Available online at www.sciencedirect.com
SCIENCE DIRECT
ELSEVIER
Discrete Applied Mathematics 154 (2006) 971–979
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

An approximation algorithm for maximum triangle packing

Refael Hassin*, Shlomi Rubinstein

Department of Statistics and Operations Research, Tel Aviv University, 69978 Tel Aviv, Israel

Received 26 April 2004; received in revised form 17 July 2005; accepted 22 November 2005
Available online 27 December 2005

Abstract

We present a randomized $(\frac{89}{169} - \varepsilon)$-approximation algorithm for the weighted maximum triangle packing problem, for any given $\varepsilon > 0$. This is the first algorithm for this problem whose performance guarantee is better than $\frac{1}{2}$. The algorithm also improves the best-known approximation bound for the maximum 2-edge path packing problem.

Keywords: Analysis of algorithms; Maximum triangle packing; 2-edge paths

1. Introduction

Let $G = (V, E)$ be a complete (undirected) graph with vertex set V such that $|V| = 3n$, and edge set E. For $e \in E$ let $w(e) \geqslant 0$ be its weight. For $E' \subseteq E$ we denote $w(E') = \sum_{e \in E'} w(e)$. For a random subset $E' \subseteq E$, $w(E')$ denotes the expected value. In this paper, a *k-path* is a simple k-edge path, and similarly a *k-cycle* is a simple k-edge cycle. A 3-cycle is also called a *triangle*. The MAXIMUM TRIANGLE PACKING PROBLEM is to compute a set of n vertex-disjoint triangles with maximum total edge weight.

In this paper we propose a randomized algorithm which is the first approximation algorithm with performance guarantee strictly better than 0.5. Specifically, our algorithm is an $(\frac{89}{169} - \varepsilon)$-approximation for any given $\varepsilon > 0$.

The MAXIMUM 2-PATH PACKING PROBLEM requires to compute a maximum weight set of vertex-disjoint 2-paths. We improve the best known approximation bound for this problem and prove that our triangle packing algorithm is also a $(\frac{35}{67} - \varepsilon)$-approximation algorithm for this problem.

2. Related literature

The problems of whether the vertex set of a graph can be covered by vertex disjoint 2-paths or vertex disjoint triangles are NP-complete (see [7, p. 76, 192], respectively). These results imply that the two problems defined in the introduction are NP-hard.

The problems considered in this paper are special cases of the 3-SET PACKING PROBLEM. In the unweighted version of this problem, a collection of sets of cardinality at the most three each, is given. The goal is to compute a maximum

* Corresponding author. Tel.: +972 3640 9281; fax: +972 3640 9357.
E-mail address: hassin@post.tau.ac.il (R. Hassin).

0166-218X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2005.11.003

number of disjoint sets from this collection. In the weighted version, each set has a weight, and the goal is to find a sub-collection of disjoint sets having maximum total weight. We now survey the existing approximation results for the 3-SET PACKING PROBLEM, which of course also apply to the problems treated in this paper.

Hurkens and Schrijver [12] proved that a natural local search algorithm can be used to give a $(\frac{2}{3}-\varepsilon)$-approximation algorithm for UNWEIGHTED 3-SET PACKING for any $\varepsilon > 0$ (see also [8]). Arkin and Hassin [1] analyzed the local search algorithm when applied to the WEIGHTED k-SET PACKING PROBLEM. They proved that for $k = 3$, the result is a $(\frac{1}{2}-\varepsilon)$-approximation. Bafna et al. [2] analyzed a restricted version of the local search algorithm in which the depth of the search is also k, for a more general problem of computing a maximum independent set in $(k+1)$-claw free graphs. For $k = 3$ it gives a $\frac{3}{7}$-approximation. A more involved algorithm for k-set packing, that combines greedy and local search ideas was given by Chandra and Halldórsson [4]. This algorithm yields improved bounds for $k \geqslant 6$. Finally, Berman [3] improved the approximation ratios for all $k \geqslant 4$, and for the maximum independent set in $(k+1)$-claw free graphs, however, the bound for $k = 3$ is still 0.5.

Kann [13] proved that the UNWEIGHTED TRIANGLE PACKING PROBLEM is APX-complete even for graphs with maximum degree 4. Chlebík and Chlebíková [5] proved that it is NP-hard to obtain an approximation factor better than 0.9929.

Feder and Subi [6] considered partitioning the vertices of an input graph G into k-sets, each of which induces a subgraph of G isomorphic to a given graph H. Assuming that G is k-partite and each vertex of a chosen subgraph must belong to a given part of G, the problem is polynomial or NP-complete depending on whether H is a forest or not. In particular, it follows that (i) partitioning the vertices of a 3-partite graph into triangles with each vertex in a distinct part is NP-complete; and (ii) partitioning into 2-paths with all middle vertices of the paths in the same given part and the end vertices in distinct other parts is in P.

