Discrete Applied Mathematics 161 (2013) 1686–1698

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Packing paths: Recycling saves time

Daniel Binkele-Raible *, Henning Fernau

Universität Trier, FB 4—Abteilung Informatik, D-54286 Trier, Germany

ARTICLE INFO

Article history:
Received 12 October 2010
Received in revised form 5 July 2011
Accepted 8 November 2011
Available online 7 February 2012

Keywords:
Parameterized algorithms
Graphs
Packings
Extremal graph theory

ABSTRACT

We consider packing problems in graphs under a parameterized perspective. Starting from a maximal path packing $\mathcal{P}$ of size j we use extremal arguments for determining how many vertices of $\mathcal{P}$ appear in a path packing of size $j+1$. Generally, one can re-use $2j$ vertices of j paths of length three, four and five. This is improved to $3j$, $2.5j$ and $3j$ vertices.

1. Introduction and definitions

1.1. Our combinatorial problem

In this paper, P_d denotes a path of length d (with $d+1$ vertices). A *P_d-packing* of a graph $G(V, E)$ is a set of vertex-disjoint copies of a P_d in G. It is maximal if any addition of a further P_d would violate the vertex disjointness property. The algorithmic problem we investigate in this paper is P_d-Packing ($d \geq 3$):

Given $G(V, E)$, and the parameter k.

We ask: Is there a P_d-packing of size k?

Hell and Kirkpatrick [8,13] showed that quite generally (Maximum) H-Packing is $\mathcal{NP}$-complete, where H is a graph with at least three vertices in some connected component. So, the task is to find at least k vertex-disjoint copies of H in the given graph G. A specific case of H-Packing is P_d-Packing, which is therefore $\mathcal{NP}$-complete if $d \geq 2$. The case $d = 1$ corresponds to the classical matching problem, which is polynomial-time solvable.

We tackle the mentioned path packing problems by an iterative approach, which tries to build a new, bigger packing from an existing maximal one. This is not a new idea in itself, and also the famous algorithm of Edmonds [3] for solving P_1-Packing uses this approach. Also deterministic approximation algorithms have been proposed that use this idea. Based on earlier works of Hurkens and Schrijver [10], Bontridder et al. [2] studied P_2-packing, considering a series of heuristics H_ℓ. H_ℓ proceeds in the following manner: it starts from a maximal P_2-packing $\mathcal{P}$ and tries to improve it by replacing ℓ P_2's by $\ell + 1$ P_2's.

When we think of exact algorithms for packing problems, we might relax the described heuristic as follows: Assume we have a path packing $\mathcal{P}$ consisting of j P_d-paths. Can we rely on re-using a certain fraction $\alpha_d \cdot (d+1) \cdot j$, with $\alpha_d \in [0, 1]$, of the $(d+1) \cdot j$ *vertices* of $\mathcal{P}$ to construct a bigger packing $\mathcal{Q}$, still being able to correctly answer that no larger packing exists, should we fail to find one in our restricted search space? Notice that the possibly more direct generalization of the H_ℓ-heuristic to the quest to re-use a certain fraction of the j paths (without re-arrangements of the paths) would not work

* Corresponding author. Fax: +49 6512013954.
E-mail addresses: raible@informatik.uni-trier.de (D. Binkele-Raible), fernau@uni-trier.de (H. Fernau).

0166-218X/$ – see front matter © 2013 Published by Elsevier B.V.
doi:10.1016/j.dam.2011.11.008

when looking for exact solutions. Already in the case of P_1-Packing, the mentioned algorithm of Edmonds works with so-called *augmenting paths*, whose very idea is to rearrange matching edges along a path and so *not* to keep as many of the old P_1's as possible.

So, we propose the combinatorial study of the *re-usability factor* α_d defined above. For instance, Edmonds's result shows that $\alpha_1 = 1$. It should be clear that also for the $\mathcal{NP}$-hard path packing problems, the knowledge of a re-usability factor close to one gives a rich source of knowledge. Then, only a few "new" vertices that are not contained in the packing $\mathcal{P}$ need to be considered to construct $\mathcal{Q}$. This combinatorial question also makes sense for other types of packing problems.

Related to packing problems are partitioning problems, which can be viewed as asking for a packing that uses up every vertex. Slightly generalizing standard terminology from matching theory, we will call such packings *perfect*.

