New Models of Graph-Bin Packing

Cs. Bujtás[a,e], Gy. Dósa[b], Cs. Imreh[c], J. Nagy-György[d], Zs. Tuza[b,e]

[a] *Department of Computer Science and Systems Technology, University of Pannonia, H-8200 Veszprém, Egyetem u. 10, Hungary*
[b] *Department of Mathematics, University of Pannonia, H-8200 Veszprém, Egyetem u. 10, Hungary*
[c] *Department of Informatics, University of Szeged, H-6720 Szeged, Árpád tér 2, Hungary*
[d] *Department of Mathematics, University of Szeged, H-6720 Szeged, Aradi Vértanúk tere 1, Hungary*
[e] *Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, H-1053 Budapest, Reáltanoda u. 13–15, Hungary*

Abstract

In [*Int. J. Found. Computer Sci.* 22 (2011) 1971–1993] the authors introduced a very general problem called Graph-Bin Packing (GBP). It requires a mapping $\mu : V(G) \to V(H)$ from the vertex set of an input graph G into a fixed host graph H, which, among other conditions, satisfies that for each pair u, v of adjacent vertices the distance of $\mu(u)$ and $\mu(v)$ in H is between two prescribed bounds. In this paper we propose two online versions of the Graph-Bin packing problem. In both cases the vertices can arrive in an arbitrary order where each new vertex is adjacent to some of the previous ones. One version is a Maker-Breaker game whose rules are defined by the packing conditions. A subclass of Maker-win input graphs is what we call 'well-packable'; it means that a packing of G is obtained whenever the mapping $\mu(u)$ is generated by selecting an arbitrary feasible vertex of the host graph for the next vertex of G in each step. The other model is connected-online packing where we are looking for an online algorithm which can always find a feasible packing. In both models we present some sufficient and some necessary conditions for packability. In the connected-online version we also give bounds on the size of used part of the host graph.

Keywords: Graph theory, packing, online algorithm, combinatorial games, combinatorial optimization

1. Introduction

In paper [4] a very general problem was introduced, which is a common generalization of many fundamental problems studied to a great extent separately in thousands of papers in the literature of combinatorial optimization, theoretical computer science, graph theory, and operations research. Since *bin packing* and *graph homomorphism* are highly important instances of our problem, we

named it *graph-bin packing*, or *graph H-bin packing* when the underlying structure (host graph) H has to be mentioned explicitly.

The input of the problem is a simple graph $G = (V, E)$ with lower and upper bounds on its edges and weights on its vertices. The vertices correspond to items which have to be packed into the vertices (bins) of a host graph, such that each host vertex can accommodate at most L weight in total, and if two items are adjacent in G, then the distance of their host vertices in H must be between the lower and upper bounds of the edge joining the two items.

It has been noticed in [4] that Graph-Bin Packing is a very general framework; it includes many widely studied algorithmic problems like Bin Packing (with or without conflicts), Scheduling (with or without incompatible jobs), Graph Homomorphism, Subgraph- and Induced Subgraph Isomorphism, (k, d)-Coloring, Distance-Constrained Labeling, Channel Assignment, Bandwidth, Partition, and 3-Partition as subproblems expressible by suitable choices of the parameters.

In [4] some results are presented about the complexity of deciding whether G is packable into a given host graph, and some sufficient and also some necessary conditions are given on packability. The optimization version where the goal is to minimize the size of a connected extension[1] of the used part of the host graph is also studied.

We note that there are further papers considering extensions of the bin packing problem to graphs. The most closely related problem is the 'polyp packing problem' studied in [11, 12] where the goal was to embed some paths in a vertex-disjoint or edge-disjoint way into some copies of a host graph. The host graphs, which were called polyps, consist of simple paths of the same length with a common endpoint. Those papers presented some complexity results and studied the extensions of the well-known First-Fit bin packing algorithm. Another problem, where the graphs present constraints on the possible packing, is the 'bin packing with conflicts' model (see [5, 6]) where the items adjacent in the conflict graph cannot be packed into the same bin. A version where the graph presents only restrictions on the order in which the items can be packed into a bin is studied in [2].

An offline algorithm completely knows the input, and is allowed to use all pieces of information before a decision is made. In this paper we will consider definitions of packing the input graph into the host graph sequentially, where the packing algorithm has less power. On the other hand, in order to exclude some rather trivial classes of negative problem instances, we will consider only connected input sequences which are vertex orders where each prefix induces a connected subgraph of the input graph.

In Section 2 the notion of well-packable graphs is introduced and studied. First we define a Maker-Breaker game where two players are packing the input graph into the host graph, one player wants to find a suitable packing at the

[1] It means a *connected* subgraph G^+ of H, that contains all those vertices of H into which at least one vertex of G has been packed.

end and the other wants to prevent the packing. An input graph is called well-packable if the packing can be finished independently of the strategies of the players. We present several sufficient and some necessary conditions in special classes of graphs for being well-packable.

Section 3 is devoted to the study of connected-online packing. A graph is called connected-online packable if there exists an online algorithm that finds a feasible packing for every connected-online input sequence. We present some simple structural statements about connected-online packable graphs, and also give some results about the optimization version of the problem.

We close the paper with collecting some further questions in Section 4.

1.1. Standard notation.

In a graph G the *distance* between two vertices x and y is defined as the length[2] of a shortest x–y path, and is denoted by $d_G(x, y)$. The diameter of graph G is $diam(G) = \sup_{x,y \in V(G)} d_G(x, y)$. A *geodesic path* means an x–y path of length $d_G(x, y)$.