The MAXIMUM TRAVELING SALESMAN PROBLEM (MAX TSP) asks to compute in an edge weighted graph a Hamiltonian cycle (or *tour*) of maximum weight. MAX TSP is a relaxation of MAX k-PATH PACKING for any k. Therefore, an α-approximation algorithm for the former problem can be used to approximate the latter [10]: simply delete every $(k+1)$-th edge from the TSP solution. Choose the starting point so that the deleted weight is at most $1/(k+1)$ the total weight. The result is an $(\alpha k/(k+1))$-approximation. By applying the $\frac{25}{33}$-approximation randomized algorithm of Hassin and Rubinstein [11] for MAX TSP, we obtain a $(\frac{50}{99}-\varepsilon)$-approximation for $k = 2$. For the MAXIMUM 3-PATH PACKING PROBLEM there is a better $\frac{3}{4}$ bound in [10].

In this paper we give the first algorithm whose performance guarantee is strictly greater than $\frac{1}{2}$ for MAXIMUM TRIANGLE PACKING. Our bound is $(\frac{89}{169}-\varepsilon)$ for every $\varepsilon > 0$. The algorithm is described in Section 3. We also improve the above-mentioned bound for MAXIMUM WEIGHTED 2-PATH PACKING. This result is described in Section 4. Specifically, our algorithm returns a $(\frac{35}{67}-\varepsilon)$-approximation for this problem.

3. Maximum weighted triangle packing

A *(perfect) binary 2-matching* (also called *2-factor* or *cycle cover*) is a subgraph in which every vertex in V has a degree of exactly 2. A *maximum binary 2-matching* is one with maximum total edge weight. Hartvigsen [9] showed how to compute a maximum binary 2-matching in $O(n^3)$ time (see [14] for another $O(n^2|E|)$ algorithm). Note that a 2-matching consists of disjoint simple cycles of at least three edges each, that together span V.

We denote the weight of an optimal weighted triangle packing by *opt*.

Algorithm *WTP* is given in Fig. 1. The algorithm starts by computing a maximum binary 2-matching. Long cycles, where $|C| > \varepsilon^{-1}$ ($|C|$ denotes the number of vertices of C), are broken into paths with at most ε^{-1} edges each, loosing a fraction of at most ε of their weight. To simplify the exposition, the algorithm completes the paths into cycles to form a cycle cover $\mathscr{C}$. The cycle cover $\mathscr{C}$ consists of vertex disjoint cycles $C_1, \ldots, C_r$ satisfying $3 \leqslant |C_i| \leqslant \varepsilon^{-1}+1$ $i = 1, \ldots, r$. Since the MAXIMUM CYCLE COVER PROBLEM is a relaxation of the WEIGHTED MAXIMUM TRIANGLE PACKING PROBLEM,

$$w(\mathscr{C}) \geqslant (1-\varepsilon)opt.$$

Algorithm *WTP* constructs three solutions and selects the best one. The first solution is attractive when a large fraction of *opt* comes from edges that belong to $\mathscr{C}$. The second is attractive when the large fraction of *opt* comes from edges

WTP
input
1. A complete undirected graph $G = (V, E)$ with weights w_{ij} $(i, j) \in E$.
2. A constant $\epsilon > 0$.
returns *A triangle packing.*
begin
Compute a maximum cycle cover $\{C'_1, \ldots, C'_{r'}\}$.
for $i = 1, \ldots, r'$:
  if $|C'_i| > \epsilon^{-1}$
    then
      Remove from C'_i $\lceil \epsilon |C'_i| \rceil$ edges of total weight at most $\frac{\lceil \epsilon |C'_i| \rceil}{|C'_i|} w(C'_i)$,
      to form a set of paths of at most ϵ^{-1} edges each.
      Close each of the paths into a cycle by adding an edge.
  end if
end for
$\mathcal{C} = \{C_1, \ldots, C_r\}$:= *the resulting cycle cover.*
$TP_1 := A1(G, \mathcal{C})$.
$TP_2 := A2(G, \mathcal{C})$.
$TP_3 := A3(G, \mathcal{C})$.
return *the solution with maximum weight among TP_1, TP_2 and TP_3.*
end *WTP*

Fig. 1. Algorithm *WTP*.

whose two vertices are on the same cycle of $\mathscr{C}$. The third solution deals with the remaining case, in which the optimal solution also uses a considerable weight of edges that connect distinct cycles of $\mathscr{C}$.

The first solution is constructed by Algorithm A1 (see Fig. 2). It selects all the 3-cycles of $\mathscr{C}$. From every k-cycle C_i with $k \neq 3, 5$, the algorithm selects 3-sets consisting of vertices of pairs of adjacent edges in C_i of total weight at least $\frac{1}{2} w(C_i)$, to form triangles in the solution. This task can be done by selecting an appropriate edge of the cycle, and then deleting this edge with one or two of its neighboring edges (depending on the value of $|C_i| \bmod 3$), and then every third edge of the cycle according to an arbitrary orientation.

