

# New Exact and Approximation Algorithms for the Star Packing Problem in Undirected Graphs

Maxim Babenko<sup>1</sup> and Alexey Gusakov<sup>2</sup>

1 Moscow State University, Yandex  
maxim.babenko@gmail.com

2 Moscow State University, Google  
agusakov@gmail.com

---

## Abstract

By a  $T$ -star we mean a complete bipartite graph  $K_{1,t}$  for some  $t \leq T$ . For an undirected graph  $G$ , a  $T$ -star packing is a collection of node-disjoint  $T$ -stars in  $G$ . For example, we get ordinary matchings for  $T = 1$  and packings of paths of length 1 and 2 for  $T = 2$ . Hereinafter we assume that  $T \geq 2$ .

Hell and Kirkpatrick devised an ad-hoc augmenting algorithm that finds a  $T$ -star packing covering the maximum number of nodes. The latter algorithm also yields a min-max formula.

We show that  $T$ -star packings are reducible to network flows, hence the above problem is solvable in  $O(m\sqrt{n})$  time (hereinafter  $n$  denotes the number of nodes in  $G$ , and  $m$  — the number of edges).

For the edge-weighted case (in which weights may be assumed positive) finding a maximum  $T$ -packing is NP-hard. A novel  $\frac{9}{4} \frac{T}{T+1}$ -factor approximation algorithm is presented.

For non-negative node weights the problem reduces to a special case of a max-cost flow. We develop a divide-and-conquer approach that solves it in  $O(m\sqrt{n} \log n)$  time. The node-weighted problem with arbitrary weights is more difficult. We prove that it is NP-hard for  $T \geq 3$  and is solvable in strongly-polynomial time for  $T = 2$ .

**1998 ACM Subject Classification** G.2.2 Graph algorithms; G.1.2 Minimax approximation and algorithms

**Keywords and phrases** graph algorithms, approximation algorithms, generalized matchings, flows, weighted packings.

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2011.519

## 1 Introduction

### 1.1 Preliminaries

Recall the classical *maximum matching problem*: given an undirected graph  $G$  the goal is to find a collection  $M$  (called a *matching*) of node-disjoint edges covering as many nodes as possible. Motivated by this definition, one may consider an arbitrary (possibly infinite) collection of undirected graphs  $\mathcal{G}$ , called *allowed*, and ask for a collection  $\mathcal{M}$  of node-disjoint subgraphs of  $G$  (not necessarily spanning) such that every member of  $\mathcal{M}$  is isomorphic to some graph in  $\mathcal{G}$ . Let the *size* of  $\mathcal{M}$  be the total number of nodes covered by the elements of  $\mathcal{M}$ . The generalized matching problem [8] asks for a  $\mathcal{G}$ -matching of maximum size.

Clearly, the tractability of the generalized problem depends solely on the choice of  $\mathcal{G}$ . The case when all graphs in  $\mathcal{G}$  are bipartite was investigated by Hell and Kirkpatrick [8]. Roughly speaking, in this case the maximum  $\mathcal{G}$ -matching problem is NP-hard unless  $\mathcal{G} =$



© Maxim Babenko, Alexey Gusakov;  
licensed under Creative Commons License NC-ND  
28th Symposium on Theoretical Aspects of Computer Science (STACS'11).  
Editors: Thomas Schwentick, Christoph Dürr; pp. 519–530



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



SYMPOSIUM  
ON THEORETICAL  
ASPECTS  
OF COMPUTER  
SCIENCE

$\{K_{1,1}, \dots, K_{1,T}\}$  for some  $T \geq 1$ . (For a precise statement, see [8, Sec. 4].) This is exactly the case we study throughout the paper.

► **Definition 1.** A  $T$ -star is a graph  $K_{1,t}$  for some  $1 \leq t \leq T$ . For an undirected graph  $G$ , a  $T$ -star packing in  $G$  is a collection of node-disjoint subgraphs in  $G$  (not necessary spanning) that are isomorphic to some  $T$ -stars.

Since 1-star packings are just ordinary matchings and are already extensively studied (see, e.g., [14]), we restrict our attention to the case  $T \geq 2$ .

The max-size  $T$ -star packing problem was addressed in [13, 1, 8] and others. An  $O(mn)$ -time ad-hoc augmenting path algorithm (hereinafter  $n := |VG|$ ,  $m := |EG|$ ) and a min-max formula are known. In [8] it is noted that a faster  $O(m\sqrt{n})$ -time algorithm can be derived using the blocking augmentation strategy (see [2, 9]), but we are not aware of any publicly available exposition. A more restrictive variant of the problem, where the stars are required to be node-induced subgraphs, is presented in [12]. An extension to node capacities is given in [15].

## 1.2 Our Contribution

This paper presents an alternative treatment of  $T$ -star packings that is based on network flows. In Section 2 we show how the max-size  $T$ -star packing problem reduces to finding a max-value flow in a digraph with  $O(n)$  nodes and  $O(m)$  arcs. This immediately implies an  $O(m\sqrt{n})$ -time algorithm for the max-size  $T$ -star packing problem.

The above reduction serves two purposes. Firstly, it mitigates the need for ad-hoc tricks and fits star packings into a widely studied field of network flows. Secondly, this reduction provides interesting opportunities for attacking other optimization problems that are related to  $T$ -star packings.

Let  $G$  be an edge-weighted graph and the goal is to find a  $T$ -star packing such that the sum of weights of edges belonging to stars is maximum. This problem is NP-hard and in Section 3 we present a  $\frac{9}{4} \frac{T}{T+1}$ -factor approximation algorithm, which is based on max-cost flows.

Finally let  $G$  be a node-weighted graph and the objective function is the sum of weights of nodes covered by stars. This case is studied in Section 4. For non-negative weights, a divide-and-conquer approach yields a nice  $O(m\sqrt{n} \log n)$ -time algorithm. For general weights, the complexity of the resulting problem depends on  $T$ . For  $T = 2$ , we give a strongly-polynomial algorithm that employs bidirected network flows. If  $T \geq 3$ , the problem is NP-hard.

## 2 Reduction to Network Flows

### 2.1 Auxiliary Digraphs

In this section we explain the core of our approach that relates star packings to network flows. We employ some standard graph-theoretic notation throughout the paper. For an undirected graph  $G$  we denote its sets of nodes and edges by  $VG$  and  $EG$ , respectively. For a directed graph we speak of arcs rather than edges and denote the arc set of  $G$  by  $AG$ . A similar notation is used for paths, trees, and etc.