The two-way infinite path is denoted by P_∞. As usual, we denote by C_n the cycle of length n, and by P_n the path on n vertices.

If $v \in V(G)$ then $G - v$ denotes the graph obtained from G by deleting v and its incident edges.

1.2. Problem definition for Graph H-Bin Packing

The problem called Graph H-Bin Packing can be specified with the following components:

- Host graph, H

  It is a fixed *connected* graph $H = (X, F)$ with vertex set X and edge set F, and vertex *capacity* $L > 0$.

  In the general setting the capacity $L(x)$ may depend on $x \in X$, and the edges may have different lengths; but in the present work we assume $L(x) = L$ for all x (possibly $L = \infty$), and unit length for all edges.

- Input graph, G

  It is a *finite* simple graph $G = (V, E)$ in which each vertex $v \in V$ has a given size $s(v)$ $(0 \le s(v) \le L)$ and each edge $e = uv \in E$ has a lower and an upper *edge-length bound* $a(e) = a(u, v)$ and $b(e) = b(u, v)$ which are integers or ∞, such that $0 \le a(u, v) \le b(u, v)$ holds.

- Packing, μ

  A mapping $\mu : V \to X$ is a *packing of* $G = (V, E)$ *into* $H = (X, F)$ if it satisfies the following two conditions.

[2] That is the number of edges, or the sum of their lengths if a length function is given on the edge set of G. Throughout this paper all edges will be assumed to have unit length.

– For each vertex $x \in X$ of H,

$$\sum_{v \in V\,:\,\mu(v)=x} s(v) \leq L.$$

– For each edge $e = uv \in E$ of G,

$$a(u,v) \leq d_H(\mu(u), \mu(v)) \leq b(u,v).$$

If such a packing μ exists, we say that G is *H-packable*, or *packable into H*.

In the following, μ denotes a feasible packing.

In the optimization version we are looking for a packing which uses the least space from the host graph. In this context the following measures can be studied.

- The minimum number of vertices, taken over all μ and all possible connected extension graphs G^+ for $\mu(V)$, is denoted by $n_H(G)$.
- The minimum diameter, taken over all μ and all possible connected extension graphs G^+ for $\mu(V)$, is denoted by $d_H(G)$.
- The minimum radius, taken over all μ and all possible connected extension graphs G^+ for $\mu(V)$, is denoted by $r_H(G)$.

It has been observed in [4] that the assumption

$$L \quad \geq \quad \max_{S \subseteq V:\ \text{if } e \subseteq S \text{ then } a(e)=0} \ \sum_{v \in S} s(v).$$

is equally unrestrictive as the simpler one $L = \infty$ or $s(v) = 0$ for all $v \in V$. That is, the two are replaceable by each other in any assertion, and therefore we shall always use the form $L = \infty$ and will not emphasize later on that the conclusions remain valid under the finite lower bound on L.

It should be noted that under a purely online scenario the condition $L = \infty$ would provide an extra information, in the sense that if $L < \infty$ then the algorithm does not know in advance whether the condition above holds. On the other hand, in our current setting the input graph is completely known, so the validity of the lower bound on L is possible to check for the case if an algorithm makes use of it.

Despite the above, vertices of size 0 may have substantial influence on packability, via their edge-length bounds.

2. Maker-Breaker games and well-packable graphs

Let now $G = (V, E)$ be a *connected* graph, and suppose that G is H-packable. In this subsection we deal with the question, "how easily" is G packable into

H. This question can be investigated (among other options) in the following context:

We define a Maker-Breaker game regarding the packability of the input graph G. Maker-Breaker games are widely studied in combinatorial game theory (see [1, 9] for a general overview). Two players, M (Maker) and B (Breaker), alternately pack the vertices of G into the host graph, packing one vertex at a time. The first move is done by M, he can choose any vertex of G to be packed first. In any turn afterwards, each vertex chosen by the players has to be adjacent in G to at least one of the previously packed ones. The packing of each vertex has to be feasible: the lower and upper bounds on the edges and also the upper bound L regarding bin capacity must be satisfied in every step. It is allowed for B to pass at any time, but M has to make a feasible move on his turn whenever there is at least one.

The purpose of M is to pack the entire graph, the purpose of B is to prevent M from this. The winner is M if at the end of the game the entire graph G is packed into H (i.e., B was not able to prevent this packing), and B wins if there comes a moment when the next step is impossible: there is no feasible choice for the position of any unpacked vertex of G for the next player.

Certainly, for every finite input graph G and every fixed host graph H, precisely one of M and B has a winning strategy. We have not analyzed whether any anomalies occur if B is not allowed to pass (but M may or may not), or if the first move is done by B. On the other hand, obviously, allowing both M and B to pass would not increase the chances of M to win, because B may e.g. apply the strategy of passing in each step when M has just passed.

Problem 1. *What conditions are necessary or sufficient to ensure a winning strategy for M?*

The following notion intends to describe the "best" instances from the viewpoint of M.

Definition 2. *A connected graph G is* well-packable *into H if the entire graph G will be packed into H at the end, no matter what strategies M and B apply.*

In other words, if a graph is well-packable, it means that starting with any vertex, packing it into any vertex of the host graph, and in any subsequent step continuing with any feasible packing of any vertex adjacent to at least one previously packed vertex, the graph remains packable. As a matter of fact, this property does not need the notion of packing game, and for such input graphs even the clumsiest algorithm reaches a feasible solution (if it respects the rules).