5-cycles obtain special treatment (since the above process would only guarantee $\frac{2}{5}$ of their weight): From each 5-cycle we select three edges. Two of these edges are adjacent and the third is disjoint from the two. The former edges define a triangle which is added to the solution. The third edge is added to a special set E'. The selection is made so that the weight of the adjacent pair plus half the weight of the third edge is maximized. After going through every cycle, a subset of E' of total weight at least $\frac{1}{2} w(E')$ is matched to unused vertices to form triangles. (Since $|V| = 3n$, there should be a sufficient number of unused vertices.) Thus, in total we gain at least half of the cycle's weight.

The second solution is constructed by Algorithm A2 (see Fig. 3). The algorithm enumerates all the possible packings of vertex disjoint 2-sets and 3-sets in the subgraph induced by the vertex set of each cycle $C_i \in \mathscr{C}$. The weight of a subset is defined to be the total edge weight of the induced subgraph (a triangle or a single edge). A dynamic programming recursion is then used to compute the maximum weight of n subsets from the collection. In this formulation, we use $F(i, j)$ to denote the maximum weight that can be obtained from at most j such subsets, subject to the condition that the vertex set of each subset must be fully contained in the vertex set of one of the cycles $C_1, \ldots, C_i$. After computing $F(r, n)$, the solution that produced this value is completed in an arbitrary way to a triangle packing TP_2.

The third solution is constructed by Algorithm A3 (see Fig. 5). It starts by deleting edges from $\mathscr{C}$ according to Procedure *Delete* described in Fig. 4. The result is a collection $\mathscr{P}$ of subpaths of $\mathscr{C}$ such that the following lemma holds (Figs. 4 and 5):

Lemma 1. *Consider a cycle $C_i \in \mathscr{C}$. Let E^i_{d} be the edge set deleted from C_i by Procedure Delete. Then*

1. $E^i_{\mathrm{d}} \neq \emptyset$.
2. *The edges in E^i_{d} are vertex disjoint.*

A1
input
1. A complete undirected graph $G = (V, E)$ *with weights* w_{ij} $(i, j) \in E$.
2. A cycle cover $\mathcal{C} = \{C_1, \ldots, C_r\}$.
returns *A triangle packing* TP_1.
begin
$TP_1 := \emptyset$.
for $i = 1, \ldots, r$:
  if $|C_i| = 3$
    then
      $TP_1 := TP_1 \cup \{C_i\}$.

  elseif $|C_i| = 5$
    then
      Let $e_1, \ldots, e_5$ *be the edges of* C_i *in cyclic order.*
      $\hat{w}_i := w_i + w_{i+1} + \frac{1}{2}w_{i+3}$ *[indices taken mod 2].*
      $\hat{w}_j = \max\{\hat{w}_i : i = 1, \ldots, 5\}$.
      $TP_1 := TP_1 \cup \{j, j+1, j+2\}$.
      $E' := E' \cup \{j+3, j+4\}$.

  elseif $|C_i| \notin \{3, 5\}$
    then
      Let $e_1, \ldots, e_c$ *be the edges of* C_i *in cyclic order.*
      Add to TP_1 *triangles induced by adjacent pairs of edges from* C_i
      of total weight at least $\frac{w(C_i)}{2}$.
      Add to V' *the vertices of* C_i *that are not incident to selected triangles.*
  end if
end for
$E'' := \left\lceil \frac{|E'|}{2} \right\rceil$ *heaviest edges from* E'.
for every $e \in E''$
    Add to TP_1 *triangles formed from* e *and a third vertex either from*
    $E' \setminus E''$ *or from* V'.
return TP_1.
end *A1*

Fig. 2. Algorithm A1.

3. *The expected size of* E^i_d *is at least* $\frac{1}{4}|C_i|$.
4. *The probability that any given vertex of* C_i *is adjacent to an edge in* E^i_d *is at least* $\frac{1}{2}$.

Proof.

1. The first property holds since an edge e_1 is deleted from each C_i.
2. The second property holds by the way edges are selected for deletion.
3. The expected value of $|E^i_d|$ is $\frac{1}{3}|C_i|$ for a triangle and $\frac{1}{4}|C_i|$ otherwise.
4. If a vertex belongs to a 3-cycle in $\mathcal{C}$ then it is incident to e_1 with probability $\frac{2}{3}$. If the vertex is in a k-cycle, $k > 3$, then since the expected size of E^i_d is $|C_i|/4$ and these edges are vertex disjoint, the expected number of vertices incident to these edges is $|C_i|/2$. □

Consider a given cycle $C \in \mathcal{C}$ with $|C| = 4k + l$. If C is a triangle then exactly one edge is deleted by Procedure *Delete*. If $l = 0$ then every fourth edge is deleted. If $l \in \{1, 2, 3\}$ and $|C| > 3$ then the number of deleted edges may be k or $k + 1$, and in each case, their location relative to e_1 is uniquely determined. Let us define a binary random variable X_C such that the number of deleted edges in C is $k + X_C$. X_C uniquely determines the *deletion pattern* in C, specifying the spaces among deleted edges, but not their specific location. (Fig. 7 illustrates the deletion patterns of small cycles, where dashed lines mark deleted edges.) Since every edge has equal probability to be chosen as e_1, the