For  $U \subseteq VG$ , the set of arcs entering (respectively leaving)  $U$  is denoted by  $\delta_G^{\text{in}}(U)$  and  $\delta_G^{\text{out}}(U)$ . Also,  $\gamma_G(U)$  denotes the set of arcs (or edges) with both endpoints in  $U$  and  $G[U]$  denotes the subgraph of  $G$  induced by  $U$ , i.e.  $G[U] = (U, \gamma_G(U))$ . When the (di-)graph is

clear from the context, it is omitted from notation. Also for a function  $\varphi: U \rightarrow \mathbb{R}$  and a subset  $U' \subseteq U$ , let  $\varphi(U')$  denote  $\sum_{u \in U'} \varphi(u)$ .

Let, as earlier,  $G$  be an undirected graph and  $T \geq 2$  be an integer. Replace each edge in  $G$  by a pair of oppositely directed arcs and denote the resulting digraph by  $\vec{G}$ . The following definition is crucial:

► **Definition 2.** A subset of arcs  $F \subseteq A\vec{G}$  is called *T-feasible* if for each node  $v \in VG$  at most  $T$  arcs in  $F$  leave  $v$  and at most one arc in  $F$  enters  $v$ .

The above  $T$ -feasible arc sets are equivalent to  $T$ -star packings in the following sense:

► **Theorem 3.** *The maximum size of a T-feasible arc set in  $G$  is equal to the maximum size of a T-star packing. Moreover, given a T-feasible arc set  $F$  one can turn it in linear time into a T-star packing of size at least  $|F|$ .*

Before presenting the proof of Theorem 3, let us explain how a max-size  $T$ -feasible arc set size can be found. To this aim, split each node  $v \in V\vec{G}$  into two copies, say  $v^1$  and  $v^2$ . Each arc  $(u, v) \in A\vec{G}$  is transformed into an arc  $(u^1, v^2)$ . Two auxiliary nodes are added: a *source*  $s$  that is connected to every node  $v^1$ ,  $v \in V\vec{G}$ , by arcs  $(s, v^1)$ , and a *sink*  $t$  that is connected to every node  $v^2$ ,  $v \in V\vec{G}$ , by arcs  $(v^2, t)$ . We endow each arc  $(s, v^1)$  with capacity equal to  $T$ , each arc  $(v^2, t)$  with unit capacity, and the remaining arcs with infinite capacities. The resulting digraph is denoted by  $H$ .

We briefly remind the basic terminology and notation on network flows (see, e.g., [5, 18] and [16, Ch. 10]). Let  $\Gamma$  be a digraph with a distinguished source node  $s$  and a sink node  $t$ . The nodes in  $V\Gamma - \{s, t\}$  are called *inner*. Let  $u: A\Gamma \rightarrow \mathbb{Z}_+$  be integer arc capacities.

► **Definition 4.** An integer *u-feasible flow* (or just *feasible flow* if capacities are clear from the context) is a function  $f: A\Gamma \rightarrow \mathbb{Z}_+$  such that: (i)  $f(a) \leq u(a)$  for each  $a \in A\Gamma$ ; and (ii)  $\text{div}_f(v) = 0$  for each inner node  $v$ .

Here  $\text{div}_f(v) := f(\delta^{\text{out}}(v)) - f(\delta^{\text{in}}(v))$  denotes the *divergence* of  $f$  at  $v$ . The *value* of  $f$  is  $\text{val}(f) := \text{div}_f(s)$ . A max-value feasible integer flow can be found in strongly polynomial time (see [18] and [16, Ch. 10]).

Let  $f$  is a feasible integer flow in  $H$  (regarded as a network with a source  $s$ , a sink  $t$ , and capacities  $u$ ). Then  $f(u^1, v^2) \in \{0, 1\}$  for each  $(u, v) \in A\vec{G}$ , since at most one unit of flow may leave  $v^2$ . (Hereinafter we abbreviate  $f((u, v))$  to  $f(u, v)$ .) Define

$$F := \left\{ (u, v) \in A\vec{G} \mid f(u^1, v^2) = 1 \right\}.$$

Then the  $u$ -feasibility of  $f$  implies the  $T$ -feasibility of  $F$ . Moreover, this correspondence between  $u$ -feasible integer flows  $f$  and  $T$ -feasible arc sets  $F$  is one-to-one.

The augmenting path algorithm of Ford and Fulkerson [5] computes a max-value flow in  $H$  in  $O(mn)$  time. Applying blocking augmentations [9, 2], the latter bound can be improved to  $O(m\sqrt{n})$ . (In fact for networks of the above “bipartite” type, one can prove the bound of  $O(m\sqrt{\Delta})$ . Here  $\Delta := \min(\Delta_s, \Delta_t)$ ,  $\Delta_s$  is the sum of capacities of arcs leaving  $s$ , and  $\Delta_t$  is the sum of capacities of arcs entering  $t$ .)

Therefore by Theorem 3, a maximum  $T$ -star packing can be found in  $O(m\sqrt{n})$  time. (The *clique compression technique* [4] implies a somewhat better time bound; however, the speedup is only sublogarithmic.)

## 2.2 Proof of Theorem 3

The proof consists of two parts. For the easy one, let  $\mathcal{P}$  be a  $T$ -star packing in  $G$ . To construct a  $T$ -feasible arc set  $F$ , take every star  $S \in \mathcal{P}$ . Let  $v$  be its *central* node (i.e. a node of maximum degree) and  $u_1, \dots, u_t$  be its *leaves* (i.e. the remaining nodes). For  $S = K_{1,1}$  the notion of a central node is ambiguous but any choice will do. Add arcs  $(v, u_1), \dots, (v, u_t)$  and also  $(u_1, v)$  to  $F$ . Clearly  $F$  is  $T$ -feasible and its size coincides with the number of nodes covered by  $\mathcal{P}$ .

The reverse reduction is more involved. Consider a  $T$ -feasible arc set  $F$ . Then  $F$  decomposes into a collection of node-disjoint weakly connected components. We deal with each of these components separately and construct a  $T$ -star packing  $\mathcal{P}$  of size at least  $|F|$ . Let  $Q$  be one of the above components. One can easily see that two cases are possible:

**Case I:**  $Q$  forms a directed out-tree  $\mathcal{T}$  where each node has at most  $T$  children and the arcs are directed towards leaves. The following pruning is applied iteratively to  $\mathcal{T}$ . Pick an arbitrary leaf  $u_1$  in  $\mathcal{T}$  of maximum depth, let  $v$  be the parent of  $u_1$  and  $u_2, \dots, u_t$  be the siblings of  $u_1$ . Clearly  $t \leq T$ . Remove nodes  $v, u_1, \dots, u_t$  together with incident arcs from  $\mathcal{T}$  and add to  $\mathcal{P}$  a copy of  $K_{1,t}$ , where  $v$  is its center and  $u_1, \dots, u_t$  are the leaves. Repeat the process until  $\mathcal{T}$  is empty or consists of a single node (the root  $r$ ). Each time a star covering  $t + 1$  nodes is added to  $\mathcal{P}$ , either  $t + 1$  (if  $u \neq r$ ) or  $t$  (if  $u = r$ ) arcs are removed from  $\mathcal{T}$ . At the end one gets a  $T$ -star packing of size at least  $|AQ|$  nodes, as required.