In this section we explore some properties of well-packable graphs.

Obviously, if G is well-packable into H then G is packable into H but the well-packability does not follow from the packability.

Problem 3. *Given a host graph H, what kind of graphs G are well-packable? Is there a characterization of the well-packable graphs?*

Example 4. *Assume that $L = \infty$.*

1. *Let $G = P_n$. G is well-packable into P_∞ and if $b(e) = 1$ for all $e \in E(G)$ then G is well-packable into P_n (and into P_k for all $k \geq 2$).*
2. *Let $n > 4$ and $G = C_n$. It is easy to see that if $b(e) = 1$ for all $e \in E(G)$ then G is not well-packable into C_n. (We note that if $a(e) = 0$ for all $e \in E(G)$, or $a(e) = 1$ for all $e \in E(G)$ and n is even then M has a winning strategy; if $a(e) = 1$ for all $e \in E(G)$ and $n \geq 5$ is odd then B has a winning strategy.)*
3. *Let $G = C_n$, $a(e) = 0$ and $b(e) = 1$ for all $e \in E(G)$. For every $n \geq 3$, G is well-packable into P_1, P_2 and P_3. On the other hand, if $n \geq 5$, G is not well-packable into P_∞ and into any P_k with $k \geq 4$ either. This example illustrates the interesting fact that if $H \subset H'$ then the well-packability into H' does not follow from the well-packability into H (however the packability into H' follows from the packability into H).*
4. *If $a(e) = 0$ for all $e \in E(G)$ then G is well-packable into P_1.*
5. *If $a(e) = 0$ and $b(e) = \infty$ for all $e \in E(G)$ then G is well-packable into each host graph with at least one vertex.*

The following proposition states that concerning well-packability, even the edges with non-restrictive bounds $a(e) = 0$ and $b(e) = \infty$ may be important. This is because of the restriction that after each step a *connected* subgraph has to be packed.

Proposition 5. *Let G be an input graph and H a host graph. Consider the graph G' with $V(G') = V(G)$, $E(G') = \binom{V(G)}{2}$, and define for each $e \in E(G')$ the edge-length bounds*

$$a(e) = \begin{cases} a(e) & e \in E(G) \\ 0 & e \notin E(G) \end{cases}, \quad b(e) = \begin{cases} b(e) & e \in E(G) \\ \infty & e \notin E(G) \end{cases}.$$

(*i*) *G is packable into H if and only if G' is packable into H.*

(*ii*) *If G' is well-packable into H then G is well-packable into H.*

(*iii*) *For every connected nontrivial host graph H, there exists an input graph G such that G is well-packable into H but G' is not well-packable into H.*

Proof. Statements (*i*) and (*ii*) are trivial. To prove (*iii*) we give a very simple example with two different edge-length functions. First consider a connected graph H of diameter at least 3 and the path $G = v_1v_2v_3$. Let us define $a(e) = b(e) = 1$ for all $e \in E(G)$ and let $L = \infty$. Since the capacity is infinite, and for every $x \in V(H)$ there is some vertex $y \in V(H)$ at distance 1 apart, G is well-packable into H. On the other hand, if the vertices of G' arrive in the order v_1, v_3, v_2, and $d_H(\mu(v_1), \mu(v_3)) \geq 3$, then there is no feasible packing for v_2. Thus, G' is not well-packable into H. If $diam(H) = 1$ or 2, we set $G = v_1v_2v_3$ and $L = \infty$ again, but here $b(e) = 0$ is assumed for all $e \in E(G)$. Then, G is

well-packable into H, but G' is not. The latter is shown by the packing where v_1 and v_3 arrive first and they are mapped into different vertices of H. □

Below we give some sufficient conditions, and also some necessary conditions.

Proposition 6. *Suppose that in the input graph $G = (V, E)$ we have $b(e) = \infty$ for all edges $e \in E$, $s_i \leq L$ for all vertices $v_i \in V$, and the diameter of the host graph H is ∞. Then G is well-packable into H.*

Proof. In a connected graph of infinite diameter, for every finite set of vertices there is a vertex at arbitrarily large distance apart. Thus, if the upper edge-length bounds are non-restrictive, then in each step there exists a vertex in H that respects all constraints on the lower edge-length bounds for the next vertex of G, and into which no vertex has been packed yet. □

Proposition 7. *Let $H = P_\infty$ with $L = \infty$. Then G is well-packable into H if and only if each biconnected subgraph of G is well-packable into H.*

Proof. It is enough to show that if each biconnected component[3] of G is well-packable into H then G is well-packable too. The other direction follows immediately by the definition. Suppose, to the contrary, that each biconnected component of G is well-packable into H and G is not well-packable into H. Then there is a vertex $v \in V(G)$ and a packing strategy such that v is not packable with this strategy. First observe that there must be at least two packed neighbors of v, otherwise we could not encounter contradicting bounds. Moreover by the connectivity condition of the packed vertices there is a biconnected component of G containing v and its packed neighbors because for any two packed neighbors of v there is at least one cycle containing them together with v. Therefore, since we cannot pack v, we get a starting strategy for this biconnected component which cannot be finished with v, and this yields a contradiction. □

Corollary 8. *If G is a tree and $L = \infty$, then G is well-packable into every H whose diameter is at least $2 \max_{e \in E} a(e) - 1$. In particular, G is well-packable into P_∞.*

Proof. In each step we have to satisfy the bounds $a(e)$ and $b(e)$ only for a single edge $e = uv$. The condition on the diameter of H ensures that H contains at least one vertex x at distance exactly $a(e)$ apart from $\mu(v)$. Since $L = \infty$, vertex u can be packed into x. □

We derive two further implications of this corollary.

