Algorithmica (2004) 39: 175–187
DOI: 10.1007/s00453-004-1087-0

Algorithmica

Approximations for Maximum Transportation with Permutable Supply Vector and Other Capacitated Star Packing Problems[1]

Esther M. Arkin,[2] Refael Hassin,[3] Shlomi Rubinstein,[3] and Maxim Sviridenko[4]

Abstract. We describe approximation algorithms for the maximum transportation with permutable supply vector and related problems.

Key Words. Transportation problem, Approximation algorithm, NP-complete problem

1. Introduction. The TRANSPORTATION PROBLEM is a central problem in Optimization. Its input consists of a complete bipartite graph $G = (V_1, V_2, E)$, with a non-negative function w: $E \to \mathbb{R}_+$, integer *supplies* $a_i \geq 0$, $i \in V_1$, and integer *demands* $b_j \geq 0$, $j \in V_2$, where without loss of generality $\sum_{i \in V_1} a_i = \sum_{j \in V_2} b_j$. The problem is to compute *flows* $x_{ij}, i \in V_1, j \in V_2$, such that $\sum_j x_{ij} = a_i$ for every $i \in V_1$, $\sum_i x_{ij} = b_j$ for every $j \in V_2$, and the total weight of the flow, $\sum_E w_{ij} x_{ij}$, is maximized (or minimized). In this paper we assume non-negative weights (representing *profits*) and the goal is then to maximize the total profit. It is well known that the transportation problem is polynomially solvable even when the flows are required to be integers.

One of the problems considered in this paper is a variation of the transportation problem which we call MAXIMUM TRANSPORTATION PROBLEM WITH PERMUTABLE SUPPLY VECTOR (or TPS for short). In this variation, supplies are not attached to the vertices of V_1, but rather to a set of capacitated *facilities* that have to be located at the vertices of V_1, one facility at each vertex. Thus, the problem is both to decide on the location of the facilities associated with the given set $\{a_i\}$ to V_1 and on the flows between V_1 and V_2 so as to maximize the total profit.

TPS has several practical applications. An obvious application mentioned in [9] and [10] arises from maximizing profit for the delivery from $|V_1|$ different production facilities to $|V_2|$ customers, where the demand of customers is given by vector b and the sizes of supply facilities can be chosen as a permutation of vector a. The problem occurs

[1] An extended abstract appeared in the proceedings of SWAT 2002. The first author was partially supported by NSF (CCR-0098172).
[2] Department of Applied Mathematics and Statistics, SUNY Stony Brook, Stony Brook, NY 11794-3600, USA. estie@ams.sunysb.edu.
[3] Department of Statistics and Operations Research, Tel Aviv University, Tel Aviv 69978, Israel. {hassin,shlomiru}@post.tau.ac.il.
[4] IBM T. J. Watson Research Center, Yorktown Heights, P.O. Box 218, NY 10598, USA. sviri@us.ibm.com.

Received August 21, 2002; revised September 17, 2003. Communicated by G. N. Frederickson.
Online publication February 25, 2004.

when we want to construct or extend these facilities but sizes are given by some external decision maker (in real life sizes are often defined before locations because of budget or political considerations). Another application of the minimization variant of the above problem is in a placement model in the design of printed circuit boards [8], [9].

Another closely related problem considered in this paper is MAXIMUM CAPACITATED STAR PACKING. The input to the problem consists of a complete undirected graph $G = (V, E)$ with a non-negative weight function w: $E \to \mathbb{R}_+$ and a vector $c = (c_1, \ldots, c_p)$ of integers. We use $w(E')$ to denote the total weight of a subset E' of edges. A *star* is a subset of edges with a common vertex called the *center* of the star. The other vertices are *leaves* of the star. The *size*, or *capacity*, of a star is its number of edges (equivalently, leaves). The *weight* of a star is the total weight of its edges. We consider the MAXIMUM CAPACITATED STAR PACKING problem in which it is required to compute a set of vertex-disjoint stars in G of sizes $c_1, \ldots, c_p$, so as to maximize their total weight, where $\sum_{i=1}^{p} c_i = |V| - p$. The problem can be thought of as a facility location problem. Facilities of given sizes c_i are to be located at vertices to serve customers, where the profit is given by the weight of the edge. We call the special case of TPS with unit demands MAXIMUM CAPACITATED STAR PACKING IN BIPARTITE GRAPHS.