**Case II:**  $Q$  consists of a directed cycle  $\Omega$  and a number (possibly zero) of directed out-trees attached to it (see Fig. 1(a) for an example). Let  $g_0, \dots, g_{l-1}$  be the nodes of  $\Omega$  (in the order of their appearance on the cycle). For  $i = 0, \dots, l - 1$ , let  $\mathcal{T}_i$  be the directed out-tree rooted at  $g_i$  in  $Q$ . (If no tree is attached to  $g_i$ , then we regard  $\mathcal{T}_i$  as consisting solely of its root node  $g_i$ .) Each node in the latter trees has at most  $T$  children, and the roots of these trees have at most  $T - 1$  children. We process the trees  $\mathcal{T}_0, \dots, \mathcal{T}_{l-1}$  like in Case I and obtain a partial packing  $\mathcal{P}$ . Our final task is to modify  $\mathcal{P}$  to satisfy the following condition: each node  $v \in VQ$  that has an incoming arc in  $F$  is covered by a star in  $\mathcal{P}$ . So far, the above condition is only violated for nodes in  $\Omega$  that are not covered by  $\mathcal{P}$ .

Two subcases are possible. First, suppose that all nodes of  $\Omega$  are not covered. Then one can cover  $\Omega$  by a collection of node-disjoint (and also disjoint from  $\mathcal{P}$ ) paths of lengths 1 and 2. Adding these paths to  $\mathcal{P}$  finishes the job. (Note that this is exactly where we use the condition  $T \geq 2$ .)

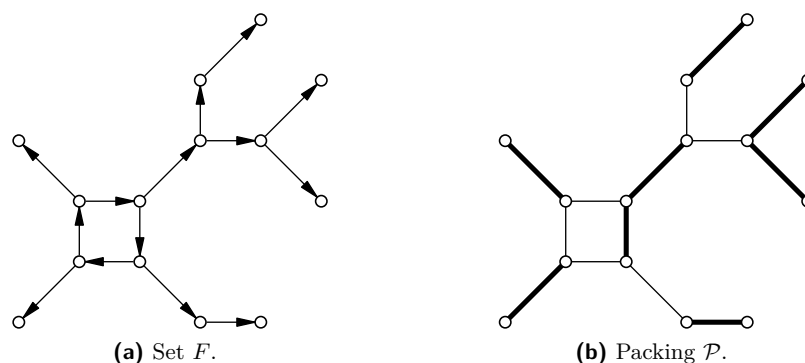
Second, suppose that  $\Omega$  contains both covered and not covered nodes. Let  $g_i, \dots, g_j$  be a maximal consecutive segment of uncovered nodes, i.e.  $g_{i-1}$  and  $g_{j+1}$  are covered (indices are taken modulo  $l$ ). If  $j - i$  is odd, then adding  $(j - i + 1)/2$  disjoint copies of  $K_{1,1}$  covering  $g_i, \dots, g_j$  completes the proof. Otherwise let  $j - i$  be even. Recall that  $g_{i-1}$  is covered by some star  $S \in \mathcal{P}$  and  $g_{i-1}$  is its central node. Since the degree of  $g_{i-1}$  in  $S$  is at most  $T - 1$ , one can augment  $S$  by adding a new leaf  $g_i$ . This way  $g_i$  gets covered and the case reduces to the previous one. An example is depicted in Fig. 1(b).

Clearly  $F$  can be converted into  $\mathcal{P}$  in linear time. ◀

## 3 Edge-Weighted Packings

### 3.1 Hardness

Consider arbitrary *edge weights*  $w: EG \rightarrow \mathbb{Q}$  and let the *edge weight*  $w(S)$  of a star  $S$  be the sum of weights of its edges. In this section we focus on finding a  $T$ -star packing  $\mathcal{P}$  that maximizes  $w(\mathcal{P}) := \sum_{S \in \mathcal{P}} w(S)$ . Allowing negative edge weights is redundant since such



■ **Figure 1** Transforming  $F$  into  $\mathcal{P}$  ( $T = 2$ ).

edges may be removed from  $G$  without changing the optimum. Therefore we assume that edge weights are non-negative.

► **Theorem 5.** *The problem of deciding, for given  $G$ ,  $T$ ,  $w$ , and  $\lambda \in \mathbb{Q}_+$ , if  $G$  contains a  $T$ -star packing of edge weight at least  $\lambda$ , is NP-hard even in the all-unit weight case.*

**Proof.** It is known (see, e.g. [8]) that deciding if  $G$  admits a perfect (i.e. covering all the nodes)  $\mathcal{G}$ -matching is NP-hard for  $\mathcal{G} = \{K_{1,T}\}$ . We reduce the latter to the edge-weighted  $T$ -star packing problem as follows. If  $|VG|$  is not divisible by  $|T| + 1$ , then the answer is negative. Otherwise set  $w(e) := 1$  for all  $e \in EG$ . A  $T$ -star packing  $\mathcal{P}$  obeys  $w(\mathcal{P}) = \frac{nT}{T+1}$  if and only if all stars in  $\mathcal{P}$  are isomorphic to  $K_{1,T}$ . Hence solving the edge-weighted  $T$ -star packing problem enables to check if  $G$  has a perfect  $\mathcal{G}$ -matching. ◀

## 3.2 Approximation

We show how to compute, in strongly-polynomial time, a  $T$ -star packing  $\mathcal{P}$  such that  $w(\mathcal{P}) \geq \text{OPT} \cdot \frac{4}{9} \frac{T+1}{T}$ , where  $\text{OPT}$  denotes the maximum weight of a  $T$ -star packing in  $G$ . Let us extend the weights from  $G$  to  $\vec{G}$ , i.e. define  $w(u, v) := w(v, u) := w(e)$  for  $e = \{u, v\} \in EG$ . Let  $\text{OPT}'$  be the maximum weight of a  $T$ -feasible arc set in  $\vec{G}$ .

► **Lemma 6.**  $\text{OPT}' \geq \text{OPT} \cdot \frac{T+1}{T}$ .

**Proof.** Fix a max-weight packing of  $T$ -stars  $\mathcal{P}_{\text{OPT}}$ . Consider a star  $S \in \mathcal{P}_{\text{OPT}}$ , and let  $e_1 = \{u, v_1\}, \dots, e_t = \{u, v_t\}$  be the edges forming  $S$  ( $t \leq T$ ). We may assume that  $e_1$  is a maximum-weight edge (among  $e_1, \dots, e_t$ ).

