**COMBINATORICA**
Bolyai Society – Springer-Verlag

Combinatorica **25** (4) (2005) 393–405

# ON CONNECTED $[g,f+1]$-FACTORS IN GRAPHS

GUOJUN LI*[†], YING XU[†], CHUANPING CHEN, ZHENHONG LIU

Received October 27, 1998
Revised August 23, 2002

Let $G=(V(G),E(G))$ be a graph with vertex set $V(G)$ and edge set $E(G)$, and $g$ and $f$ two positive integral functions from $V(G)$ to $Z^+-\{1\}$ such that $g(v)\le f(v)\le d_G(v)$ for all $v\in V(G)$, where $d_G(v)$ is the degree of the vertex $v$. It is shown that every graph $G$, including both a $[g,f]$-factor and a hamiltonian path, contains a connected $[g,f+1]$-factor. This result also extends Kano's conjecture concerning the existence of connected $[k,k+1]$-factors in graphs.

## 1. Introduction

The graphs considered in this paper will be finite, undirected and simple. Let $G=(V(G),E(G))$ be a graph with vertex set $V(G)$ and edge set $E(G)$. Without ambiguity, we will often substitute $G$ for $V(G)$ or $E(G)$. Let $g$ and $f$ be two nonnegative integral functions from $V(G)$ to $Z^+-\{1\}$ such that $g(v)\le f(v)\le d_G(v)$ for all $v\in V(G)$. A subgraph $F$ of $G$ is said to be a $[g,f]$-factor of $G$ if $F$ is a spanning subgraph of $G$ such that $g(v)\le d_F(v)\le f(v)$ for each $v\in V(G)$. If $g(v)=f(v)$ for every $v\in V(G)$, we say the $[g,f]$-factor of $G$ to be an $f$-factor. For two constants $a$ and $b$, if $g(v)=a$ and $f(v)=b$ for

*Mathematics Subject Classification (2000):* 05C85

* The work of this author was supported by NSFC of China under Grant No. 10271065, No. 60373025.

† The work of these authors was also supported in part by the US Department of Energy's Genomes to Life program (http://doegenomestolife.org/) under project, "Carbon Sequestration in Synechococcus sp.: From Molecular Machines to Hierarchical Modeling" (www.genomes2life.org) and by National Science Foundation (NSF/DBI-0354771,NSF/ITR-IIS-0407204).

all $v \in V(G)$, then the $[g,f]$-factor is called an $[a,b]$-factor. Further, when $k$ is a nonnegative integer such that $a=b=k$, then the $[a,b]$-factor is called a $k$-factor. The $[g,f]$-factor is called a connected $[g,f]$-factor if it is connected.

The concept of connected factors was first proposed by M. Kano [6]. It is easy to see that the problem of whether a given graph $G$ contains a connected $[g,f]$-factor is $NP$-hard because a connected 2-factor is just a Hamiltonian cycle. It seems to us that the connected factor problem is an interesting research topic since it is closely concerned with hamiltonian problem and information networks.

In order to persuade that our result indeed extends Kano's conjecture [6], we need the following known theorems.

**Theorem 1** ([7]). *Let $G$ be a graph of order $n$. If the degree sum of each pair of nonadjacent vertices of $G$ is at least $n$, then $G$ is hamiltonian.*

**Theorem 2** ([4]). *Let $k$ be a positive integer, and $G$ a graph of order $n$ with $n \geq 4k-5$, $kn$ even, and minimum degree at least $k$. Then $G$ has a $k$-factor if the degree sum of each pair of nonadjacent vertices of $G$ is at least $n$.*

**Theorem 3** ([5]). *If $k \geq 3$ is an integer and $G$ is a connected graph of order $n$ such that $n \geq 4k-3$, $kn$ is even, and $G$ has minimum degree at least $k$ and $\max\{d_G(u), d_G(v)\} \geq n/2$ for each pair of nonadjacent vertices $u,v$ in $G$, then $G$ has a $k$-factor.*

**Theorem 4** ([2]). *If $k \geq 2$ and $G$ is a graph of order $n$ with minimum degree at least $k$, and with $n \geq 8k-16$ for even $n$, and $n \geq 6k-13$ for odd $n$, such that the degree sum of each nonadjacent pair of vertices of $G$ is at least $n$, then for any Hamiltonian cycle $C$ of $G$, $G$ has a $[k,k+1]-$factor containing $C$.*

In 1993, M. Kano [6] presented the following two problems.

**Problem 1** ([6]). Find sufficient conditions for a graph to have a connected $[a,b]$-factor from known results on hamiltonian cycles and paths.

**Problem 2** ([6]). Find sufficient conditions for a graph to have a connected $[k,k+1]$-factor.

Furthermore, he made the following conjecture.

**Conjecture 1.** Let $k$ be a positive integer and $G$ a graph of order $n$ with $n \geq 4k-5$, $kn$ even, and minimum degree at least $k$. If the degree sum of each pair of nonadjacent vertices of $G$ is at least $n$, then $G$ has a connected $[k,k+1]$-factor.

Corresponding to the above conjecture and problems, M. Cai [1] has shown the following theorem.