Corollary 9. *Suppose that for each cycle C in G for every edge e in C we have $b(e) = 0$. Then G is well-packable into P_∞ with $L = \infty$.*

[3] often called *block* in the literature

Corollary 10. *Suppose that for each cycle C in G for every edge e in C we have $b(e) = \infty$. Then G is well-packable into P_∞ with $L = \infty$.*

In general, the conditions $L = \infty$ and $a(e) = 0$ for every $e \in E$ are not sufficient for well-packability:

Proposition 11. *For every graph G containing at least one cycle, there exists a function $b : E \to \mathbb{N}$ such that G together with the bounds $b(e)$ and $a(e) = 0$ (for every $e \in E$) and with $L = \infty$ is not well-packable into P_∞.*

Proof. Choose an edge e_0 of a cycle of length ℓ and let $b(e_0) = \ell$, whilst for every edge $e \neq e_0$ define $b(e) = 1$. If the ends of e_0 are packed at distance ℓ in the first step, then the packing cannot be completed under the given constraints. □

We have already observed in part 4 of Example 4 that, under the conditions of Proposition 11, graph G is well-packable into a host graph with one vertex. Further, it is well-packable into any host graph if $b(e) = 0$ for every $e \in E(G)$.

Motivated by the construction in the previous proof, the following condition is obtained:

Theorem 12. *Suppose that G is well-packable into H with $diam(H) = \infty$, and let $C = u_1u_2 \dots u_k$ be any cycle in G. If the set $\{u_2, \dots, u_k\}$ induces a path in G, then the inequality*

$$b(u_k, u_1) + b(u_1, u_2) \geq \sum_{i=2}^{k-1} b(u_i, u_{i+1}) \tag{1}$$

has to hold. Moreover if $a(e) = 0$ for all $e \in E(G)$ and condition (1) holds for all cycles C of G then G is well-packable into P_∞ with $L = \infty$.

Proof. Recall that if G is well-packable then each induced subgraph of G is well-packable into H. Let us observe further that if some cycle with two of its edges violates (1), then there also exists an *induced* cycle with the same property. Since $u_2u_3 \dots u_k$ is an induced path, we can start with packing these vertices in this order into a long geodesic path of H, going in a fixed direction, such that for each $3 \leq i \leq k$, vertex u_i is packed at distance $b(u_{i-1}, u_i)$ after $\mu(u_{i-1})$. Hence, the distance $d_H(\mu(u_2), \mu(u_k))$ equals $\sum_{i=2}^{k-1} b(u_i, u_{i+1})$, and if the inequality in the proposition is not valid, vertex u_1 cannot be packed properly.

For the second part of the statement suppose, to the contrary, that G is a minimal counterexample; that is, G is not well-packable but each proper induced subgraph of G is well-packable into P_∞. It is clear that $n = |V(G)| > 3$.

By Proposition 7 we can assume that G is biconnected, otherwise it would have a biconnected subgraph which is not well-packable and this contradicts the minimality of the counterexample. Moreover by the minimality assumption, v_n is the only unpacked vertex in G; the other vertices can be packed using

arbitrary strategy. We denote by μ the packing obtained for $G - v_n$. By the biconnectivity condition there is $1 \leq j < n$ such that $G - \{v_j, v_n\}$ is connected, so it is well-packable. Moreover, since G is a minimal counterexample, $G - \{v_j\}$ is also well-packable into P_∞. Then we can suppose that we pack the vertices of $G - \{v_j\}$ in the order which is ended by v_n. So there is a vertex x of P_∞ such that $d_{P_\infty}(\mu(v_i), x) \leq b(v_i, v_n)$ for all $i \neq j$. Choose among these vertices the vertex x where $d_{P_\infty}(\mu(v_j), x)$ is smallest. Since we are considering a counterexample, we have $d_{P_\infty}(\mu(v_j), x) > b(v_j, v_n)$. Further, since we minimized $d_{P_\infty}(\mu(v_j), x)$, there is an $1 \leq \ell < n$ such that $d_{P_\infty}(\mu(v_\ell), x) = b(v_\ell, v_n)$, and $\mu(v_\ell)$ and $\mu(v_j)$ are on the opposite sides of x in P_∞. Therefore $d_{P_\infty}(\mu(v_\ell), \mu(v_j)) = d_{P_\infty}(\mu(v_\ell), x) + d_{P_\infty}(\mu(v_j), x) > b(v_\ell, v_n) + b(v_j, v_n)$. Then using an induced path between v_j and v_ℓ, and applying the fact that μ is a valid packing of these vertices, we get a contradiction to condition (1). □

The following example shows that the above condition is not sufficient if there is an $e \in E(G)$ with $a(e) > 0$. Let $L = \infty$, $V(G) = \{v_1, \ldots, v_5\}$,

$$E(G) = \{v_i v_{i+1} \mid i = 1, 2, 3, 4\} \cup \{v_1 v_5, v_2 v_5, v_3 v_5\},$$

$$\begin{array}{ccccccc} b(v_1v_2) & = & b(v_2v_3) & = & b(v_3v_4) & = & 2, \\ & & b(v_4, v_5) & = & b(v_1v_5) & = & 3, \\ & & b(v_2v_5) & = & b(v_3v_5) & = & 1. \end{array}$$

If $a(e) = 0$ for all $e \in E(G)$, then G is well-packable; if $a(e) = b(e)$ for all $e \in E(G)$, then G is not well-packable (however it is packable) into P_∞.