Consider the arc set  $\{(u, v_1), (v_1, u), (u, v_2), (u, v_3), \dots, (u, v_t)\}$  (i.e.  $e_1$  generates a pair of opposite arcs while the other edges — just a single one). Taking the union of all these arc sets one gets a  $T$ -feasible arc set  $F$  obeying  $w(F) \geq \sum_{S \in \mathcal{P}} \frac{T+1}{T} w(S) = \text{OPT} \cdot \frac{T+1}{T}$ , as claimed. ◀

Applying the correspondence between feasible integer flows in  $H$  and  $T$ -feasible arc sets and regarding arc weights as costs, a max-weight  $T$ -feasible arc set  $F$  can be found by a max-cost flow algorithm in strongly-polynomial time, see [18, Sec. 8.4]. (For arc costs  $c: AH \rightarrow \mathbb{Q}$  and a flow  $f$  in  $H$ , the *cost* of  $f$  is  $c(f) := \sum_a c(a)f(a)$ .)

We turn  $F$  into a  $T$ -star packing  $\mathcal{P}$  obeying  $w(\mathcal{P}) \geq \frac{4}{9} w(F)$  as follows. Consider the weakly-connected components of  $F$  and perform a case splitting similar to that in the proof

of Theorem 3. For each component  $Q$ , we extract a  $T$ -star packing  $\mathcal{P}_Q$  covering some nodes of  $Q$  such that  $w(\mathcal{P}_Q) \geq \frac{4}{9}w(Q)$  and then take the union  $\mathcal{P} := \bigcup_Q \mathcal{P}_Q$ .

**Case I:**  $Q$  is a directed out-tree  $\mathcal{T}$  rooted at a node  $r$ . Call an arc  $(u, v)$  in  $\mathcal{T}$  *even* (respectively *odd*) if the length of the  $r$ - $u$  path in  $\mathcal{T}$  is even (respectively odd). Let  $E^0$  (respectively  $E^1$ ) denote the set of edges (in  $G$ ) corresponding to even (respectively odd) arcs of  $\mathcal{T}$ . Sets  $E^0$  and  $E^1$  generate  $T$ -star packings  $\mathcal{P}^0$  and  $\mathcal{P}^1$  in  $G$ . Choose from these a packing with the largest weight and denote it by  $\mathcal{P}_Q$ . Then  $w(\mathcal{P}_Q) \geq \frac{1}{2}(w(\mathcal{P}^0) + w(\mathcal{P}^1)) = \frac{1}{2}w(Q) \geq \frac{4}{9}w(Q)$ .

**Case II:**  $Q$  is a directed cycle  $\Omega$  with a number of out-trees attached to it. Let  $g_0, \dots, g_{l-1}$  be the nodes of  $\Omega$  (numbered in the order of their appearance) and  $\mathcal{T}_0, \dots, \mathcal{T}_{l-1}$  be the corresponding trees ( $\mathcal{T}_i$  is rooted at  $g_i$ ,  $i = 0, \dots, l-1$ ).

**Subcase II.1:**  $l$  is even. Choose an arbitrary node  $r$  on  $\Omega$  and label the arcs of  $Q$  as *even* and *odd* as in Case I. (Note that for any node  $v$  in  $Q$ , there is a unique simple  $r$ - $v$  path in  $Q$ .) This way, a  $T$ -star packing  $\mathcal{P}_Q$  obeying  $w(\mathcal{P}_Q) \geq \frac{1}{2}w(Q) \geq \frac{4}{9}w(Q)$  is constructed.

**Subcase II.2:**  $l$  is odd. We construct a collection of  $3l$  packings (each covering a subset of nodes of  $Q$ ) of total weight at least  $\frac{3l-1}{2}w(Q)$ . To this aim, label the arcs of  $\mathcal{T}_0, \dots, \mathcal{T}_{l-1}$  as *even* and *odd* like in Case I (starting from their roots). For  $i = 0, \dots, l-1$ , let  $E_i^0$  (respectively  $E_i^1$ ) be the set of edges (in  $G$ ) corresponding to even (respectively odd) arcs of  $\mathcal{T}_i$ . Also let  $e_i = \{g_i, g_{i+1}\}$  be the  $i$ -th edge of  $\Omega$  (hereinafter indices are taken modulo  $l$ ). Consider the (edge sets of the) following  $l$  packings (taking  $i = 0, \dots, l-1$ ):

$$\{e_i, e_{i+1}\} \cup \{e_{i+3}, e_{i+5}, \dots, e_{i+l-2}\} \cup \\ (E_i^1 \cup E_{i+1}^1 \cup E_{i+2}^1) \cup (E_{i+3}^0 \cup E_{i+4}^1) \cup (E_{i+5}^0 \cup E_{i+6}^1) \cup \dots \cup (E_{i+l-2}^0 \cup E_{i+l-1}^1).$$

Also consider the (edge sets of the) following  $2l$  packings (taking each value  $i = 0, \dots, l-1$  twice):

$$\{e_{i+1}, e_{i+3}, e_{i+5}, \dots, e_{i+l-2}\} \cup \\ E_i^0 \cup (E_{i+1}^0 \cup E_{i+2}^1) \cup (E_{i+3}^0 \cup E_{i+4}^1) \cup \dots \cup (E_{i+l-2}^0 \cup E_{i+l-1}^1).$$

By a straightforward calculation, one can see that the total weight of these  $3l$  packings is

$$\frac{3l-1}{2} \sum_{i=0}^l w(e_i) + \frac{3l-1}{2} \sum_{i=0}^l w(E_i^0) + \frac{3l+1}{2} \sum_{i=0}^l w(E_i^1) \geq \\ \frac{3l-1}{2} \left( \sum_{i=0}^l w(e_i) + \sum_{i=0}^l w(E_i^0) + \sum_{i=0}^l w(E_i^1) \right) = \frac{3l-1}{2} w(Q).$$

Choosing a max-weight packing  $\mathcal{P}_Q$  among these  $3l$  instances, one gets  $w(\mathcal{P}_Q) \geq \frac{1}{3l} \cdot \frac{3l-1}{2} w(Q) \geq \frac{4}{9}w(Q)$  (since  $l \geq 3$ ), as claimed.

The above postprocessing converting  $F$  into  $\mathcal{P}$  can be done in strongly-polynomial time. Together with Lemma 6 this proves the following:

► **Theorem 7.** A  $\frac{9}{4} \frac{T}{T+1}$ -factor approximation to the edge-weighted  $T$ -star packing problem can be found in strongly polynomial time.

## 4 Node-Weighted Packings

### 4.1 General Weights

Now consider a node-weighted counterpart of the problem. Let  $w: VG \rightarrow \mathbb{Q}$  be node weights, and let the *weight* of a  $T$ -star packing  $\mathcal{P}$  be the sum of weights of nodes covered by  $\mathcal{P}$ .