**Theorem 5.** *Let $k$ be a positive integer and $G$ a connected graph of order $n$. If $G$ has a $k$-factor $F$ and, moreover, among any three independent vertices of $G$ there are (at least) two with degree sum at least $n-k$, then $G$ has a matching $M$ such that $M$ and $F$ are edge disjoint and $M \cup F$ is a connected $[k, k+1]$-factor of $G$.*

It is easy to see that Theorem 5 implies that Conjecture 1 is true. Our main aim in this paper is to present a more succinct sufficient condition for a graph to have a connected $[g, f+1]$-factor. From Theorem 1 and Theorem 2, the graph $G$ satisfying the conditions of Conjecture 1 contains both a hamiltonian path and a $k$-factor. Our main result in this paper is the following theorem.

**Theorem 6.** *Let $G$ be a graph, and $g$ and $f$ two positive integral functions from $V(G)$ to $Z^+ - \{1\}$ such that $g(v) \leq f(v) \leq d_G(v)$ for all $v \in V(G)$. If $G$ has both a $[g, f]$-factor and a hamiltonian path, then $G$ contains a connected $[g, f+1]$-factor.*

Theorems 1 and 2 imply that our theorem indeed extends Conjecture 1. We will prove the conjecture by an algorithmic approach. The structure of this paper will be arranged as follows. In section 2, we will design an algorithm to find a connected $[g, f+1]$-factor. In section 3, we are going to prove the correctness of the algorithm.

## 2. Design of Algorithm

The graph under consideration in this section contains a $[g, f]$-factor $F$ and a hamiltonian path $P$. Intuitively, it seems to possibly find a connected $[g, f+1]$-factor which consists of a $[g, f]$-factor $F$ and some edges of a hamiltonian path $P$, whereas it is impossible to do so. Let us see the following example.

**Example 1.** The graph depicted in Fig. 1 contains a 2-factor $F$ and a hamiltonian path $P = a_1a_2a_3xb_1b_2b_3yc_1c_2c_3zd_1d_2d_3$. The 2-factor $F$ consists of five triangles which are drawn with thick lines. It is easy to check that this graph does not contain a connected $[2, 3]$-factor which consists of $F$ and some edges of $P$. Nevertheless, it contains a connected $[2, 3]$-factor $(F - \{yz\}) \cup \{xa_3, yb_3, zc_3, zd_1\}$.

Notice that the deletion of the edge $yz$ from $F$ does not disconnect the component, i.e., the triangle $xyz$, of $F$. Such an edge will play an important role in the existence of a connected $[g, f+1]$-factor.

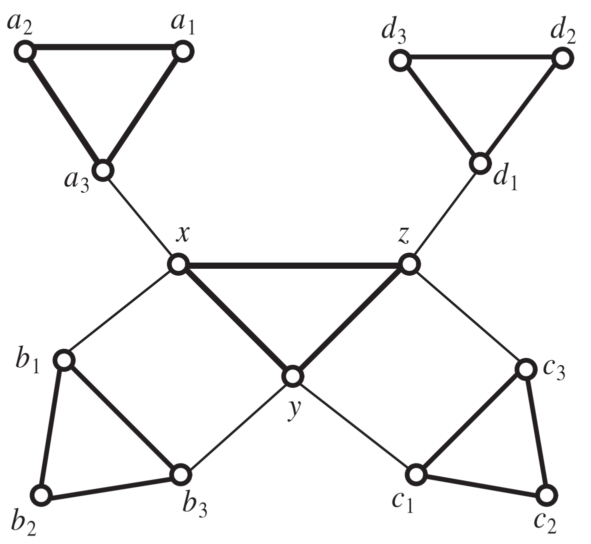

**Fig. 1.** A graph containing a 2-factor and a hamiltonian path

In general, the required connected $[g, f+1]$-factor will be found concretely by an algorithm procedure. At any step of the algorithm, we have a current spanning subgraph and a current edge of $P$ at hand, then we have to decide whether a current edge of $P$ should be added to it and whether a specified edge of $F$ should be discarded.

We are now going to design an algorithm to find a connected $[g, f+1]$-factor. To do so, we first define some special symbols and terminologies. Let $F$ be a $[g, f]$-factor of $G$ and $P_n = v_1v_2 \cdots v_n$ a hamiltonian path of $G$. We use $P_k = v_1v_2 \cdots v_k$ to denote a subpath of $P_n$. For a spanning subgraph $B$ of $G$ and a vertex $x$ of $G$, by $B_x$ we denote the component of $B$ containing $x$. A vertex $v_i$ is said to be *singular* if both its predecessor $v_{i-1}$ (or $v_i^-$) and its successor $v_{i+1}$ (or $v_i^+$) along the path $P_n$ are not in $F_{v_i}$. If every vertex of a component of $F$ is *singular*, we call the component *singular*. For a *singular component* $C$ of $F$ and an edge $xy$ of $C$, if $C - \{xy\}$ is still connected then the edge $xy$ is said to be a *friendly edge*, and the vertices $x$ and $y$ are called *friendly mutually*. As seen in Example 1, the friendly edges will play an important role in the existence of a connected $[g, f+1]$-factor. It is obvious that if $g \geq 2$ then every *singular component* $C$ of $F$ has a *friendly edge*.

We may assume that the $[g, f]$-factor $F$ of $G$ is not connected or otherwise we have nothing to do. To begin with, we take one and only one *friendly edge* in each *singular component* of $F$, and color the edge and its endpoints with red color. Simultaneously, we color all the *nonsingular vertices* blue. If $xy$ is a red edge, that is, $x$ and $y$ are *friendly mutually*, then we use $\bar{x}$

(resp. $\bar{y}$) to denote the vertex $y$ (resp. $x$). Note that every singular vertex is either colorless or red, and all the nonsingular vertices are blue.