We also note that the condition above is not necessary for packability: if $G = C_5$ with $a(e) = 0$ and $b(e) = 1$ for all $e \in E(G)$, then G is packable into P_∞.

Corollary 13. *Let H be a host graph with $diam(H) = \infty$. If the graph G is well-packable into H, then for every induced cycle C_k of length k in G, the following properties have to hold:*

- *(i) If C_k contains an edge e_i with $b(e_i) = \infty$, then no two consecutive edges in the cycle can have finite upper bounds $b(e_j), b(e_{j+1}) < \infty$.*
- *(ii) For $k = 3$, the upper edge-length bounds of every C_3 have to satisfy the triangle inequality.*
- *(iii) For $k = 4$, if the edges of C_4 are e_1, e_2, e_3, e_4 (in this order) and each of them has finite upper bound, then $b(e_1) = b(e_3)$ and $b(e_2) = b(e_4)$ must hold.*
- *(iv) If $k \geq 5$, and every edge of C_k has finite upper bound, then for each edge e of C_k the equality $b(e) = 0$ has to hold.*

Moreover if $L = \infty$, $a(e) = 0$ for all $e \in E(G)$, and properties (i)–(iv) hold for every cycle C_k of length k in G, then G is well-packable into P_∞.

Proof. The conditions (i)–(iv) simultaneously are equivalent to the following condition: if $C = u_1 u_2 \ldots u_k$ is any induced cycle in G, then inequality (1) has to hold. □

3. Connected-online input sequences

Between the classes of packable graphs and well-packable graphs, there is an intermediate class interesting on its own. The concept applies to a large variety of online graph problems.

Definition 14. *A* connected-online input sequence *of a connected graph $G = (V, E)$ is an ordering $v_1, \ldots, v_n$ of V such that the set $\{v_j \mid 1 \leq j \leq i\}$ induces a connected subgraph of G for all $1 \leq i \leq n$. We say that a connected input graph of* Graph H-Bin Packing *is* connected-online packable *into H if there exists an on-line algorithm that finds a feasible packing for every connected-online input sequence of G.*

It should be emphasized that the entire graph G and all its edge-length bounds $a(e), b(e)$ are known already at the beginning, but each vertex v_i arrives without the indication of its label (name), only its size s_i and the adjacencies to the earlier vertices v_j with $1 \leq j < i$ and the corresponding values $a(v_iv_j), b(v_iv_j)$ are revealed upon arrival. For example, if all the s_i are equal then we have zero information in the first step, which vertex of G is v_1; and similarly, if also all pairs $(a(e), b(e))$ are the same, then after the arrival of v_1 and v_2 we do not know which edge of G corresponds to v_1v_2.

3.1. Connected-online packability

Problem 15. *Given a host graph H, what kind of graphs G are connected-online packable into H ? Is there a characterization of the connected-online packable graphs?*

Obviously if G is well-packable into H then G is connected-online packable into H, and if G is connected-online packable into H then G is packable into H. Therefore the sufficient conditions of well-packability (see for example Propositions 6 and 8) are sufficient conditions of connected-online packability, moreover the necessary conditions of packability (for example $b(v_1v_2) + b(v_2v_3) \geq a(v_1v_3)$ if $v_1v_2, v_2v_3, v_1v_3 \in E(G)$) are necessary conditions of connected-online packability.

Example 16. *Assume that $L = \infty$.*

1. *Let G be a tree. If $b(e) = 1$ for all $e \in E(G)$ then (by Corollary 8) G is well-packable into P_∞ so it is connected-online packable into P_∞ (and also into P_k for all $k \geq 2$). Similarly, assuming $b(e) = 1$, every bipartite graph is connected-online packable into any H with at least one edge.*
2. *Let $G = C_n$. It is easy to see that if $b(e) = 1$ for all $e \in E(G)$ then G is connected-online packable (but if $n \geq 5$ then it is not well-packable) into C_n.*
3. *Let $n > 4$ and $G = C_n$, $a(e) = 0$ and $b(e) = 1$ for all $e \in E(G)$. Then G is connected-online packable into P_k for all $k \geq 1$ (but if $k \geq 4$ then it is not well-packable).*

4. *If $a(e) = 0$ for all $e \in E(G)$ then G is connected-online packable into* every *H (but not necessarily well-packable).*
5. *Let G be a C_6 plus an edge e' connecting two antipodal vertices, with $a(e) = b(e) = 1$ if $e \neq e'$ and $a(e') = b(e') = 3$. Then G is packable but not connected-online packable into P_∞.*

Remark 17. *Two further simple cases of connected-online packability can be mentioned.*

- *If the edges of G can be identified (e.g. the pairs $(a(e), b(e))$ are distinct) then G is connected-online packable into H if and only if G is packable into H.*
- *Let G be a clique with $a(e) = a$ and $b(e) = b$ for all $e \in E(G)$, and let H be an arbitrary host graph. Then G is connected-online packable into H if and only if G is packable into H.*

We note that if $H \subset H'$ then the connected-online packability into H' follows from the connected-online packability into H. This monotonicity is not true, however, for well-packability.

