$. If $d \geq 7$ remove all edges of C except u_1v_1, add $d - d^*(u) + 1$ new pendant edges to each $u \in C \backslash \{u_1, v_1\}$ and $d - d^*(u)$ new pendant edges to $u \in \{u_1, v_1\}$.

Case 3e-3. $I = \emptyset$ and C is a five-cycle.
Write $C = x_1x_2x_3x_4x_5x_1$. Remove all edges from C except x_1x_2. Add $d - d(x_i) + 2$ new pendant edges for $i = 1, 2$ and $d - d(x_i) + 3$ new pendant edges for $i = 3, 4, 5$.

Case 3e-4. $I = \emptyset$ and C is isomorphic to $K_{2,2k}$ for some $k \geq 2$.
Let $A(C) = \{s, t\}$ and $B(C) = \{s_1, \ldots, s_{2k}\}$ be the partition classes of C. Remove all edges ts_i of $E(C)$ for $i = 1, \ldots, 2k$. Add $d - d(t) + 2k + 1$ new pendant edges to t. Add $d - d(s_i) + 2$ new pendant edges to each s_i for $i = 1, \ldots, 2k$. To s add $d - d(s) + 1$ new pendant edges if $d(s) \leq d$ or one new pendant edge if $d(s) \geq d + 1$.

□

Step 5: translate the dominating systems back into 2-factors

Once we have obtained a dominating system $\mathcal{S}$ for the preimage graph H with $cl(G) = L(H)$, it is easy to translate this back into a 2-factor of $cl(G)$ in polynomial time:

- the stars in $\mathcal{S}$ correspond to complete graphs in $cl(G)$ on at least three vertices; a hamiltonian cycle can clearly be constructed in polynomial time;
- the circuits in $\mathcal{S}$ and the edges they dominate correspond to hamiltonian subgraphs in $cl(G)$; one can construct a hamiltonian cycle by traversing the circuit, picking up the edges (vertices in $cl(G)$) one by one and inserting dominated edges at the first instance an end vertex of a dominated edge is encountered. For traversing the circuits we use the polynomial algorithm that finds a eulerian tour in an even connected graph (cf. [4]).

Step 6: translate 2-factors in $cl(G)$ to 2-factors in G

We first introduce some notations. Let $C = v_1 v_2 \ldots v_p v_1$ be a cycle with a fixed orientation. The successor v_{i+1} of v_i is denoted by $v_i^{+C} = v_i^+$ and its predecessor v_{i-1} by $v_i^{-C} = v_i^-$. The segment $v_i v_{i+1} \ldots v_j$ is denoted by $v_i \overrightarrow{C} v_j$, where the subscripts are to be taken modulo $|C|$. The converse segment $v_j v_{j-1} \ldots v_i$ is denoted by $v_j \overleftarrow{C} v_i$. We use similar notations for paths.

We assume we are given a 2-factor F' of $cl(G)$ of a claw-free graph G. Let k be the number of components of F'. Here, we show how to obtain in polynomial time a 2-factor F of G such that F has *at most* k components. We base our translation of the following new theorem, which generalizes a similar result for hamiltonicity [2] in algorithmic sense.

Theorem 11. *Let G be a graph and let $\{u, v, x, y\}$ be a subset of four vertices of $V(G)$ such that $uv \notin E(G)$ and $\{x, y\} \subseteq N(u) \cap N(v)$. Let $N(x) \subseteq N(u) \cup N(v) \cup \{u, v\}$ and let $N(y) \setminus (N(x) \cup \{x\})$ induce a complete graph (or be empty). Then we can find a 2-factor of G with at most k components in polynomial time, if $G + uv$ has a 2-factor with k components.*

Proof. Suppose $G + uv$ has a 2-factor F' with at most k components. Below we give a number of polynomial time transformations of F' such that we obtain a 2-factor F of G with at most k components. If $uv \notin E(F')$ then F' is a 2-factor of G. Suppose $uv \in E(D)$ for some (cycle) component D of F', say $v = u^-$. Let $P = u\overrightarrow{D}v$. We distinguish the following three cases.

First suppose $x \notin V(D)$. Let $x \in V(D')$ for some (cycle) component D' of F. By our assumptions, we may without loss of generality assume that $x^{+D'}u \in E(G)$. Then we replace D and D' by a new cycle $ux^{+D'}\overrightarrow{D'}xv\overleftarrow{P}u$, and we are done.