It is obvious that if every vertex of $G$ is blue (nonsingular) then $G$ has a connected $[g, f+1]$-factor. In fact, let $P' \subseteq E(P)$ be a set of edges connecting two distinct components of $F$, and then $F \cup P'$ is exactly a connected $[g, f+1]$-factor of $G$.

Spontaneously, readers may want to know the reason why we color those vertices. As seen at the above paragraph, it is easy to deal with all the blue vertices. For a red vertex, since the friendly edge incident with it can be discarded, both the two edges of $P$ incident with it can be picked up without overloading the red vertex with degree. Of course, if we discard a friendly edge $xy$, we should ensure that at least one edge on $P$ incident with $x$ (resp. $y$) must be picked up when $g(x) = f(x)$ (resp. $g(y) = f(y)$) so as to avoid lack of degree. We will see later that the hardness of finding a desired connected factor lies in those singular vertices which are colorless. In the process of our algorithm, some other vertices will also be colored. We will see that those colored vertices will play an important role in both the description and the proof of the algorithm.

Let $M$ denote the set of all red (friendly) edges. The algorithm designed below will be proceeded along the hamiltonian path edge by edge. At the beginning of our algorithm, we have a current subgraph $B^1 = F$ and the current vertex $v_1$. We decide whether or not the edge $v_1v_2$ is added to the current subgraph $B^1$ according to whether or not $v_2$ is not in the component $B^1_{v_1}$ of $B^1$. Then we get a subgraph $B^2$ such that $B^2 = B^1 + \{v_1v_2\}$ if $v_2$ is not in $B^1_{v_1}$ and $B^2 = B^1$ otherwise. In the general step, we have a current subgraph $B^k$ and current vertex $v_k$. Then we need some quite novel technique to decide whether or not the edge $v_kv_{k+1}$ should be added to the current subgraph $B^k$. The process of deciding if the edge $v_kv_{k+1}$ is added to $B^k$ constitutes an outloop of our algorithm. In the course of execution of our algorithm, some edges of $P_n$ will be added to the current subgraph and some edges in $M$ will be deleted from the current graph so that the final subgraph obtained by the algorithm is indeed a connected $[g, f+1]$-factor of $G$.

**Algorithm: ConnFactor.** *Initially, input: $B^1 = F$, $P_n = v_1v_2 \ldots v_n$, and $M$=the set of all red edges. We set $B^2 = B^1$ if $v_2 \in F_{v_1}$, and $B^2 = B^1 + \{v_1v_2\}$ otherwise. Set $k = 2$, go to step 1.*

1. *If $k = n$, output $B^n = B^n - M$; otherwise, go to step 2.*

**Comment.** In next section it will be proved that the $B^n$ output by this algorithm is a connected $[g, f+1]$-factor.

*2. If $v_k$ has been colored, go to step 3; otherwise, color the vertex $v_k$ blue iff the previous edge $v_{k-1}v_k$ is not added to $B^k$, go to step 3.*

**Comment.** If the vertex $v_k$ has been colored, then the color (red or blue) must be gotten before starting the algorithm. Otherwise, $v_k$ must be singular since all the nonsingular vertices were colored blue before starting the algorithm. If the previous edge $v_{k-1}v_k$ is not added to $B^k$, the next edge $v_kv_{k+1}$ never overloads the current vertex $v_k$ with degree. This is why we color it blue when $v_{k-1}v_k$ is not added to $B^k$. Therefore, the next edge $v_kv_{k+1}$ never overloads the current vertex $v_k$ as long as the current vertex $v_k$ is blue. Note that if $v_k$ is red, we need to decide whether the friendly edge $v_k\bar{v}_k$ remains in the current subgraph $B^k$ or not. Let $\bar{v}_k = v_{\bar{k}}$ where $\bar{k}$ stands for the subscript of the friendly vertex of $v_k$. When $\bar{k} < k$ and at least one of $v_{\bar{k}}v_{\bar{k}}^-$ and $v_{\bar{k}}v_{\bar{k}}^+$ is added to $B^k$, the next edge $v_kv_{k+1}$ never overloads the current vertex $v_k$ since we can delete the friendly edge $v_kv_{\bar{k}}$.

*3. If $v_{k+1} \notin B^k_{v_k}$, go to step 3.1; otherwise, go to step 3.2.*

**Comment.** $v_{k+1} \notin B^k_{v_k}$ implies that $v_k$ and $v_{k+1}$ lie in different components of the current subgraph $B^k$.

*3.1. If $v_k$ is colored, we set $B^{k+1} = B^k + \{v_kv_{k+1}\}$, $k = k+1$, and return to step 1; otherwise, we set $B^{k+1} = B^k$, $k = k+1$, and return to step 1.*

**Comment.** If $v_k$ is colored, the next edge $v_kv_{k+1}$ can always be added to $B^k$ without overloading the current vertex $v_k$ (note that if $v_k$ is red, the algorithm ensures that at least one of $v_{\bar{k}}v_{\bar{k}}^-$ and $v_{\bar{k}}v_{\bar{k}}^+$ was added to $B^k$ when $\bar{k} < k$ or will be added to $B^k$ when $\bar{k} > k$). Otherwise, step 2 implies that the previous edge $v_{k-1}v_k$ must have been added to $B^k$. Since $v_k$ may have not a friendly vertex, the next edge $v_kv_{k+1}$ should be given up in order to avoid overloading the current vertex $v_k$. However, we do not worry about that it will disconnect $v_k$ and $v_{k+1}$ (see the proof of the correctness).