The following statement, which considers the same question as Proposition 5 for well-packing, states that concerning connected-online packability, even the edges with non-restrictive bounds $a(e) = 0$ and $b(e) = \infty$ may be important. This is because of the restriction that after each step a *connected* subgraph has to be packed. Part (i) is repeated for the sake of completeness.

Proposition 18. *Let G be an input graph and H a host graph. Consider G' with $V(G') = V(G)$, $E(G') = \binom{V(G)}{2}$, and if $e \in E(G')$ then let*

$$a(e) = \begin{cases} a(e) & e \in E(G) \\ 0 & e \notin E(G) \end{cases}, \quad b(e) = \begin{cases} b(e) & e \in E(G) \\ \infty & e \notin E(G) \end{cases}.$$

(i) *G is packable into H if and only if G' is packable into H.*

(ii) *If G' is connected-online packable into H then G is connected-online packable into H.*

(iii) *There is an input graph G such that G is connected-online packable into H but G' is not connected-online packable into H.*

Proof. Statements (i) and (ii) are trivial. To prove (iii) consider $G = P_4 = wxyz$ with $a(e) = b(e) = 1$ for all $e \in E(G)$, and let $L = \infty$. Obviously G is connected-online packable into P_∞ but G' is not connected-online packable into P_∞ because if v_1v_2 is a non-edge, we cannot know whether it is wy or wz. □

On the other hand we note that a statement similar to Proposition 7 cannot be formulated in case of connected-online packability. To see this, consider a graph which consists of two cycles of length $2k$ sharing a vertex. For one of

the cycles each edge e has $a(e) = b(e) = 1$, for the other cycle we have an edge f not containing their common vertex with $a(f) = b(f) = 2k - 1$ and for the other edges $a(e) = b(e) = 1$. Then the biconnected subgraphs are the cycles and it is easy to see that both of them are connected-online packable into P_∞. The vertices of the first cycle have to be packed into two adjacent vertices of the host graph, the vertices of the second cycle have to be packed into different vertices in the host graph. On the other hand when we receive a part of one of the cycles from the graph, we do not know from which cycle it is taken, thus we do not know which strategy should be used. Therefore the entire graph is not connected-online packable.

It is easy to see that the following propositions hold.

Proposition 19. *G is connected-online packable into H if and only if each induced subgraph of G is connected-online packable into H.*

Proposition 20. *If a graph G is connected-online packable into P_∞, then it is connected-online packable into any host graph which has infinite diameter (and the same capacity L).*

Proof. Since $diam(H) = \infty$, we can identify two vertices $x, y \in V(H)$ and a geodesic path between them such that $d_H(x, y) \geq 1 + 2\sum_{e \in E(G)} b(e)$. (In fact, a shorter geodesic x–y path of H would also suffice to start with.) Packing the first vertex from the input sequence into the central vertex of the geodesic path, we can apply the same strategy as for the host graph P_∞. □

The following proposition belongs to the case where the packing problem models graph coloring. We denote by $\chi_{OL}(G)$ the *online chromatic number* of graph G, which is the minimum number of colors any online algorithm can use to properly color G when the vertices arrive in an unknown order.

Proposition 21. *Suppose $H = K_k$ and $L = \infty$. Then a graph $G = (V, E)$ with $a(e) = 1$ for all $e \in E$ is*

- *H-packable if and only if $\chi(G) \leq k$;*
- *connected-online packable into H if $\chi_{OL}(G) \leq k$, but this condition is not necessary.*

Proof. We can define the vertices of the host graph as the color classes. Therefore, if a coloring or online coloring exists then we can use the coloring algorithm to pack the items. In case of packability a packing algorithm can be transformed into a coloring algorithm, thus we have equivalence. In case of online packing the assumption that we always have a connected subgraph might make the problem easier than online coloring, as the following example shows. It is known (see [8]) that there is a tree T_k on 2^{k-1} vertices with $\chi_{OL}(T_k) = k$, but it is easy to see that every connected bipartite graph is connected-online 2-colorable. □

3.2. Optimization problems

As usual, we measure the performance of online algorithms by competitive ratios. For a minimization problem (which are the determination of $n_H(G)$, $d_H(G)$, and $r_H(G)$) an online algorithm is called *α-competitive* if, on any input, the value of solution found by the algorithm is at most α times the offline optimum. We refer to [3, 7, 10] for an overview on competitive analysis. Note that α needs not be a constant, it may get larger as input size tends to infinity.

We take our first example from graph coloring, which corresponds to the assumptions $H = P_\infty$, $L = \infty$, moreover $a(e) = 1$ and $b(e) = \infty$ for every edge e of the input graph. Then $\chi(G) = n_H(G)$.

Remark 22. *Restricting the input sequences to connected ones may or may not have a substantial consequence on approximability, as shown by the following two problem classes.*

1. *Despite that the maximum online chromatic number of trees of order n is $\Omega(\log n)$, every tree is connected-online 2-colorable.*
2. *It follows from the next proposition that the connected-online chromatic number of 2-trees[4] of order n is $\Omega(\log n)$. All k-trees are chordal, and their chromatic number is equal to $k+1$ except that $\chi(K_k) = k$.*

The second part of this remark can be stated in the following more general assertion. The idea is applicable to many other types of online graph problems, too, but in the proposition we again restrict ourselves to coloring. In notation, the *complete join* $G' + G''$ of two graphs has vertex set $V(G') \cup V(G'')$ (vertex-disjoint union) and edge set $E(G') \cup E(G'') \cup \{v'v'' \mid v' \in V(G'),\, v'' \in V(G'')\}$.