Now one cannot freely assume that weights are non-negative. Indeed, removing a node with a negative weight may change the optimum (consider  $G = K_{1,T}$ , where the weight of the central node is negative while the weights of the others are positive). In fact, for  $T \geq 3$  and arbitrary  $w$ , we get an NP-hard problem:

► **Theorem 8.** *The problem of deciding, for given  $G$ ,  $T \geq 3$ ,  $w$ , and  $\lambda \in \mathbb{Q}$ , if  $G$  contains a  $T$ -star packing of node weight at least  $\lambda$ , is NP-hard.*

**Proof.** Recall (see [11] and [14, Sec.12.3]) that the following *perfect 3-uniform hypergraph matching problem* is NP-hard: given a nonempty finite domain  $V$ , a collection of subsets  $\mathcal{E} \subseteq 2^V$ , where each element  $X \in \mathcal{E}$  is of size 3, and an integer  $\mu$ , decide if  $V$  can be covered by at exactly  $\mu := |V|/3$  elements of  $\mathcal{E}$ .

We reduce this problem to node-weighted 3-star packings as follows. Construct a bipartite graph  $G$  taking  $V$  as the left part. For each  $X = \{v_1, v_2, v_3\} \in \mathcal{E}$  add a node  $X$  to the right part and connect it to nodes  $v_1, v_2, v_3$  in the left part. The weights of nodes in the left part are set to  $M$ , where  $M$  is a sufficiently large positive integer; the weights of nodes in the right part are  $-1$ .

Each subcollection  $\mathcal{E}' \subseteq \mathcal{E}$  obeying  $\bigcup \mathcal{E}' = V$  generates a packing  $\mathcal{P}$  of 3-stars (with centers located in the right part and leafs — in the left one). Clearly  $w(\mathcal{P}) = M \cdot |V| - |\mathcal{E}'|$ .

Vice versa, consider a max-weight packing  $\mathcal{P}$  of 3-stars. Assuming  $\bigcup \mathcal{E} = V$ ,  $\mathcal{P}$  must cover all nodes in the left part of  $G$  (since  $M$  is large enough). Let  $\mathcal{E}'$  be the set of nodes in the right part of  $G$  that are covered by  $\mathcal{P}$ . Then  $\bigcup \mathcal{E}' = V$  and  $w(\mathcal{P}) = M \cdot |V| - |\mathcal{E}'|$ . Therefore  $V$  can be covered by  $\mu$  elements of  $\mathcal{E}$  if and only if  $G$  admits a 3-star packing of weight at least  $\lambda := M \cdot |V| - \mu$ . The reduction is complete. ◀

## 4.2 Non-Negative Weights

If node weights are non-negative then the problem is tractable. Recall the construction of the auxiliary network  $H$  and assign non-negative arc costs  $c: AH \rightarrow \mathbb{Q}$  as follows:  $c(v^2, t) := w(v)$  for all  $v \in VG$  and  $c(a) := 0$  for the other arcs  $a$ . Then by Theorem 3 computing a max-cost flow in  $H$  also solves the maximum weight  $T$ -star packing problem. The max-cost flow problem is solvable in strongly-polynomial time (see [6, 7] and also [16, Ch.12] for a survey) but using a general method here is an overkill. Note that the costs are non-zero only on arcs incident to the sink. This makes the problem essentially lexicographic.

In what follows, we employ an equivalent treatment, which involves *multi-terminal* networks. Namely, let  $\Gamma$  be a digraph endowed with arbitrary arc capacities  $u$ . Consider a set of *sources*  $S$  and a sink  $t$  ( $S \subseteq V\Gamma$ ,  $t \in V\Gamma$ ,  $t \notin S$ ). Nodes in  $V\Gamma - S - \{t\}$  are called *inner*. The notion of feasible flows (see Definition 4) extends to multi-terminal networks. Sometimes we use the term  $S$ - $t$  flow to emphasize that  $f$  is a multi-source flow.

The *value* of an  $S$ - $t$  flow  $f$  is  $\text{val}(f) := \sum_{s \in S} \text{div}_f(s)$ . Also let  $w: S \rightarrow \mathbb{Q}_+$  be *weights* of sources. The *weight* of  $f$  is defined as  $w(f) := \sum_{s \in S} w(s) \text{div}_f(s)$ . The goal is to find a feasible  $S$ - $t$  flow  $f$  of maximum weight  $w(f)$ . When  $S = \{s\}$  and  $w(s) = 1$ , this coincides with the usual max-value flow problem.

Clearly this problem is equivalent to its *multi-sink* counterpart (where weights are assigned to sinks rather than sources). Consider the digraph  $H$  constructed in Section 2. Splitting the sink  $t$  into  $n$  copies (one for each node in  $VG$ ) and assigning weights to these new sinks appropriately, one reduces the node-weighted star packing problem to the max-weight multi-sink flow problem.

In what follows, we deal with the max-weight multi-source flow problem in  $\Gamma$ . To solve the



latter, we present a divide-and-conquer algorithm, which is inspired by [17]. Our flow-based approach, however, is more general and is also much simpler to explain.

For  $S', T' \subseteq V\Gamma$ ,  $S' \cap T' = \emptyset$ , a subset  $X \subseteq V\Gamma$  such that  $S' \subseteq X$ ,  $T' \cap X = \emptyset$ , is called an  $S'-T'$  cut. When  $S'$  or  $T'$  is singleton the notation is abbreviated accordingly. A cut  $X$  is called *minimum* (among all  $S'-T'$  cuts) if  $c(\delta^{\text{out}}(X))$  is minimum. A  $u$ -feasible flow  $f$  is said to *saturate*  $X$  if  $f(a) = u(a)$  for all  $a \in \delta^{\text{out}}(X)$  and  $f(a) = 0$  for all  $a \in \delta^{\text{in}}(X)$ . In other words,  $f(\delta^{\text{out}}(X)) = u(\delta^{\text{out}}(X))$  and  $f(\delta^{\text{in}}(X)) = 0$ .

Recall that for a  $u$ -feasible flow  $f$  in a digraph  $\Gamma$ , the *residual graph*  $\Gamma_f = (V\Gamma_f := V\Gamma, A\Gamma_f)$  contains *forward* arcs  $a = (u, v) \in A\Gamma$ , where  $f(a) < u(a)$  (endowed with the *residual* capacity  $u_f(a) := u(a) - f(a)$ ), and also *backward* arcs  $a^{-1} = (v, u)$ , where  $a = (u, v) \in A\Gamma$ ,  $f(a) > 0$  (endowed with the *residual* capacity  $u_f(a^{-1}) := f(a)$ ). For a  $u$ -feasible flow  $f$  in  $\Gamma$  and a  $u_f$ -feasible flow  $g$  in  $\Gamma_f$  the *sum*  $f \oplus g$  is a  $u$ -feasible flow in  $\Gamma$  defined by  $(f \oplus g)(a) := f(a) + g(a) - g(a^{-1})$  (where terms corresponding to non-existent arcs are assumed to be zero).