*3.2. If $v_k$ is colorless, we set $B^{k+1} = B^k$ and $k = k+1$, return to step 1; otherwise, go to step 3.2.1.*

**Comment.** $v_k$ being colorless implies that it is singular and thus $v_k$ and $v_{k+1}$ are in different components of $F$. However, they are in the same component $B^k_{v_k}$ of $B^k$ since the algorithm executes step 3.2 only when $v_{k+1} \in B^k_{v_k}$. Therefore, it is natural to abandon the next edge $v_kv_{k+1}$. When $v_k$ is colored, the situation is more complicated.

*3.2.1. If $v_{k+1}$ is red, we set $B^{k+1} = B^k + \{v_kv_{k+1}\}$ and $k = k+1$, return to step 1; otherwise, go to step 3.2.2.*

**Comment.** $v_{k+1}$ being red implies that it is a friendly vertex. Although $v_k$ and $v_{k+1}$ are in the same component $B^k_{v_k}$ of the current subgraph $B^k$, we pick up the edge $v_kv_{k+1}$ in order to guarantee that, for every friendly vertex, at least one of the two edges incident with it on $P_n$ will be picked up. If $v_{k+1}$ is not red, it is either blue or colorless, then go to step 3.2.2 to consider the current vertex $v_k$. Note that $v_k$ is either blue or red at this time.

*3.2.2. If $v_k$ is blue, we set $B^{k+1}=B^k$ and $k=k+1$, return to step 1; otherwise, $v_k$ must be red, go to step 3.2.3.*

**Comment.** Note that at this time both $v_k$ and $v_{k+1}$ are in the same component of the current subgraph and $v_{k+1}$ is not red. If $v_k$ is blue, we set $B^{k+1}=B^k$ so as to ensure that $v_{k+1}$ will be blue after step 2 of next outloop. Otherwise $v_k$ must be red, now we should consider its friendly vertex $\bar{v}_k$.

*3.2.3. If $\bar{v}_k \notin P_k$, we set $B^{k+1}=B^k$ when $v_{k-1}v_k \in B^k$, and $B^{k+1}=B^k+\{v_kv_{k+1}\}$ when $v_{k-1}v_k \notin B^k$, then we set $k=k+1$, return to step 1. Otherwise, i.e., $\bar{v}_k \in P_k$. If $\bar{v}_k^-\bar{v}_k, \bar{v}_k\bar{v}_k^+ \in B^k$, we set $B^{k+1}=B^k$ when $v_{k-1}v_k \in B^k$, and $B^{k+1}=B^k+\{v_kv_{k+1}\}$ when $v_{k-1}v_k \notin B^k$, then we set $k=k+1$, return to step 1; or else we set $M=M-\{v_k\bar{v}_k\}$, $B^{k+1}=B^k$ and $k=k+1$, return to step 1.*

**Comment.** If $\bar{v}_k \notin P_k$, i.e., $\bar{k}>k$ where $\bar{k}$ is the subscript of the friendly vertex of $v_k$, then it must be that exactly one of $v_{k-1}v_k$ and $v_kv_{k+1}$ is added to the current subgraph $B^k$. Otherwise, two cases must be considered: In the first case that $\bar{v}_k^-\bar{v}_k, \bar{v}_k\bar{v}_k^+ \in B^k$, it is ensured that exactly one of $v_{k-1}v_k$ and $v_kv_{k+1}$ should be added to the current subgraph $B^k$, and the friendly edge $v_k\bar{v}_k$ is going to be deleted from the current subgraph (i.e., $v_k\bar{v}_k$ is kept in $M$). In the opposite case, the next edge $v_kv_{k+1}$ must be abandoned, and the friendly edge $v_k\bar{v}_k$ will be kept in the current subgraph (i.e., $v_k\bar{v}_k$ must be deleted from $M$).

*4. Stop the algorithm.*

**Example 2.** Algorithm *ConnFactor* is applied to the graph depicted in Fig. 1. At the beginning of the algorithm, all vertices apart from $x$, $y$ and $z$ are colored blue. The triangle $xyz$ is uniquely singular component of $F$. We can choose arbitrary one of three edges $xy$, $yz$ and $xz$, say $xy$, as a friendly edge of the component. Then both $x$ and $y$ are colored red, and $z$ is colorless. It is easy to check that the algorithm *ConnFactor* outputs a connected [2,3]-factor $\{F-\{xy\}\}\cup\{xa_3, xb_1, yb_3, yc_1, zd_1\}$.

Similarly, if we choose $yz$ (resp. $xz$) as a friendly edge of the component $xyz$, then the algorithm will output a connected [2,3]-factor $\{F-\{yz\}\}\cup\{xa_3, yb_3, yc_1, zc_3, zd_1\}$ (resp. $\{F-\{xz\}\}\cup\{xa_3, xb_1, yc_1, zc_3, zd_1\}$).

In the next section, we will prove that the $B^n$ output by our algorithm is indeed a connected $[g,f+1]$-factor.