A2
input
1. A complete undirected graph $G = (V, E)$ *with weights* w_{ij} $(i, j) \in E$.
2. A cycle cover $\mathcal{C} = \{C_1, \ldots, C_r\}$.
returns *A triangle packing* TP_2.
begin
$V_1, \ldots, V_r :=$ *the vertex sets of* $C_1, \ldots, C_r$, *respectively.*
for $j = 0, \ldots, n$:
$F(0, j) = 0$.
end for
$t := \left\lfloor \frac{1}{2}\left(1 + \frac{1}{\epsilon}\right) \right\rfloor$.
for $i = 1, \ldots, r$:
$F(i, 0) := 0$.
for $k = 1, \ldots, t$:
$W(i, k) :=$ *maximum weight of at most* k *disjoint 3-sets and 2-sets contained in* V_i.
end for
for $j = 1, \ldots n$
$F(i, j) := \max\{W(i, k) + F(i - 1, j - k) : k \leq \min\{j, t\}\}$.
end for
end for
$P(r, n) :=$ *a collection of 2-sets and 3-sets that gives* $F(r, n)$.
return TP_2, *a completion of* $P(r, n)$ *into a triangle packing.*
end *A2*

Fig. 3. Algorithm A2.

Delete
input *A set of cycles* $\mathcal{C} = \{C_1, \ldots, C_r\}$.
returns *A set of paths* $\mathcal{P}$.
begin
for $i = 1, \ldots, r$:
Randomly select an edge from C_i *and mark it* e_1.
Delete e_1.
Denote the edges of C_i *in cyclic order according to an arbitrary orientation and starting at* e_1 *by* $e_1, \ldots, e_c$, *where* $c = |C_i| = 4k + l$ *and* $l \in \{0, \ldots, 3\}$.
Delete from C_i *the edges* e_j *such that* $j = 1 \bmod 4$ *and* $j \leq c - 3$.
if $l = 1$
then
Delete e_{c-1} *with probability* $\frac{1}{4}$.

elseif $l = 2$
then
Delete e_{c-1} *with probability* $\frac{1}{2}$.

elseif $l = 3$ *and* $c > 3$
then
Delete e_{c-2} *with probability* $\frac{3}{4}$.
end if
end for
Denote the resulting path set by $\mathcal{P}$.
return $\mathcal{P}$.
end *Delete*

Fig. 4. Procedure *Delete*.

possible mappings of the deletion pattern into C have equal probabilities. Suppose that the deletion pattern is known. Every mapping of it into C specifies a subset S of vertices that are incident to deleted edges, and thus these vertices are the end vertices of the paths in $\mathcal{P}$. We call these vertices *free*.

A3

input

1. A complete undirected graph $G = (V, E)$, $|V| = 3n$, with weights w_{ij} $(i, j) \in E$.

2. A cycle cover $\mathcal{C} = \{C_1, \ldots, C_r\}$.

returns *A triangle packing TP_3.*

begin

Let E' be the edges of G with two ends in different cycles of $\mathcal{C}$.

Compute a maximum weight matching $M' \subset E'$.

$\mathcal{P} :=$ *Delete*$(\mathcal{C})$.

$M := \{(i, j) \in M' : i$ *and* j *are end vertices of paths in* $\mathcal{P}\}$.

% $M \cup \mathcal{P}$ *consists of paths* $P_1^*, \ldots, P_s^*$ *and cycles* $C_1^*, \ldots, C_t^*$ *such that each cycle contains at least two edges from* M.%

$\mathcal{P}^* := \{P_1^*, \ldots, P_s^*\}$.

begin cycle canceling step:

for $i = 1, \ldots, t$:

Randomly select an edge $e \in C_i^* \cap M$.

Add $C_i^* \setminus \{e\}$ to $\mathcal{P}^*$.

end for

end cycle canceling step

Arbitrarily complete $\mathcal{P}^*$ to a Hamiltonian cycle T.

Delete n edges from T to obtain a collection $\mathcal{P}_3$ of 2-paths of total weight at least $\frac{2}{3}w(T)$.

return TP_3, *a completion of $\mathcal{P}_3$ into a triangle packing.*

end *A3*

Fig. 5. Algorithm A3.