W.l.o.g. no arc enters a source and no arc leaves a sink in  $\Gamma$ . Sort the sources in the order of decreasing weight:  $w(s_1) \geq w(s_2) \geq \dots \geq w(s_k)$ . For  $i = 1, \dots, k$ , define  $S_i := \{s_1, \dots, s_i\}$ . We find a feasible  $S$ - $t$  flow  $f$  and a collection of cuts  $X_1, \dots, X_k$  such that:

- (1) (i)  $X_1 \subseteq X_2 \subseteq \dots \subseteq X_k$ ;
- (ii) for  $i = 1, \dots, k$ ,  $X_i \cap S = S_i$ ,  $t \notin X_i$ , and  $f$  saturates  $X_i$ .

► **Lemma 9.** *If (1) holds, then  $f$  is both a max-weight and a max-value flow.*

**Proof.** Let  $d_i := w(s_i) - w(s_{i+1})$  for  $i = 1, \dots, k-1$  and  $d_k := w(s_k)$ . For  $i = 1, \dots, k$ , define  $v_i := \text{div}_f(s_1) + \dots + \text{div}_f(s_i)$ . Applying Abel transformation, one gets  $w(f) = d_1 v_1 + \dots + d_k v_k$ .

Fix  $i = 1, \dots, k$  and describe  $f$  as a sum  $f' + f''$ , where  $f'$  is a feasible  $\{s_1, \dots, s_i\}$ - $t$  flow and  $f''$  is a feasible  $\{s_{i+1}, \dots, s_k\}$ - $t$  flow (such  $f'$ ,  $f''$  exist due to flow decomposition theorems, see [5]). Clearly  $\text{val}(f') = v_i$ , therefore  $v_i \leq c(\delta^{\text{out}}(X_i))$ . Summing over  $i = 1, \dots, k$ , we get  $w(f) \leq d_1 c(\delta^{\text{out}}(X_1)) + \dots + d_k c(\delta^{\text{out}}(X_k))$ . By (1)(ii), the above inequality holds with equality, hence  $f$  is a max-weight flow. Also taking  $i = k$  in (1)(ii), we see that  $X_k$  is an  $S$ - $t$  cut saturated by  $f$ . Therefore  $f$  is a max-value flow. ◀

It remains to explain how one can find  $f$  and  $X_i$  obeying (1). Consider an instance  $I = (\Gamma, S = \{s_1, \dots, s_k\}, t)$  (the capacities  $u$  and the weights  $w$  remain fixed during the whole computation and are omitted from notation). If  $k = 1$ , then solving  $I$  reduces to finding a max-value  $s_1$ - $t$  flow  $f$  and a minimum  $s_1$ - $t$  cut  $X_1$ .

Otherwise define  $l := \lfloor k/2 \rfloor$ ,  $S^1 := \{s_1, \dots, s_l\}$ , and  $S^2 := \{s_{l+1}, s_{l+2}, \dots, s_k\}$ . Compute a max-value  $S^1$ - $t$  flow  $h$  and the corresponding minimum  $S^1$ - $t$  cut  $Z$ , which is saturated by  $h$ . Since no arc enters a source, we may assume that  $Z \cap S = S^1$ . To proceed with recursion, construct a pair of problem instances as follows. First, contract  $\bar{Z} := V\Gamma - Z$  in  $\Gamma$  into a new sink  $t^1$  and denote the resulting instance by  $I^1 := (\Gamma^1 := \Gamma/\bar{Z}, S^1, t^1)$ . Second, remove the subset  $Z$  in  $\Gamma_h$  (together with the incident arcs) and denote the resulting instance by  $I^2 := (\Gamma^2 := \Gamma_h - Z, S^2, t)$ .

Let  $f^1$  and  $f^2$  be optimal solutions to  $I^1$  and  $I^2$ , respectively, which are found recursively and satisfy (1) (for  $f := f^1$ ,  $S := S^1$  and for  $f := f^2$ ,  $S := S^2$ ). Construct an optimal solution to  $I$  as follows. First,  $Z$  is a minimum  $S^1$ - $t^1$  cut in  $\Gamma^1$  (since  $Z$  is a minimum  $S^1$ - $t$  cut in  $\Gamma$ ) and by Lemma 9,  $f^1$  is a max-value flow. Hence  $f^1$  saturates  $Z$ . Second,  $f^2$  may be regarded as an  $S^2$ - $t$  flow in  $\Gamma_h$ . The sum  $h \oplus f^2$  forms a  $u$ -feasible  $S$ - $t$  flow in  $\Gamma$  that



also saturates  $Z$ . “Glue”  $f^1$  and  $h \oplus f^2$  along  $\delta^{\text{in}}(Z)$ ,  $\delta^{\text{out}}(Z)$  and construct an  $S$ - $t$  flow  $f$  in  $\Gamma$  as follows:

$$f(a) := \begin{cases} f^1(a) & \text{for } a \in \gamma(Z), \\ (h \oplus f^2)(a) & \text{for } a \in \gamma(\overline{Z}), \\ u(a) & \text{for } a \in \delta^{\text{out}}(Z), \\ 0 & \text{for } a \in \delta^{\text{in}}(Z). \end{cases}$$

Let  $X_1^1, X_2^1, \dots, X_l^1$  and  $X_{l+1}^2, X_{l+2}^2, \dots, X_k^2$  be the sequence of nested cuts (as in (1)) for  $f^1$  and  $f^2$  (respectively). Then clearly  $X_1^1, X_2^1, \dots, X_l^1, Z \cup X_{l+1}^2, Z \cup X_{l+2}^2, \dots, Z \cup X_k^2$  and  $f$  obey (1). The description of the algorithm is complete.

Let  $\Phi(n', m')$  denote the complexity of a max-flow computation in a network with  $n'$  nodes and  $m'$  arcs. Let the above recursive algorithm be applied to a network with  $n$  nodes,  $m$  arcs, and  $k$  sources. Then its running time  $T(n, m, k)$  obeys the recurrence

$$T(n, m, k) = \Phi(n, m) + T(n^1, m^1, \lfloor k/2 \rfloor) + T(n^2, m^2, \lceil k/2 \rceil) + O(n + m),$$

where  $n^1 + n^2 = n + 1$ ,  $m^1 + m^2 = m$ . For a “natural” time bound  $\Phi$  this yields  $T(n, m, k) = O(\Phi(n, m) \cdot \log k)$  (see [10, Sec. 2.3]).

► **Theorem 10.** *In a network with  $n$  nodes,  $m$  arcs, and  $k$  sources a max-weight flow can be found in  $O(\Phi(n, m) \cdot \log k)$  time.*

For node-weighted star packings,  $\Phi(n, m) = O(m\sqrt{n})$  for the max-flow problems arising during the recursive process (due to results of [2, 9]).