## 3. Proof of Correctness

In this section, we shall concentrate our attentions on the proof of the correctness of our algorithm. From the design of our algorithm, it is easy to see that $B^n$ is a $[g,f+1]$-factor. For preciseness, we show this fact in Theorem 7. However, the connectivity of $B^n$ is quite tricky. We will attack the tricky in Theorem 8.

**Theorem 7.** *$B^n$ is a $[g,f+1]$-factor of $G$.*

We are now going to show the connectivity of the final subgraph $B^n$ output by the algorithm. For convenience, we define the following notations.

**Definition.** Let $k$ be a positive integer less than or equal to $n$, $B^k$ a current subgraph at the time point when $v_k$ is a current vertex, and $D$ a component of $B^k$. If at least one of the following statements holds, then $D$ is said to be a *kth-good component* of $B^k$; otherwise, a *kth-bad component* of $B^k$:

1. At least one blue vertex in $D$ is not in $P_k$;
2. At least one red vertex, say $v_i$, in $D$ is not in $P_k$ (i.e., $i>k$) such that at least one of $v_{\bar{i}-1}v_{\bar{i}}$ and $v_{\bar{i}}v_{\bar{i}+1}$ is not yet added to $B^k$.

Furthermore, if either $B^k$ is connected or every component of $B^k - B^k_{v_k}$ is *kth-good*, then $B^k$ is called *kth-good*; otherwise, $B^k$ is called *kth-bad*.

In order to verify the connectivity of $B^n$, we only need to show the following theorem.

**Theorem 8.** *$B^k$ is kth-good for any positive integer $2\le k\le n$.*

**Proof.** Suppose, to the contrary, that we can choose a smallest positive integer $k\le n$ such that $B^k$ is $kth$-bad, that is, $B^k$ is not connected and at least one component of $B^k - B^k_{v_k}$ is $kth$-bad. By the choice of $k$, we get that.

- $v_{k-1}v_k \notin B^k$ and $v_k \notin B^{k-1}_{v_{k-1}}$.

Assume to the contrary that the above property is not true. Then $B^k_{v_k} = B^k_{v_{k-1}}$, i.e., both $v_{k-1}$ and $v_k$ are in the same component of $B^k$. Therefore, the $kth$-bad component of $B^k - B^k_{v_k}$ is also a $(k-1)th$-bad component of $B^{k-1} - B^{k-1}_{v_{k-1}}$ (note that $B^k_{v_k} = B^{k-1}_{v_{k-1}}$ if $v_k \in B^{k-1}_{v_{k-1}}$ and $B^{k-1}_{v_{k-1}} \subseteq B^k_{v_k}$ if $v_{k-1}v_k \in B^k$). This implies that $B^{k-1}$ is $(k-1)th$-bad, violating the choice of $k$. ∎

Now that $B^{k-1}$ is $(k-1)th$-good, $v_{k-1}v_k \notin B^k$ and $v_k \notin B^{k-1}_{v_{k-1}}$, the $kth$-bad component of $B^k - B^k_{v_k}$ must be $B^k_{v_{k-1}}$, and thus we have that

- $B^{k-1}_{v_{k-1}}$ is a $(k-1)th$-bad component of $B^{k-1}$.

In fact, that $v_{k-1}v_k \notin B^k$ ensures that $B^{k-1}_{v_{k-1}} = B^k_{v_{k-1}}$. ▮

Recalling step 3.1, we observe the following fact.

- $v_{k-1}$ is colorless (singular).

Assume to the contrary that $v_{k-1}$ is colored. Since $v_k \notin B^{k-1}_{v_{k-1}}$, the algorithm executes step 3.1 to pick up the edge $v_{k-1}v_k$, a contradiction. ▮

Since $v_{k-1}$ was not colored in step 2, it follows that

**Pr.** $1^{(1)}$**.** $v_{k-2}v_{k-1} \in B^{k-1}$.

Since otherwise, at the time point when $v_{k-1}$ is current vertex, $v_{k-1}$ should had been colored blue by step 2. ▮

Recalling that $B^{k-1}_{v_{k-1}}$ is a $(k-1)th$-bad component of $B^{k-1}$ and that $v_{k-1}$ is colorless, noting more that $B^{k-2}_{v_{k-1}}$ is a subgraph of $B^{k-1}_{v_{k-1}}$, it follows that $B^{k-2}_{v_{k-1}}$ must be a $(k-2)th$-bad component of $B^{k-2}$. By the choice of $k$ we have that

**Pr.** $2^{(1)}$**.** $v_{k-1} \in B^{k-2}_{v_{k-2}}$.

Since otherwise $B^{k-2}_{v_{k-1}}$ is a $(k-2)th$-bad component of $B^{k-2} - B^{k-2}_{v_{k-2}}$ because $B^{k-1}_{v_{k-1}}$ is $(k-1)th$-bad and $v_{k-1}$ is colorless, violating the choice of $k$. ▮

Now we consider the time point at which $v_{k-2}$ is a current vertex. By Pr. $2^{(1)}$ the algorithm had to execute step 3.2. Then Pr. $1^{(1)}$ ensures that $v_{k-2}$ must be colored since otherwise the edge $v_{k-2}v_{k-1}$ should had been abandoned. Now that $v_{k-2}$ is colored, the algorithm goes to step 3.2.1 and then to step 3.2.2 to check whether $v_{k-2}$ is blue or red as $v_{k-1}$ is colorless. Step 3.2.2 together with Pr. $1^{(1)}$ guarantees that