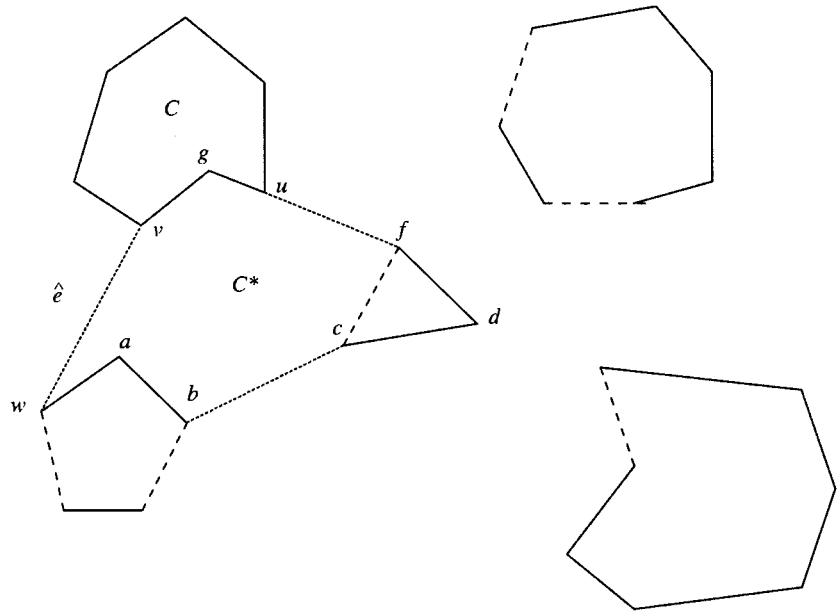

Fig. 6. A cycle in $M \cup \mathcal{P}$.

Algorithm A3 computes a maximum matching M' over the set $E' \subset E$ of edges with ends in distinct cycles. Only the subset $M \subseteq M'$ of edges whose two ends become free in the deletion procedure is useful, and it is used to generate paths with the undeleted edges from $\mathcal{C}$. However, cycles may also be generated this way, and a cycle canceling step further deletes edges to cancel these cycles. Fig. 6 illustrates this situation, where broken lines denote deleted edges and dotted lines denote edges that belong to M. The cycle C^* consists of the vertices $(v, w, a, b, c, d, f, u, g)$, and one of the edges (v, w), (b, c) and (f, u) that belong to M will be deleted to cancel C^*.

Consider an edge $\hat{e} = (v, w) \in M$. Denote by $\hat{\pi}$ the probability that $\hat{e}$ is deleted in the cycle canceling step, given that its vertices v and w are free (i.e., $v, w \in S$).

Lemma 2. $\hat{\pi} \leqslant \frac{1}{4}$.

Proof. Suppose that $v \in C$. For every $u \in C$, define $dist(u, v)$ to be the minimum (over the two possibilities) number of edges on a $u - v$ subpath of C. (For example, $dist(u, v) = 2$ in Fig. 6.)

In order to prove that $\hat{\pi} < \frac{1}{4}$, we define the following events:

$E_{\delta,x}$: $X_C = x$; $v, w \in S$; $\mathscr{P} \cup M$ contains a $u - v$ path P such that $\hat{e}$ is on P, $C \cap P = \{u, v\}$, $u \in S$, and $dist(u, v) = \delta$. (For example, $P = (v, w, a, b, c, d, f, u)$ and $\delta = 2$ in Fig. 6.)

E_C: u and v belong to a *common* path in $\mathscr{P}$.

E_D: u and v belong to *different* paths in $\mathscr{P}$.

If $\hat{e}$ belongs to a cycle $C^* \subseteq \mathscr{P} \cup M$, then in the event E_C, $|C^* \cap M| \geqslant 2$, whereas in the event E_D, $|C^* \cap M| \geqslant 4$. Therefore the respective probabilities that $\hat{e}$ is deleted in the cycle canceling step are at most $\frac{1}{2}$ and $\frac{1}{4}$. Using this observation,

$$\hat{\pi} \leqslant \sum_{\delta=1}^{\lfloor \frac{|C|}{2} \rfloor} \sum_{x=0}^{1} Pr(E_{\delta,x}) \left(\frac{1}{2} Pr(E_C | E_{\delta,x}) + \frac{1}{4} Pr(E_D | E_{\delta,x}) \right). \qquad \square$$

The proof is completed by Lemma 3.

Let $\pi(\delta, x) = \frac{1}{2} Pr(E_C | E_{\delta,x}) + \frac{1}{4} Pr(E_D | E_{\delta,x})$.

Lemma 3. *For every δ and x,*

$$\pi(\delta, x) \leqslant \frac{1}{4}.$$

Proof. For a cycle C with $c = 4k + l$ (where $l \in \{0, 1, 2, 3\}$), we consider every possible value δ of $dist(u, v)$ and $x \in \{0, 1\}$. We denote $p_C = Pr(E_C | E_{\delta,x})$, $p_D = Pr(E_D | E_{\delta,x})$, and $\pi = \pi(\delta, x)$. Note that to contradict the claim there must be vertices with $p_C > 0$. Otherwise, even if $p_D = 1$ for every vertex in S, we still have $\pi = \frac{1}{4}$. Therefore, our proof will check the occurrences of $p_C > 0$ and show that when they exist there are sufficient vertices with $p_D \leqslant \frac{1}{2}$ so that still $\pi \leqslant \frac{1}{4}$. Fig. 7 illustrates the analysis for small values of c. The numbers attached to the free vertices are the values of p_C, p_D given that the specified vertex is mapped to v. We skipped trivial cases where both probabilities are 0 for every free vertex. The analysis below applies to all cases except for three special cases where $c = 2\delta$, namely ($c = 6, x = 1$), ($c = 8$), and ($c = 10, x = 0$). These cases are analyzed in Fig. 7.