► **Corollary 11.** *The node-weighted  $T$ -star packing problem with non-negative weights is solvable in  $O(m\sqrt{n} \log n)$  time.*

### 4.3 Node-Weighted Packings of 2-Stars

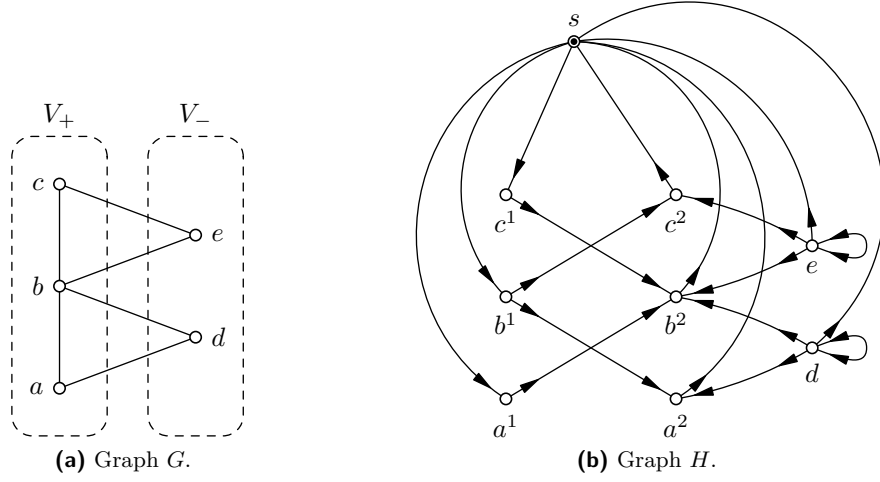
We still have a case where neither a polynomial algorithm nor a hardness result are established. Let  $T = 2$  and node weights be arbitrary. Hence  $T$ -stars are just paths of length 1 and 2. This case is tractable but the needed machinery is of a bit different nature.

Recall the proof of Theorem 8. The latter fails for  $T = 2$  because it shows a reduction from a version of the set cover problem where all subsets are restricted to be of size 1 and 2. The latter set cover problem is equivalent to finding a minimum cardinality *edge cover* in a general (i.e. not necessarily bipartite) graph. Both cardinality and weighted problems regarding edge covers are polynomially solvable (see [16, Ch.27]), so no hardness result can be obtained this way. However, this gives a clue on what techniques may apply here.

We employ the concept of bidirected graphs, which was introduced by Edmonds and Johnson [3] (more about bidirected graphs can be found in, e.g., [16, Ch. 36].) Recall that in a *bidirected* graph edges of three types are allowed: a usual directed edge, or an *arc*, that leaves one node and enters another one; an edge directed *from both* of its ends; and an edge directed *to both* of its ends. When both ends of an edge coincide, the edge becomes a loop.

The notion of a flow is extended to bidirected graphs in a natural fashion. Namely, let  $\Gamma$  is a bidirected graph whose edges are endowed with integer capacities  $u: E\Gamma \rightarrow \mathbb{Z}_+$  and let  $s$  be a distinguished node (a *terminal*). Nodes in  $V\Gamma - \{s\}$  are called *inner*.

► **Definition 12.** A  $u$ -feasible (or just feasible) integer bidirected flow  $f$  is a function  $f: E\Gamma \rightarrow \mathbb{Z}_+$  such that: (i)  $f(e) \leq u(e)$  for each  $e \in E\Gamma$ ; and (ii)  $\text{div}_f(v) = 0$  for each inner node  $v$ .



■ **Figure 2** Reduction to a bidirected graph.

Here, as usual,  $\text{div}_f(v) := f(\delta^{\text{out}}(v)) - f(\delta^{\text{in}}(v))$ , where  $\delta^{\text{in}}(v)$  denotes the set of edges entering  $v$  and  $\delta^{\text{out}}(v)$  denotes the set of edges leaving  $v$ . It is important to note that a loop  $e$  entering (respectively leaving) a node  $v$  is counted **two times** in  $\delta^{\text{in}}(v)$  (respectively in  $\delta^{\text{out}}(v)$ ) and hence contributes  $\pm 2f(e)$  to  $\text{div}_f(v)$ . Similar to flows in digraphs,  $f(\{u, v\})$  is abbreviated to  $f(u, v)$ .

Consider an undirected graph  $G$  endowed with arbitrary node weights  $w: VG \rightarrow \mathbb{Q}$ . We reduce the node-weighted 2-star packing problem in  $G$  to finding a feasible max-cost integer bidirected flow in an auxiliary bidirected graph. The latter is solvable in strongly polynomial time [16, Ch. 36].

To construct the desired bidirected graph  $H$ , denote  $V_+ := \{v \in VG \mid w(v) \geq 0\}$  and  $V_- := VG \setminus V_+$ . Like in Section 2, consider two disjoint copies of  $V_+$  and denote them by  $V_+^1$  and  $V_+^2$ . Also add a terminal  $s$  and define  $VH := V_+^1 \cup V_+^2 \cup V_- \cup \{s\}$ .

One may assume that no two nodes in  $V_-$  are connected by an edge since these edges may be removed without changing the optimum. For an edge  $\{u, v\} \in EG$ ,  $u, v \in V_+$ , construct edges  $\{u^1, v^2\}$  (leaving  $u^1$  and entering  $v^2$ ) and  $\{v^1, u^2\}$  (leaving  $v^1$  and entering  $u^2$ ). For an edge  $\{u, v\} \in EG$ ,  $u \in V_-$ ,  $v \in V_+$ , construct an edge  $\{u, v^2\}$  (leaving  $u^1$  and entering  $v^2$ ). All these bidirected edges are endowed with infinite capacities and zero costs.

For each node  $v \in V_+$ , add an edge  $\{s, v^1\}$  (entering  $v^1$ ) of capacity 2 and zero cost, and an edge  $\{v^2, s\}$  (leaving  $v^2$ ) of capacity 1 and cost  $w(v)$ . For each node  $v \in V_+$ , add a loop  $\{v, v\}$  (entering  $v$  twice) of capacity 1 and cost  $w(v)$  and an edge  $\{v, s\}$  (leaving  $v$ ) of infinite capacity and zero cost. (Since  $s$  is a terminal, directions of edges at  $s$  are irrelevant.) An example is depicted in Fig. 2.

► **Theorem 13.** *The maximum cost of a feasible integer bidirected flow in  $H$  coincides with the maximum weight of a 2-star packing in  $G$ .*

**Proof.** We first show how to turn a max-weight 2-star packing  $\mathcal{P}$  in  $G$  into a feasible integer bidirected flow  $f$  in  $H$  of cost  $w(\mathcal{P})$ . Start with  $f := 0$ . Let  $S$  be a star in  $\mathcal{P}$ . The following cases are possible.

**Case I:**  $S$  covers two nodes, say  $p$  and  $q$ , and  $\{p, q\}$  is the edge of  $S$ .

**Subcase I.1:**  $p, q \in V_+$ . Increase  $f$  by one along the paths  $(s, p^1, q^2, s)$  and  $(s, q^1, p^2, s)$ . This preserves zero divergences at inner nodes and adds  $w(p) + w(q) = w(S)$  to  $c(f)$ .