1.2. Previous and new work on re-usability factors

We are not aware of any study of any type of re-usability factor that could claim to have determined any such factor exactly, except from $\alpha_1 = 1$. From an algorithmic perspective, it is quite interesting to find non-trivial lower bounds on the re-usability factor. This is the topic of the present paper.

P_d-Packing could be seen as a special case of $(d+1)$-Set Packing. In that problem, we are given a collection $\mathcal{C}$ of subsets of a finite universe U satisfying $S \in \mathcal{C} \Rightarrow |S| \leq d+1$ and an integer parameter k, and we are facing the task to find (at least) k pairwisely disjoint sets in $\mathcal{C}$. It was shown in [15], although stated in different terminology, that the re-usability factor of 3-Set Packing is at least $\frac{2}{3}$. As we will see in this paper, that result easily generalizes to ℓ-Set Packing, proving that the corresponding re-usability factor is lower bounded by $\frac{2}{\ell}$.

The re-usability factor of P_2-packing was studied by the authors in [5], showing a lower bound $\alpha_2 \geq \frac{5}{6}$. In this paper, we will prove lower bounds on the re-usability factor for P_d-packing, with $3 \leq d \leq 5$. Namely, we show that $\alpha_3 \geq \frac{3}{4}$, $\alpha_4 \geq \frac{1}{2}$, and $\alpha_5 \geq \frac{1}{2}$. To prove our results, we will use extremal combinatorial arguments.

1.3. Algorithmic consequences

Our original aim was to use re-usability to obtain better running times for parameterized algorithms for path packing problems. This means that we are interested in obtaining running time upper bounds of the form $\mathcal{O}(f(k)p(n))$, where p is some polynomial depending on the number of vertices of the input graph, f is some arbitrary (computable) function and k is the number of P_d we seek to find. It is now common to use $\mathcal{O}^*(f(k))$ as a shorthand of a running time estimate of the form above, suppressing in particular polynomial-time factors depending on the overall input size. More technically speaking, a problem is *fixed-parameter tractable* iff it admits an algorithm running in time $\mathcal{O}^*(f(k))$.

How can we exploit our combinatorial results for obtaining better exponential-time parameterized algorithms? Obviously, the combinatorial question is linked to an iterative approach: assume we have a maximal packing $\mathcal{P}$ of size j, try to find a packing $\mathcal{Q}$ of size $j+1$ or report that there is no such packing. Given a lower bound k on the size of the packing we are looking for, we have to do at most k iteration steps.

There are two obvious ways to implement this idea for parameterized algorithms (focussing on graph packing problems in our terminology):

- If we could rely on the graph being of order bounded by some $c \cdot k$, i.e., technically speaking and using standard terminology of the area of parameterized algorithms, if we knew about a linear vertex-kernel, then we could use our knowledge on a lower bound on the re-usability factor by cycling through all subsets S up to a certain size (given by the re-usability factor) within the vertex set $V(\mathcal{P})$ of $\mathcal{P}$ of size j and through vertex subsets S' in the complement of $V(\mathcal{P})$ such that $S \cup S'$ might contain a perfect packing; this has to be verified using, e.g., dynamic programming.

  Based on a published kernel for P_2-Packing containing at most $7k$ vertices, see [20], this strategy did work out for P_2-packings, as described in [5] and led to the currently fastest published deterministic algorithm for P_2-Packing, running in time $\mathcal{O}^*(2.448^{3k})$.[1]
- If no linear vertex-kernel is available, it is not possible to cycle through vertex subsets S' in the complement of $V(\mathcal{P})$ in time $\mathcal{O}^*(f(k))$. However, one can use color coding or similar techniques to look for these sets. The advantage over a direct application of color coding lies in the fact that fewer colors need to be involved in this algorithmically rather expensive part. In this sense, recycling saves time, as the paper title indicates.

Unfortunately, there are no linear kernels known for P_d-Packing when $d \geq 3$, so we should use the second approach for the packing problems we investigate.

[1] It has been recently report in the *Workshop on Kernels* at Bergen in 2009 that a group of researchers obtained a $6k$ kernel for P_2-Packing, which immediately improves the mentioned algorithm to run in time $\mathcal{O}^*(2.371^{3k})$.