1. $\delta \geqslant 4$. In this case there is at most one pair of vertices with $(p_C, p_D) = (\frac{1}{2}, \frac{1}{2})$, and the other vertices have $p_C = 0$. On the other hand, if there is such a pair then there is also a pair of vertices with $(p_C, p_D) = (0, \frac{1}{2})$. Therefore, even if the rest of the vertices have $p_D = 1$ we still have $\pi \leqslant \frac{1}{4}$.
2. $l = 0$, or $l > 0$ and $x = 0$. In these cases, $p_C > 0$ is possible only with $d = 3$, in which case $(p_C, p_D) = (\frac{1}{2}, 0)$ for every vertex, except for at most one pair of vertices separated by more than three undeleted edges, where we get $(0, 0)$.
3. $l > 0$ and $\delta = 3$. This case is dominated (in terms of the value of π) by the case $l = 0$ and $\delta = 3$. The probabilities are identical except for that now there is also a pair of vertices with $p_C = 0$.
4. $l \in \{1, 2\}$, $x = 1$, and $\delta = 1$. In these cases there is a pair of vertices with $(p_C, p_D) = (\frac{1}{2}, \frac{1}{2})$, but all the others have $(p_C, p_D) = (0, \frac{1}{2})$. ($\pi$ is maximized when $c = 5$ and decreases for higher values.)
5. $l = 1, x = 1$ and $\delta = 2$. There is a pair with $(p_C, p_D) = (\frac{1}{2}, \frac{1}{2})$ and a pair with $(p_C, p_D) = (0, \frac{1}{2})$. The others have $(0, 0)$. (Again, π is maximized when $c = 5$ and decreases for higher values.)
6. $l = 2, x = 1$ and $\delta = 2$. In this case $p_C = 0$ for every free vertex. $\square$

Theorem 1. $\max\{w(TP_1), w(TP_2), w(TP_3)\} \geqslant \frac{89}{169}(1 - \varepsilon)opt$.

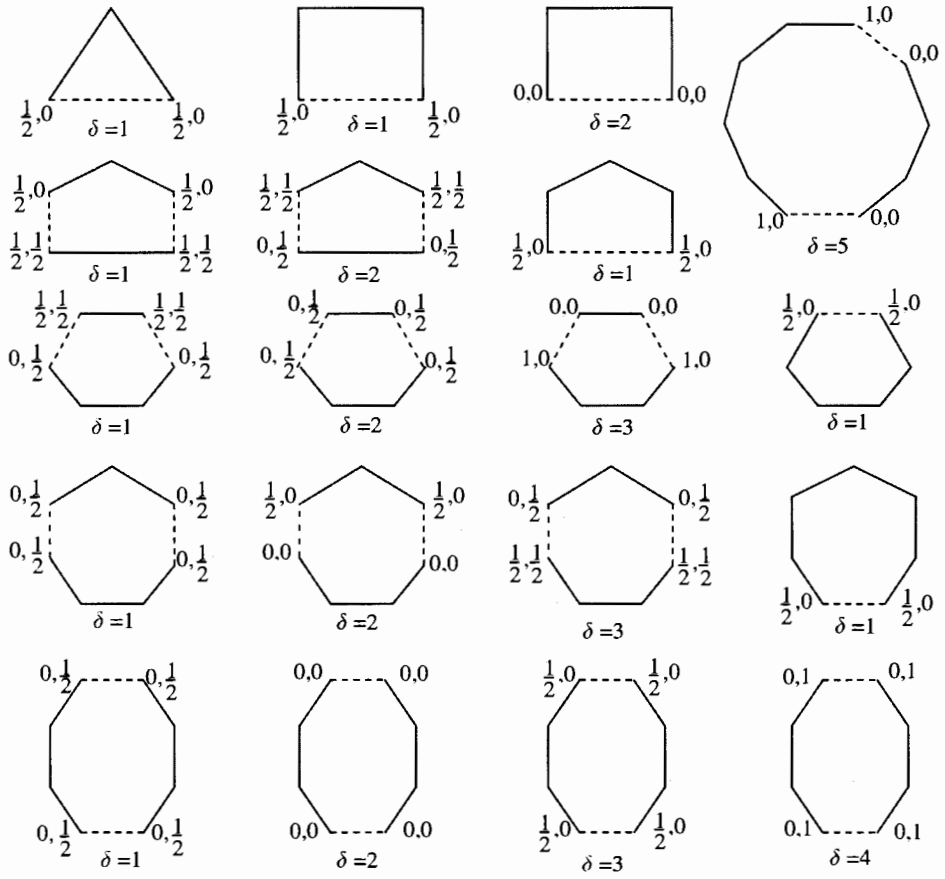

Fig. 7. Deletion patterns of small cycles.