Proposition 23. *For a class $\mathfrak{G}$ of graphs, define $\mathfrak{G}^+ := \{G + K_1 \mid G \in \mathfrak{G}\}$. If the graphs of order n in $\mathfrak{G}$ have online chromatic number at least f_n, then the graphs of order $n+1$ in $\mathfrak{G}^+$ have connected-online chromatic number at least $f_n + 1$.*

Proof. To each online input sequence of any $G \in \mathfrak{G}$ we can associate a connected-online input sequence of $G + K_1$ by choosing the vertex of K_1 first. Since the color of this vertex cannot occur in G, the lower bound follows. □

In the previous section we have seen in Corollary 8 that, under the assumption $L = \infty$, if G is a tree with $b(e) = \infty$ for all $e \in E(G)$ and $diam(H) \geq 2\max_{e \in E(G)} a(e) - 1$ then G is well-packable into H. Here we prove a slightly stronger bound under the connected-online scenario on trees. If G is a connected graph and A is an online algorithm, then denote by $d_H^A(G)$ the diameter of packing G into H by A; and let $d_H^{OL}(G) = \min_A d_H^A(G)$.

[4] A *k-tree* is a graph that can be built recursively from the complete graph of order k, by joining a new vertex in each step to a complete subgraph of order k in the part constructed previously.

Theorem 24. *Restrict the input of the connected-online problem to the class of trees. Suppose that $H = P_\infty$ and $L = \infty$. Then*

$$d_H^{OL}(G) \leq \max_{e,e' \in E,\ e \neq e'} a(e) + a(e') - 1$$

holds for every input graph $G = (V, E)$, independently of the upper edge-length bounds $b(e)$.

Moreover there is a tree G with

$$d_H^{OL}(G) = \max_{e,e' \in E,\ e \neq e'} a(e) + a(e') - 1.$$

Proof. We view the vertices of P_∞ as the integer points of the real line. In this way a closed interval $[p, q]$ will represent the set $\{x_i \mid i \in \mathbb{Z},\ p \leq i \leq q\}$.

Assume that the vertices arrive in a sequence $v_1, \ldots, v_n$ such that the graph induced by $\{v_j \mid 1 \leq j \leq i\}$ is a tree (i.e., connected) for each $i = 1, \ldots, n$. We will find a mapping μ with the following properties: If $v_i v_j \in E(G)$ then $|\mu(v_i) - \mu(v_j)| = a(v_i v_j)$, moreover if x_p (x_q) is the first (respectively, last) vertex into which some vertex has been packed, then there exist edges $e(p), e(q) \in E$ such that

- $x_p \in \mu(e(p))$ and $x_q \in \mu(e(q))$,
- assuming $\mu(e(p)) = \{x_p, x_{p'}\}$ and $\mu(e(q)) = \{x_{q'}, x_q\}$, we have $q' < p'$ ($\mu(e(p))$ and $\mu(e(q))$ are overlapping).

This situation is easily achieved after the proper packing of v_1, v_2, v_3 if we ensure that the interval of one of the two edges contains the other. We are going to prove that the conditions can be maintained in every step.

Suppose that the packing has been made for $v_1, \ldots, v_i$, and v_{i+1} arrives next, connected to the current subtree with the edge $e_i = v_{i+1} v_j$. Let $a_i := a(e_i)$ and suppose $\mu(v_j) = x_r$.

If $r < p'$ we define $\mu(v_{i+1}) = x_{r+a_i}$. If $r + a_i \leq q$ then the properties remain true. If $r + a_i > q$ then x_{r+a_i} will be the last vertex into which a vertex has been packed, moreover $\mu(e(p))$ and $\mu(e(r + a_i)) = \mu(e_i)$ are overlapping, so the properties remain true.

If $r \geq p' > q'$ we define $\mu(v_{i+1}) = x_{r-a_i}$. If $r - a_i \geq p$ then the properties remain true. If $r - a_i < p$ then x_{r-a_i} will be the first vertex into which a vertex has been packed, moreover $\mu(e(q))$ and $\mu(e(r - a_i)) = \mu(e_i)$ are overlapping, so the properties remain true.

Since the actual intervals belonging to the two extremal edges share at least two vertices after each step, the diameter of the packing can never exceed $\max_{e,e' \in E,\ e \neq e'} a(e) + a(e') - 1$.

To show the tightness part of the theorem, consider P_4 with the bounds $a(v_1 v_2) = b(v_1 v_2) = 2$, $a(v_2 v_3) = b(v_2 v_3) = 1$, $a(v_3 v_4) = b(v_3 v_4) = 2$. First v_1 and v_2 arrive any algorithm has to pack them to vertices of distance 2. Then the next vertex is v_3. If the algorithm packs it in a way that $d_H(\mu(v_1), \mu(v_3)) = 3$ then we are done. Otherwise v_3 is packed to the vertex in P_∞ which is between

$\mu(v_1)$ and $\mu(v_2)$. Then v_4 arrives and after packing it either $d_H(\mu(v_1), \mu(v_4)) = 3$ or $d_H(\mu(v_2), \mu(v_4)) = 3$ will be valid.