Second suppose $x \in V(D)$ but $y \notin V(D)$. Let $y \in V(D^*)$ for some (cycle) component D^* of F. Let $y' = y^{+D^*}$ and $y'' = y^{-D^*}$ be the neighbors of y on D^*. Suppose $y'y'' \in E(G)$. Then we replace D^* by $y'\overrightarrow{D^*}y''y'$ and D by $uyv\overleftarrow{P}u$, and we are done. Suppose $y'y'' \notin E(G)$. Since $N(y)\backslash(N(x)\cup\{x\})$ induces a complete graph, we find that one of the edges xy', xy'', say xy', must exist in G. By our assumptions, we then find that $y'u$ or $y'v$ belongs to $E(G)$, and we are done by the same argument as in the previous case.

Third suppose $\{x, y\} \subset V(D)$. Say x is on $u\overrightarrow{P}y$. First suppose $xy \in E(D)$. We replace D by $u\overrightarrow{P}xv\overleftarrow{P}yu$, and we are done. Now suppose $xy \notin E(D)$. Then $x^+ \neq y$. By our assumptions, $x^+ \in N(u) \cup N(v)$. Suppose $ux^+ \in E(G)$. We replace D by $ux^+\overrightarrow{P}vx\overleftarrow{P}u$. Hence we may assume $vx^+ \in E(G)$. Suppose $y^- = x^+$. Then we replace D by $uy\overrightarrow{P}vy^-\overleftarrow{P}u$. Hence we may assume $y^- \neq x^+$. Suppose $y^-x \in E(G)$. Then we replace D by $u\overrightarrow{P}xy^-\overleftarrow{P}x^+v\overleftarrow{P}yu$. Hence we may assume $y^-x \notin E(G)$. Suppose $y^+ = v$. Then we replace D by $u\overrightarrow{P}xvx^+\overrightarrow{P}yu$. Hence we may assume $y^+ \neq v$. Suppose $y^+x \in E(G)$. Then we replace D by $u\overrightarrow{P}xy^+\overrightarrow{P}vx^+\overrightarrow{P}yu$. Hence we may assume $y^+x \notin E(G)$. As $y^-x \notin E(G)$, we then find $y^-y^+ \in E(G)$ due to our assumptions. Then we replace D by $u\overrightarrow{P}y^-y^+\overrightarrow{P}vyu$. This proves Theorem 11. □

Note that in Theorem 11, x and y can be nonadjacent, and G does not have to be claw-free. However, the following observation is easy to see.

Observation 1 ([2]) *If G is claw-free, then the conditions of Theorem 11 are satisfied if x and y are adjacent.*

Then, by the following observation, we can indeed transform a 2-factor of $cl(G)$ that has k components to a 2-factor of G that has at most k components. This means we have proven our main result. For convenience we include the proof of the next observation.

Observation 2 ([2]) *Let x be a vertex of a claw-free graph G with $G[N(x)]$ connected and non-complete. Then the local completion of G at x can be obtained by iteratively joining pairs $\{u, v\} \subseteq N(x)$ that satisfy the conditions in Theorem 11 for some $y \in N(u) \cap N(v)$.*

Proof. Consider the subgraph H_x of G induced by $N(x) \cup \{a \in V(G) \mid ab \in E(G)$ for some $b \in N(x)\}$. Note that x is a vertex of H_x and that H_x is claw-free. Hence, by Observation 1, x and y satisfy the conditions of Theorem 11 (in H_x) for every $y \in N(x)$. Since we only join nonadjacent pairs in $N(x)$, $N(x)$ and $N(y)$ will keep these properties for all $y \in N(x)$. □

Note that the above approach gives a short constructive proof for Theorem 4 (the result of Ryjáček, Saito & Schelp in [19]).

We note that Theorem 9 has the following consequence as a byproduct. We need a few definitions before we can state the result. A *cut vertex* of a graph G is a vertex whose

removal increases the number of components. A *block* of G is a maximal subgraph of G without cut vertices (of itself). Hence if G has no isolated vertices, its blocks are either K_2's or (maximal) 2-connected subgraphs. For the purpose of our next result we call a block B of a claw-free graph G a *semiclique* if B becomes a complete subgraph of $cl(G)$. Since a claw-free graph in which every block is a semiclique has a forest as its preimage, we obtain the following consequence of Theorem 9.

Corollary 1. *Let G be a claw-free graph in which all blocks are semicliques. If G has a 2-factor, then we can construct a minimum 2-factor of G in polynomial time.*

4 Conclusions

In [1] we obtained sharp upper bounds for the minimum number of components of a 2-factor in a claw-free graph. Here we extended these results by presenting a polynomial algorithm that constructs a 2-factor of a claw-free graph with minimum degree at least four whose number of components meets this bound. As a byproduct we showed that the problem of obtaining a minimum 2-factor (if it exists) is polynomially solvable for a subclass of claw-free graphs in which all blocks are semicliques. As another byproduct we gave a short constructive proof for a result of Ryjáček, Saito & Schelp.