**Pr.** $3^{(1)}$**.** $v_{k-2}$ must be red.

Since otherwise $v_{k-2}$ must be blue, the algorithm would have executed step 3.2.2 to abandon the edge $v_{k-2}v_{k-1}$, contradicting Pr. $1^{(1)}$. ▮

Now that $v_{k-2}$ is red, the algorithm had to go to step 3.2.3. Then it follows that

**Pr.** $4^{(1)}$**.** $\bar{v}_{k-2} \in P_{k-2}$.

Assume to the contrary that $\bar{v}_{k-2} \notin P_{k-2}$. Then step 3.2.3 and Pr. $1^{(1)}$ ensure that $v_{k-3}v_{k-2} \notin B^{k-2}$. Note that $\bar{v}_{k-2}$, $v_{k-2}$ and $v_{k-1}$ are all in the same component of $B^{k-1}$. Therefore, there is a red vertex $\bar{v}_{k-2}$ in the component $B^{k-1}_{v_{k-1}}$ of $B^{k-1}$, but not in $P_{k-1}$ such that $v_{k-3}v_{k-2} \notin B^{k-1}$. It follows from the definition that $B^{k-1}_{v_{k-1}}$ is a $(k-1)th$-good component of $B^{k-1}$, a contradiction. ∎

Recall that $v_{k-1} \in B^{k-2}_{v_{k-2}}$, $v_{k-2}$ is red, and $v_{k-1}$ is colorless, then at the time point when $v_{k-2}$ was current vertex, the algorithm had to execute step 3.2.3. Now that $\bar{v}_{k-2} \in P_{k-2}$, and step 3.2.3 picked up the edge $v_{k-2}v_{k-1}$, it follows that

**Pr.** $5^{(1)}$**.** $\bar{v}^-_{k-2}\bar{v}_{k-2}, \bar{v}_{k-2}\bar{v}^+_{k-2} \in B^{k-2}$, and $v_{k-3}v_{k-2} \notin B^{k-2}$.

Since $v_{k-1}$ is colorless, $\bar{v}^-_{k-2}\bar{v}_{k-2}, \bar{v}_{k-2}\bar{v}^+_{k-2} \in B^{k-3}$, and $B^{k-1}_{v_{k-1}}$ is $(k-1)th$-bad (i.e., there is no vertex in $B^{k-1}_{v_{k-1}}$ satisfying the conditions 1 or 2 of the definition), there is still no vertex in $B^{k-3}_{v_{k-1}}$ satisfying the conditions 1 or 2 of the definition. Thus it follows that

**Pr.** $6^{(1)}$**.** $B^{k-3}_{v_{k-1}}$ is still a $(k-3)th$-bad component of $B^{k-3}$.

By the choice of $k$ and Pr. $6^{(1)}$, we have that

**Pr.** $7^{(1)}$**.** $v_{k-1}, v_{k-2} \in B^{k-3}_{v_{k-3}}$.

Since otherwise $B^{k-3}_{v_{k-1}} (= B^{k-3}_{v_{k-2}})$ would had been a $(k-3)th$-bad component of $B^{k-3} - B^{k-3}_{v_{k-3}}$, contradicting the choice of $k$. ∎

Then it follows that

**Pr.** $8^{(1)}$**.** $B^{k-3}_{v_{k-1}} = B^{k-3}_{v_{k-2}} = B^{k-3}_{v_{k-3}}$ is a $(k-3)th$-bad component of $B^{k-3}$.

**Pr.** $9^{(1)}$**.** $v_{k-3}$ is colorless.

Assume to the contrary that $v_{k-3}$ is colored. Then at the time point when $v_{k-3}$ is the current vertex, the algorithm had to execute step 3.2 since $v_{k-2} \in B^{k-3}_{v_{k-3}}$, and then went to step 3.2.1 to pick up the edge $v_{k-3}v_{k-2}$, violating Pr. $5^{(1)}$. ∎

Now that $v_{k-3}$ has the same property as $v_{k-1}$, we can repeatedly use the same methods as used above to consider the vertices $v_{k-4}$ and $v_{k-5}$, and so on. In general, assume that we have gotten the following properties by repeating the above methods $l$ times.

**Pr.** $1^{(l)}$**.** $v_{k-2j}v_{k-2j+1} \in B^{k-2j+1}, \quad j = 1, 2, \ldots, l$.

**Pr.** $2^{(l)}$**.** $v_{k-2j+1} \in B^{k-2l}_{v_{k-2l}}, \quad j = 1, 2, \ldots, l$.

**Pr.** $3^{(l)}$**.** $v_{k-2j}, \quad j = 1, 2, \ldots, l$, must be red.

**Pr.** $4^{(l)}$**.** $\bar{v}_{k-2j} \in P_{k-2j}, \quad j = 1, 2, \ldots, l$.

**Pr.** $5^{(l)}$**.** $\bar{v}^-_{k-2j}\bar{v}_{k-2j}, \bar{v}_{k-2j}\bar{v}^+_{k-2j} \in B^{k-2l}$, and $v_{k-2j-1}v_{k-2j} \notin B^{k-2j}, \quad j = 1, 2, \ldots, l$.