Proof. Algorithm *WTP* first breaks long cycles loosing at most a fraction ε of their weight, and obtains a cycle cover $\mathscr{C}$. Thus, $w(\mathscr{C}) \geqslant (1-\varepsilon)opt$.

Let α denote the proportion of $w(\mathscr{C})$ contained in 3-edge cycles. The solution TP_1 contains all the triangles of $\mathscr{C}$ and at least half of the weight of any other cycle. Thus,

$$w(TP_1) \geqslant \left(\alpha + \frac{1-\alpha}{2}\right) w(\mathscr{C}) = \frac{1+\alpha}{2} w(\mathscr{C}) \geqslant \frac{1+\alpha}{2}(1-\varepsilon)opt. \tag{1}$$

Consider an optimal solution. Define T_{int} to be the edges of this solution whose end vertices are in the same connectivity component of $\mathscr{C}$, and suppose that $w(T_{\text{int}}) = \beta\, opt$. Then

$$w(TP_2) \geqslant w(T_{\text{int}}) = \beta\, opt. \tag{2}$$

Algorithm A3 computes a Hamiltonian path T, and then constructs a triangle packing of weight at least $\frac{2}{3}w(T)$. T is built as follows: First $\frac{1}{3}$ of the weight of any triangle, and $\frac{1}{4}$ of any other cycle of $\mathscr{C}$ is deleted, leaving weight of $[\frac{2}{3}\alpha + \frac{3}{4}(1-\alpha)]w(\mathscr{C})$. Then edges from the matching M' are added. Originally $w(M') \geqslant \frac{(1-\beta)}{2}\, opt$. However only edges with two free ends are used, and by Lemma 1(4) their expected weight is at least $\frac{1}{4}w(M') \geqslant \frac{(1-\beta)}{8}\, opt$. Then cycles are broken, deleting at most $\frac{1}{4}$ of the remaining weight (by Lemma 2). Hence the added weight is at least $\frac{3}{32}(1-\beta)$. Altogether,

$$w(T) \geqslant [\tfrac{2}{3}\alpha + \tfrac{3}{4}(1-\alpha)]w(\mathscr{C}) + \tfrac{3}{32}(1-\beta)opt \geqslant [\tfrac{2}{3}\alpha + \tfrac{3}{4}(1-\alpha)](1-\varepsilon)opt + \tfrac{3}{32}(1-\beta)opt.$$

From this, a solution is formed after deleting at most $\frac{1}{3}$ of the weight. Hence,

$$w(TP_3) \geqslant \left(\tfrac{9}{16} - \tfrac{1}{18}\alpha - \tfrac{1}{16}\beta\right)(1-\varepsilon)opt. \tag{3}$$

It is now easy to prove that $\max\{w(TP_1), w(TP_2), w(TP_3)\} \geqslant \frac{89}{169}(1-\varepsilon)opt$: If $\alpha > \frac{89}{169}$ then $w(TP_1) > \frac{89}{169}(1-\varepsilon)opt$, if $\beta > \frac{89}{169}$ then $w(TP_2) > \frac{89}{169}(1-\varepsilon)opt$, and if neither of these conditions holds then $w(TP_3) > \frac{89}{169}(1-\varepsilon)opt$. □

The time consuming parts of the algorithm are the computation of a maximum 2-matching and the dynamic program in Algorithm A2. The first can be computed in time $O(n^3)$ as in Ref. [9]. Since the latter is executed on cycles with at most ε^{-1} edges each, it also takes $O(n^3)$ for any constant ε.

4. 2-path packing

Consider now the MAXIMUM 2-PATH PACKING PROBLEM. We apply Algorithm *WTP*, with two slight changes: one is that we do not complete 2-paths into triangles, and the other is that in Algorithm A1 we select from every triangle of $\mathscr{C}$ the two heaviest edges to form a 2-path. The analysis is identical, except for that the bound guaranteed by Algorithm A1 is only $w(TP_1) \geqslant [\frac{2}{3}\alpha + \frac{1}{2}(1-\alpha)]w(\mathscr{C})$. The resulting approximation bound is $\frac{35}{67} - \varepsilon$.