We can prove that for any $N > 1$ there is a path P_N with $\max_{e \in E(P_n)} a(e) = N$ for which $d_H^{OL}(P_N) \geq 2N - 1$. At first consider an arbitrary connected-online algorithm A and suppose, to the contrary, that if P_N is a path with $\max_{e \in E(P)} a(e) = N$ then $d_H^A(P_N) = M < 2N - 1$. Then there is an input sequence $v_1, \ldots, v_m$ with $\max_{i<m} d_H(\mu_A(v_i), \mu_A(v_m)) = M$. Now give edges with $a(v_m v_{m+1}) = \lceil M/2 \rceil$ and $a(v_{m+1} v_{m+2}) = \lceil M/2 \rceil + 1$. Algorithm A cannot pack v_m and v_{m+1} with $\max_{i<j \leq M+2} d_H(\mu_A(v_i), \mu_A(v_j)) \leq M$ so this is a contradiction. It is easy to see that there is a function f such that the length of $P_N \leq f(N)$, therefore there is a path P_N with finite length and $\max_{e \in E(P)} a(e) = N$ and $d_H^{OL}(P_N) \geq 2N - 1$. □

The following example shows that if G is not a tree then we can achieve much worse bound: set $G = K_n$ with $a(e) = N$, $b(e) = \infty$ for all $e \in E(G)$, $H = P_\infty$ and $L = \infty$. Then G is connected-online packable into H, moreover $d_H(G) = (n-1)N$. However Lemma 24 has a corollary in a special case.

Corollary 25. *If $H = P_\infty$, $L = \infty$, and for every $e \in E(G)$ there is at most one C_e cycle in G containig e and*

$$b(e) \geq \max_{e', e'', e''' \in E(C_e)} a(e') + a(e'') + a(e''') - 1$$

then G is connected-online packable into H.

Proof. If the given subgraph is a tree then we can use the algorithm of Lemma 24, so if a vertex closing a cycle arrives then it is easy to pack it. After the arrival of a cycle delete the last two edges of it and use the algorithm on the components of the remaining graph. It will work well because the deleted edges will not be in another cycle. □

4. Concluding remarks and further problems

In this section we summarize the most important open questions concerning the models introduced in our paper.

Problem 26. *Given a host graph H and a class $\mathfrak{G}$ of graphs with vertex sizes and edge-length bounds, what is the complexity of determining whether a generic input graph $G \in \mathfrak{G}$ is*

- *H-packable?*
- *well-packable into H?*
- *connected-online packable into H?*

Problem 27. *What conditions are necessary and/or sufficient for packability, connected-online packability and well-packability if $L < \infty$?*

Problem 28. *What bounds on competitive ratios of connected-online algorithms can one achieve if $L < \infty$?*

Problem 29. *What bounds on $d_H(G)$, $n_H(G)$, and $r_H(G)$ can be achieved if $L < \infty$?*

Another track of research could be the study of analogous problems where we only know that the input graph is taken from a specified graph class $\mathfrak{G}$, but it is not known which graph it is.

It would also be of interest to investigate packings of *directed graphs* into host *di*graphs, but we have not considered this variant here. Moreover, we restricted ourselves to finite input graphs, but at least the existence problem of a packing $G \to H$ where both G and H are infinite would make sense. Infinity raises various complications, however, for instance one can observe that the usual compactness theorem fails to be valid on packing.

Acknowledgements

J. Nagy-György was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP-4.2.4.A/ 2-11/1-2012-0001 'National Excellence Program'. Gy. Dósa was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP-4.2.2.B-15/1/KONV-2015-0004. Research of Cs. Bujtás, Gy. Dósa and Zs. Tuza was supported by the Hungarian Scientific Research Fund NKFIH/OTKA under the grant SNN 116095. Research of Cs. Imreh was supported by the Alexander von Humboldt Foundation.

[1] J. Beck, *Combinatorial Games: Tic-Tac-Toe Theory*, Cambridge University Press, 2008.

[2] A. Bódis, Bin packing with directed stackability conflicts, *Acta Universitatis Sapientiae, Informatica*, **7(1)**, 31–57, 2015.

[3] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, 1998.

[4] Cs. Bujtás, Gy. Dósa, Cs Imreh, J. Nagy-György, and Zs. Tuza, The Graph-Bin Packing problem, *International Journal of Foundations of Computer Science* **22** (2011), 1971–1993.

[5] L. Epstein, Online variable-sized bin packing with conflicts, *Discrete Optimization* **8(2)**, (2011), 333–343

[6] L. Epstein and A. Levin, On bin packing with conflicts, *SIAM J. Optimization*, **19(3)**, 1270–1298, 2008.

[7] A. Fiat and G. J. Woeginger (eds.), *Online algorithms: The State of the Art, LNCS 1442*, Springer-Verlag Berlin, 1998.

[8] A. Gyárfás and J. Lehel, On-line and first-fit colorings of graphs, *Journal of Graph Theory*, **12** (1988), 217–227.

[9] D. Hefetz, M. Krivelevich, M. Stojakovic, and T. Szabó, *Positional Games* Ober- wolfach Seminars, vol. 44. Birkhäuser (2014)

[10] Cs. Imreh, Competitive analysis, In *Algorithms of Informatics Volume 1*, ed. Antal Iványi, mondAt, Budapest 2007, 395–428.

[11] G. Y. Katona, Edge disjoint polyp packing, *Discrete Applied Mathematics*, **78(1-3)**, (1997), 133–152.

[12] G. Y. Katona, Vertex disjoint polyp packing, *Annales Universitatis Scientiarum Budapestinensis de Rolando Eotvos Nominatae Sectio Computatorica*, **21**, (2002), 81–118.