**Pr.** $6^{(l)}$**.** $B^{k-2l-1}_{v_{k-1}}$ is a $(k-2l-1)th$-bad component of $B^{k-2l-1}$.

**Pr.** $7^{(l)}$**.** $v_{k-j}\in B^{k-2l-1}_{v_{k-2l-1}}, \quad j=1,2,\dots,2l.$

**Pr.** $8^{(l)}$**.** $B^{k-2l-1}_{v_{k-1}}=B^{k-2l-1}_{v_{k-2}}=\cdots=B^{k-2l-1}_{v_{k-2l-1}}$ is a $(k-2l-1)th$-bad component of $B^{k-3}$.

**Pr.** $9^{(l)}$**.** $v_{k-2j-1}, \quad j=1,2,\dots,2l$, is colorless.

Repeating once more the above methods to check the vertices $v_{k-2l-2}$ and $v_{k-2l-3}$, we can show the following properties.

**Pr.** $1^{(l+1)}$**.** $v_{k-2l-2}v_{k-2l-1}\in B^{k-2l-1}$.

Since otherwise, at the time point when $v_{k-2l-1}$ is current vertex, $v_{k-2l-1}$ should have been colored blue by step 2 contradicting Pr. $9^{(l)}$. ∎

Recalling that $B^{k-2l-1}_{v_{k-2l-1}}$ is a $(k-2l-1)th$-bad component of $B^{k-2l-1}$ and that $v_{k-2l-1}$ is colorless, noting more that $B^{k-2l-2}_{v_{k-2l-1}}$ is a subgraph of $B^{k-2l-1}_{v_{k-2l-1}}$, it follows that $B^{k-2l-2}_{v_{k-2l-1}}$ must be a $(k-2l-2)th$-bad component of $B^{k-2l-2}$. By the choice of $k$ we have that

**Pr.** $2^{(l+1)}$**.** $v_{k-2l-1}\in B^{k-2l-2}_{v_{k-2l-2}}$.

Since otherwise $B^{k-2l-2}_{v_{k-2l-1}}$ is a $(k-2l-2)th$-bad component of $B^{k-2l-2}-B^{k-2l-2}_{v_{k-2l-2}}$ because $B^{k-2l-1}_{v_{k-2l-1}}$ is $(k-2l-1)th$-bad and $v_{k-2l-1}$ is colorless, violating the choice of $k$. ∎

Now we consider the time point at which $v_{k-2l-2}$ is a current vertex. By Pr. $2^{(l+1)}$ the algorithm had to execute step 3.2. Then Pr. $1^{(l+1)}$ ensures that $v_{k-2l-2}$ must be colored since otherwise the edge $v_{k-2l-2}v_{k-2l-1}$ should had been abandoned. Now that $v_{k-2l-2}$ is colored, the algorithm goes to step 3.2.1 and then to step 3.2.2 to check whether $v_{k-2l-2}$ is blue or red as $v_{k-2l-1}$ is colorless. Step 3.2.2 together with Pr. $1^{(l+1)}$ guarantees that

**Pr.** $3^{(l+1)}$**.** $v_{k-2l-2}$ must be red.

Since otherwise $v_{k-2l-2}$ must be blue, the algorithm had executed step 3.2.2 to abandon the edge $v_{k-2l-2}v_{k-2l-1}$, contradicting Pr. $1^{(l+1)}$. ∎

Now that $v_{k-2l-2}$ is red, the algorithm had to go to step 3.2.3. Then it follows that

**Pr.** $4^{(l+1)}$**.** $\bar{v}_{k-2l-2}\in P_{k-2l-2}$.

Assume to the contrary that $\bar{v}_{k-2l-2}\notin P_{k-2l-2}$. Then step 3.2.3 and Pr. $1^{(l+1)}$ ensure that $v_{k-2l-3}v_{k-2l-2}\notin B^{k-2l-2}$. Note that $\bar{v}_{k-2l-2}$, $v_{k-2l-2}$ and $v_{k-2l-1}$ are all in the same component of $B^{k-2l-1}$. Therefore, there is a red vertex $\bar{v}_{k-2l-2}$ in the component $B^{k-2l-1}_{v_{k-2l-1}}$ of $B^{k-2l-1}$, but not in $P_{k-2l-1}$ such that $v_{k-2l-3}v_{k-2l-2}\notin B^{k-2l-1}$. It follows from the definition that $B^{k-2l-1}_{v_{k-2l-1}}$ is a $(k-2l-1)th$-good component of $B^{k-2l-1}$, a contradiction. ∎

Recall that $v_{k-2l-1} \in B^{k-2l-2}_{v_{k-2l-2}}$, $v_{k-2l-2}$ is red, and $v_{k-2l-1}$ is colorless, then at the time point when $v_{k-2l-2}$ was current vertex, the algorithm had to execute step 3.2.3. Now that $\bar{v}_{k-2l-2} \in P_{k-2l-2}$, and step 3.2.3 picked up the edge $v_{k-2}v_{k-1}$, it follows that

**Pr.** $5^{(l+1)}$**.** $\bar{v}^{-}_{k-2l-2}\bar{v}_{k-2l-2}, \bar{v}_{k-2l-2}\bar{v}^{+}_{k-2l-2} \in B^{k-2l-2}$, and $v_{k-2l-3}v_{k-2l-2} \notin B^{k-2l-2}$.