References

[1] E. Arkin, R. Hassin, On local search for weighted packing problems, Math. Oper. Res. 23 (1998) 640–648.
[2] V. Bafna, B. Narayanan, R. Ravi, Nonoverlapping local alignments (weighted independent sets of axis-parallel rectangles), Discrete Appl. Math. 71 (1996) 41–53.
[3] P. Berman, A $d/2$ approximation for maximum weight independent set in d-claw free graphs, Nordic J. Comput. 7 (2000) 178–184.
[4] B. Chandra, M.M. Halldórsson, Greedy local improvement and weighted set packing approximation, J. Algorithms 39 (2001) 223–240.
[5] M. Chlebík, J. Chlebíková, Approximating hardness for small occurrence instances of NP-hard problems, in: R. Petreschi, G. Parsiano, R. Silvestri (Eds.), Algorithms and Complexity, CIAC 2003, Lecture Notes in Computer Science, vol. 2653, pp. 152–164.
[6] T. Feder, C. Subi, Partition into k-vertex subgraphs of k-partite graphs, manuscript, 2004.
[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1978.
[8] M.M. Halldórsson, Approximating discrete collections via local improvements, in: Proceedings of the Sixth Annual ACM–SIAM Symposium on Discrete Algorithms SODA'95, 1995, pp. 160–169.
[9] D. Hartvigsen, Extensions of matching theory, Ph.D. Thesis, Carnegie-Mellon University, 1984.
[10] R. Hassin, S. Rubinstein, An approximation algorithm for maximum packing of 3-edge paths, Inform. Process. Lett. 63 (1997) 63–67.
[11] R. Hassin, S. Rubinstein, Better approximations for Max TSP, Inform. Process. Lett. 75 (2000) 181–186.
[12] C.A.J. Hurkens, A. Schrijver, On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems, SIAM J. Discrete Math. 2 (1989) 68–72.
[13] V. Kann, Maximum bounded 3-dimensional matching is MAX SNP-complete, Inform. Process. Lett. 37 (1991) 27–35.
[14] J.F. Pekny, D.L. Miller, A staged primal-dual algorithm for finding a minimum cost perfect two-matching in an undirected graph, ORSA J. Comput. 6 (1994) 68–81.

Available online at www.sciencedirect.com
ScienceDirect
Discrete Applied Mathematics 154 (2006) 2620
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Erratum

Erratum to "An approximation algorithm for maximum triangle packing" [Discrete Applied Mathematics 154 (2006) 971–979]

Refael Hassin, Shlomi Rubinstein

Department of Statistics and Operations Research, Tel Aviv University, Tel Aviv, 69978, Israel

Received 9 May 2006; accepted 9 May 2006

We thank Zhi-Zhong Chen (Tokyo Denki University) for drawing our attention to the following typos in our paper.

Lemma 2. (i) In the definition of $E_{\delta,x}$ the term $u \in S$ should be deleted. This term should be added instead to the definitions of E_C and E_D. (ii) The numbers attached to the upper vertices of the first graph in the second row of Fig. 7 should be $(0, \frac{1}{2})$ (instead of $(\frac{1}{2}, 0)$).

To facilitate understanding of the proof of Lemma 2 we would like to add the following clarifications: (i) in the stated upper bound on $\hat{\pi}$, the conditional probabilities are over all possible mappings of the deletion pattern to the cycle with the specified δ and x. (ii) E_C and E_D are the events that u and v are ends of, respectively, a common or a different path in $\mathscr{P}$. Since it is possible that $u \notin S$, the sum of probabilities of these events may be less than 1. (iii) The values attached to vertices in Fig. 7 are the values of P_C and P_D given that the specified vertex is mapped to v. These values are taken over the (generally) two possible cases for the location of u on the cycle, given that it is at distance δ from v.

Theorem 1. The correct bound is 43/83. (Note that the bound for the 2-edge path packing does not change.) Some numbers appearing in the proof of Theorem 1 should be changed as follows:

Originally $w(M') \geqslant (1-\beta)/3opt$.

Hence $\frac{1}{4}w(M') \geq (1-\beta)/12opt$.

Hence the added weight is at least $\frac{1}{16}(1-\beta)$;

Altogether,

$$w(T) \geq \left[\tfrac{2}{3}\alpha + \tfrac{3}{4}(1-\alpha)\right](1-\varepsilon)opt + \tfrac{1}{16}(1-\beta)opt.$$

Hence,

$$w(TP_3) \geq \left(\tfrac{13}{24} - \tfrac{1}{18}\alpha - \tfrac{1}{24}\beta\right)(1-\varepsilon)opt. \tag{1}$$

It follows that $\max\{w(TP_1), w(TP_2), w(TP_3)\} \geqslant \frac{43}{83}(1-\varepsilon)opt$: if $\alpha > \frac{3}{83}$ then $w(TP_1) > \frac{43}{83}(1-\varepsilon)opt$, if $\beta > \frac{43}{83}$ then $w(TP_2) > \frac{43}{83}(1-\varepsilon)opt$, and if neither of these conditions holds then $w(TP_3) > \frac{43}{83}(1-\varepsilon)opt$.

DOI of original article: 10.1016/j.dam.2005.11.003.

E-mail addresses: hassin@post.tau.ac.il (R. Hassin), shlomiru@post.tau.ac.il (S. Rubinstein).

doi:10.1016/j.dam.2006.05.005