Our polynomial time algorithm yields a 2-factor with a number of components below a guaranteed upper bound. This upper bound is completely determined by an upper bound we find for the number of elements of a dominating system of a certain tree (that is obtained from he corresponding triangle-free graph in Theorem 8). As this upper bound is sharp (cf. [20]), our next goal will be to determine the extremal tree cases and try to exclude these from happening. This refined analysis may lead to a better upper bound for claw-free graphs for which the current upper bound is not sharp. Another way to improve our algorithm is trying to refine the algorithm that constructs the tree H^* in Lemma 2 such that we have more information on the number of circuits in the even subgraph X of H.

Finally, Corollary 1 shows that our algorithm yields a 2-factor with a minimum number of components for claw-free graphs with arbitrary minimum degree in which all blocks are semicliques. In future research we aim to generalize this result, i.e, to find a larger class of claw-free graphs for which our (possibly modified) algorithm solves the problem of finding a minimum 2-factor. We will also analyze the class of claw-free graphs with minimum degree 3 that have a 2-factor more carefully.

References

1. H.J. Broersma, D. Paulusma and K. Yoshimoto, *Sharp upper bounds for the minimum number of components of 2-factors in claw-free graphs*, Graphs Combin., to appear.
   see http://www.dur.ac.uk/daniel.paulusma/Papers/Submitted/claw.pdf
2. H.J. Broersma and H. Trommel, *Closure concepts for claw-free graphs*, Discrete Math. **185** (1998) 231–238.
3. S.A. Choudum and M.S. Paulraj, *Regular factors in $K_{1,3}$-free graphs*, J. Graph Theory **15** (1991) 259–265.

4. R. Diestel, *Graph Theory, Second edition*, Graduate Texts in Mathematics **173**, Springer (2000).
5. Y. Egawa and K. Ota, *Regular factors in $K_{1,n}$-free graphs*, J. Graph Theory **15** (1991) 337–344.
6. R.J. Faudree, O. Favaron, E. Flandrin, H. Li, Z. Liu, *On 2-factors in claw-free graphs*, Discrete Math. **206** (1999) 131–137.
7. R. Faudree, E. Flandrin, and Z. Ryjáček *Claw-free graphs—a survey*, Disc. Math. **164** (1997) 87–147.
8. D. Fronček, Z. Ryjáček, and Z. Skupień, *On traceability and 2-factors in claw-free graphs*, Discussiones Mathematicae Graph Theory **24** (2004) 55–71.
9. M.R. Garey and D.S. Johnson, *Computers and Intractability*. W.H. Freeman and Co., New York, 1979.
10. R. Gould and E. Hynds, *A note on cycles in 2-factors of line graphs*, Bull. of ICA. 26 (1999) 46–48.
11. R.J. Gould and M.S. Jacobson, *Two-factors with few cycles in claw-free graphs*, Discrete Math. **231** (2001) 191–197.
12. F. Harary, *Graph Theory*, Addison-Wesley, Reading MA, 1969.
13. B. Jackson and K. Yoshimoto, *Even subgraphs of bridgeless graphs and 2-factors of line graphs*, Discrete Math. **307** (2007) 2775–2785.
14. L. Lovasz and M.D. Plummer, *Matching Theory*, North-Holland Mathematics Studies 121, North-Holland, Amsterdam.
15. M.M. Matthews and D.P. Sumner, *Hamiltonian results in $K_{1,3}$-free graphs*, J. Graph Theory **8** (1984) 139–146.
16. M.D. Plummer, *Graph factors and factorization: 1985-2003: A survey*, Discrete Math. **307** (2007) 791–821.
17. N.D. Roussopoulos, *A $\max\{m, n\}$ algorithm for determining the graph H from its line graph G*, Information Processing Letters **2** (1973) 108-112.
18. Z. Ryjáček, *On a closure concept in claw-free graphs*, J Combin Theory Ser B **70** (1997) 217–224.
19. Z. Ryjáček, A. Saito and R.H. Schelp, *Closure, 2-factor, and cycle coverings in claw-free graphs*, J. Graph Theory **32** (1999) 109–117.
20. K. Yoshimoto, *On the number of components in a 2-factor of a claw-free graph*, Discrete Math. **307** (2007) 2808–2819.