Since $v_{k-2l-1}$ is colorless, $\bar{v}^{-}_{k-2l-2}\bar{v}_{k-2l-2}, \bar{v}_{k-2l-2}\bar{v}^{+}_{k-2l-2} \in B^{k-2l-3}$, and $B^{k-2l-1}_{v_{k-2l-1}}$ is $(k-2l-1)th$-bad (i.e., there is no vertex in $B^{k-2l-1}_{v_{k-2l-1}}$ satisfying the conditions 1 or 2 of the definition), there is still no vertex in $B^{k-2l-3}_{v_{k-2l-1}}$ satisfying the conditions 1 or 2 of the definition. Thus it follows that

**Pr.** $6^{(l+1)}$**.** $B^{k-2l-3}_{v_{k-2l-1}}$ is still a $(k-2l-3)th$-bad component of $B^{k-2l-3}$.

By the choice of $k$ and Pr. $6^{(l+1)}$, we have that

**Pr.** $7^{(l+1)}$. $v_{k-2l-1}, v_{k-2l-2} \in B^{k-2l-3}_{v_{k-2l-3}}$.

Since otherwise $B^{k-2l-3}_{v_{k-2l-1}} (= B^{k-2l-3}_{v_{k-2l-2}})$ would had been a $(k-2l-3)th$-bad component of $B^{k-2l-3} - B^{k-2l-3}_{v_{k-2l-3}}$, contradicting the choice of $k$. ∎

Then it follows that

**Pr.** $8^{(l+1)}$. $B^{k-2l-3}_{v_{k-2l-1}} = B^{k-2l-3}_{v_{k-2l-2}} = B^{k-2l-3}_{v_{k-2l-3}}$ is a $(k-2l-3)th$-bad component of $B^{k-2l-3}$.

**Pr.** $9^{(l+1)}$. $v_{k-2l-3}$ is colorless.

Assume to the contrary that $v_{k-2l-3}$ is colored. Then at the time point when $v_{k-2l-3}$ is the current vertex, the algorithm had to execute step 3.2 since $v_{k-2l-2} \in B^{k-2l-3}_{v_{k-2l-3}}$, and then went to step 3.2.1 to pick up the edge $v_{k-2l-3}v_{k-2l-2}$, violating Pr. $5^{(l+1)}$. ∎

Repeating the above methods infinitely, we get a sequence of vertices, $\{v_{k-j} : j = 0, 1, 2, \ldots, \}$, such that $\bar{v}_{k-2j} \in P_k, j = 1, 2, \ldots$, and $\{v_{k-2j} : j = 1, 2, \ldots, \} \cap \{v_{k-2j} : j = 1, 2, \ldots, \} = \emptyset$, contradicting the finiteness of $k$. The proof of Theorem 8 is completed. ∎

In Theorem 8, let $k = n$. Since $B^n$ is $nth$-good component, $B^n$ must be connected (otherwise, every component of $B^n - B^n_{v_n}$ is a $nth$-good component, by the definition of $nth$-good component, there is at least one vertex, which is red or blue, not in $P_n$, contradicting that $P_n$ is a hamiltonian path). Therefore, Theorem 7 and Theorem 8 ensure that $B^n$ is indeed a connected $[g, f+1]$-factor. Theorem 6 is proved.

**Corollary 1.** *If a connected graph contains a $k$-factor with $k \geq 1$ and a hamilton path then $G$ has a connected $[k, k+1]$-factor.*

**Proof.** If $k = 1$ the proof is trivial; otherwise the Corollary 1 is right a special case of Theorem 6. ∎

## 4. Acknowledgement

We would like to appreciate the anonymous referees for their useful suggestions to improve and polish this paper. We also thank Professor Y. J. Zhu for his checking the original paper.

## References

[1] M. C. Cai: A degree condition for the existence of connected $[k,k+1]$-factors, *Systems Science and Mathematical Sciences* **8(4)** (1995), 364–368.

[2] M. C. Cai, Y. J. Li and M. Kano: A $[k,k+1]$-factor containing given Hamiltonian cycle, *Sci. China Ser. A* **41(9)** (1998), 933–938.

[3] J. A. Bondy and U. S. R. Murty: *Graph theory with applications*, Macmillan, London and Elsevier, New York (1976).

[4] T. Iida and T. Nishimura: An Ore-type conditions for the existence of $k$-factors in graphs, *Graphs and Combinatorics* **7** (1991), 353–361.

[5] T. Nishimura: A degree condition for the existence of $k$-factors, *J. Graph Theory* **16(2)** (1992), 141–151.

[6] M. Kano: Some current results and problems on factors of graphs, in: *Proc. 3rd China–USA Internat. Conf. on Graph Theory and Its Application* (Beijing, 1993).

[7] O. Ore: Note on Hamilton circuits, *Amer. Math. Monthly* **67** (1960), 55.

Guojun Li

*School of Mathematics and System Sciences*
*Shandong University*
*Jinan 250100, China*
gjli@sdu.edu.cn
*and*
*Institute of Software*
*Chinese Academy of Sciences*
*Beijing 100080, China*
*and*
*CSBL, Department of Biochemistry and Molecular Biology*
*University of Georgia*
*GA 30605, USA*

Ying Xu

*CSBL*
*Department of Biochemistry and Molecular Biology*
*University of Georgia*
*GA 30605*
*USA*
xyn@bmb.uga.edu

Chuanping Chen

*Institute of Mathematics and System Sciences*
*Chinese Academy of Sciences*
*Beijing 100080*
*China*

Zhenhong Liu

*Institute of Mathematics and System Sciences*
*Chinese Academy of Sciences*
*Beijing 100080*
*China*