**Subcase I.2:**  $p \in V_+$ ,  $q \in V_-$ . Increase  $f$  by one along the path  $(s, p^2, q, q, s)$  (where the  $q, q$  fragment denotes the loop at  $q$ ). Divergences at inner nodes are preserved,  $c(f)$  is increased by  $w(p) + w(q) = w(S)$ .

**Case II:**  $S$  covers three nodes, say  $p$ ,  $q$ , and  $r$ , and  $\{p, q\}, \{q, r\}$  are the edges of  $S$ .

**Subcase II.1:**  $p, q, r \in V_+$ . Increase  $f$  by one along the paths  $(s, q^1, p^2, s)$ ,  $(s, q^1, r^2, s)$ , and  $(s, p^1, q^2, s)$ . Divergences at inner nodes are preserved,  $c(f)$  is increased by  $w(p) + w(q) + w(r) = w(S)$ .

**Subcase II.2:**  $p, r \in V_+$  and  $q \in V_-$ . Increase  $f$  by one along the path  $(s, p^2, q, q, r^2, s)$  (as above, the  $q, q$  fragment is the loop at  $q$ ). Divergences at inner nodes are preserved,  $c(f)$  is increased by  $w(p) + w(q) + w(r) = w(S)$ .

Since  $\mathcal{P}$  is optimal, the other cases are impossible. Applying the above to all  $S \in \mathcal{P}$  one gets a feasible integer bidirected flow of cost  $w(\mathcal{P})$ , as claimed.

For the opposite direction, consider a feasible max-cost integer bidirected flow  $f$  in  $H$  and construct a 2-star packing  $\mathcal{P}$  obeying  $w(\mathcal{P}) \geq c(f)$  as follows. Define

$$F_+ := \{(u, v) \mid u, v \in V_+, f(u^1, v^2) > 0\},$$

$$F_- := \{(u, v) \mid u \in V_-, v \in V_+, f(u, v^2) > 0\}.$$

Then  $F := F_+ \cup F_-$  is a 2-feasible arc set in  $\vec{G}$ . (Recall that  $\vec{G}$  is obtained from  $G$  by replacing each edge with a pair of opposite arcs.) Indeed, every arc in  $F$  leaving a node  $u \in V_+$  corresponds to a unit of flow along the edge  $\{s, u^1\}$  and the capacity of the latter is 2. Every arc in  $F$  leaving a node  $u \in V_-$  corresponds to a unit of flow along the edge  $\{u, v^2\}$ ,  $v \in V_+$ , and since the capacity of the loop  $\{v, v, \}$  is 1, there can be at most 2 such arcs. Next, if an arc in  $F$  enters a node  $v \in V_+$  then this arc adds a unit of flow along the edge  $\{v^2, s\}$  (whose capacity is 1). Finally, no arc in  $F$  enters a node in  $V_-$ .

By Theorem 3,  $F$  generates a packing of 2-stars  $\mathcal{P}$  in  $G$ . We claim that  $w(\mathcal{P}) \geq c(f)$ . We show that each edge  $e \in EH$  with  $c(e) > 0$  and  $f(e) = 1$  corresponds to a node  $v_e \in VG$  covered by  $\mathcal{P}$  such that  $c(e) = w(v_e)$ . Also each node  $v \in V_-$  covered by  $\mathcal{P}$  corresponds to an edge  $e_v \in EH$  with  $f(e_v) = 1$  such that  $c(e_v) = w(v)$ . (The mappings  $e \mapsto v_e$  and  $v \mapsto e_v$  are injective.) These observations complete the proof of Theorem 13.

For the first part, consider an edge  $e = \{v^2, s\}$ , where  $f(e) = 1$  and  $v \in V_+$ . Then  $v$  is entered by an arc in  $F$ , hence  $\mathcal{P}$  covers  $v_e := v$ . For the second part, consider a node  $v \in V_-$  covered by  $\mathcal{P}$ . Then  $v$  must be an endpoint of an arc  $a \in F$ . No arc in  $F$  can enter  $v$  (by the construction of  $F$ ), hence  $a = (v, u)$  for  $u \in V_+$ . Therefore  $a \in F_-$  corresponds to the edge  $\{v, u^2\}$ . Since  $f(v, u^2) > 0$  one has  $f(e_v) = 1$ , where  $e_v := \{v, v\}$  is the loop at  $v$ . ◀

**Acknowledgements.** We thank anonymous referees for useful suggestions.

## References

- 1 A. Amahashi and M. Kano. On factors with given components. *Discrete Math.*, 42(1):1–6, 1982.
- 2 E. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl.*, 11:1277–1280, 1970.
- 3 J. Edmonds and E. L. Johnson. Matching, a well-solved class of integer linear programs. In *Proc. Calgary Int. Conf. on Comb. Structures and Their Appl.*, pages 89–92, NY, 1970. Gordon and Breach.
- 4 T. Feder and R. Motwani. Clique partitions, graph compression and speeding-up algorithms. *J. Comput. Syst. Sci.*, 51:261–272, October 1995.

- 5 L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
- 6 A. Goldberg and R. Tarjan. Solving minimum-cost flow problems by successive approximation. In *Proc. 18th Annual ACM Conference on Theory of Computing*, pages 7–18, 1987.
- 7 A. Goldberg and R. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *J. ACM*, 36(4):873–886, 1989.
- 8 P. Hell and D. Kirkpatrick. Packings by complete bipartite graphs. *SIAM J. Algebraic Discrete Methods*, 7(2):199–209, 1986.
- 9 J. Hopcroft and R. Karp. An  $n^{\frac{5}{2}}$  algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.
- 10 T. Ibaraki, A. Karzanov, and H. Nagamochi. A fast algorithm for finding a maximum free multiflow in an inner eulerian network and some generalizations. *Combinatorica*, 18(1):61–83, 1998.
- 11 R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
- 12 A. Kelmans. Optimal packing of induced stars in a graph. *Discrete Math.*, 173(1-3):97–127, 1997.
- 13 M. Las Vergnas. An extension of Tutte’s 1-factor theorem. *Discrete Math.*, 23:241–255, 1978.
- 14 L. Lovász and M. D. Plummer. *Matching Theory*. North-Holland, NY, 1986.
- 15 Q. Ning. On the star packing problem. In *Proc. 1st China-USA International Graph Theory Conference*, volume 576, pages 411–416, 1989.
- 16 A. Schrijver. *Combinatorial Optimization*. Springer, Berlin, 2003.
- 17 T. Spencer and E. Mayr. Node weighted matching. In *Proc. 11th Colloquium on Automata, Languages and Programming*, pages 454–464, London, UK, 1984. Springer-Verlag.
- 18 R. Tarjan. *Data structures and network algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1983.