Previous Work. The MAXIMUM STAR PACKING PROBLEM is NP-hard even when all capacities are 2 and 0/1 weights [5, p. 76]. The NP-hardness of the minimum TPS problem and polynomially solvable subcases of the problem were studied by Meusel and Burkard [9] and Hujter [6]. Most of their results are valid for the maximization problem, too, e.g., the TPS with $a_i \in \{0, 1, 2\}$ is polynomially solvable by reduction to the maximum weight f-factor problem [6]. Wolsey [11] analyzed "greedy" heuristics for several discrete facility location problems in which monotone submodular functions are maximized. In particular, his results imply the existence of a $(1 - e^{-1})$-approximation algorithm for the special case of TPS with $b_j = 1$, $j \in V_2$, and $a_i \in \{0, A\}$, $i \in V_1$, for some positive integer A. Unfortunately his approach cannot be generalized for the more general problems considered in this paper since the objective function is not necessarily submodular. For example, consider a MAX CAPACITATED STAR PACKING instance defined on a graph on eight nodes, $V = \{x, y, 1, 2, 3, 4, 5, 6\}$, with $c = (3, 1, 1)$. The edges $(x, 1)$, $(x, 2)$, $(x, 3)$, $(y, 4)$, $(y, 5)$, $(y, 6)$, $(5, 6)$ have weight 1, and all other edges have weight 0. Let $f(S)$ denote the optimal solution when centers are allowed only among nodes of $S \subset V$. Then $f(\{y\}) = f(\{y, 5\}) = 3$, $f(\{x, y\}) = 4$, and $f(\{x, y, 5\}) = 5$ showing that f is not submodular (the addition of 5 to y does not increase f while its addition to $\{x, y\}$ does).

The MAXIMUM QUADRATIC ASSIGNMENT PROBLEM is a generalization of the MAXIMUM CAPACITATED STAR PACKING problem as well as many other problems. Three $n \times n$ non-negative symmetric matrices $A = (a_{ij})$, $B = (b_{ij})$, and $C = (c_{ij})$ are given and the objective is to compute a permutation π of $V = \{1, \ldots, n\}$ so that $\sum_{i,j \in V, i \neq j} a_{\pi(i),\pi(j)} b_{i,j} + \sum_{i \in V} c_{i,\pi(i)}$ is maximized. A $\frac{1}{4}$-approximation algorithm for the MAXIMUM QUADRATIC ASSIGNMENT problem assuming the triangle inequality on matrix B was given in [3]. The problem is also a special case of the MAXIMUM k-SET PACKING PROBLEM where k is a maximum star size. The results of Hurkens and Schrijver [7] imply that for 0-1 weights a local search algorithm is almost a $2/k$-approximation algorithm. Arkin and Hassin [2] showed that the same local search is almost a $1/(k-1)$-

approximation algorithm when general non-negative weights are allowed. Chandra and Halldórsson [4] showed that a combination of a greedy algorithm and local search improves the bound for $k \geq 5$ to $3/2(k+1)$.

Our Results. In this paper we design approximation algorithms for the TPS and MAXIMUM CAPACITATED STAR PACKING, improving previously known approximation algorithms. We prove that:

- A greedy type algorithm with modified edge weights is a $\frac{1}{2}$-approximation for the TPS.
- A local search algorithm can be made arbitrarily close to be a $\frac{1}{2}$-approximation algorithm for the TPS by increasing its depth.
- There is an example showing that a natural greedy algorithm does not guarantee any bound for the MAXIMUM CAPACITATED STAR PACKING.
- A low-depth local search algorithm is a $\frac{1}{3}$-approximation algorithm for MAXIMUM CAPACITATED STAR PACKING.
- A matching-based algorithm is a $\frac{1}{2}$-approximation algorithm for MAXIMUM CAPACITATED STAR PACKING if the edge weights satisfy the triangle inequality.

Note that all algorithms studied in this paper are rather simple, natural, and very fast and thus they are good candidates for practical usage.

2. Star Packing in Bipartite Graphs. In this section we derive approximation algorithms for a special case of the TPS. In the next section we show how they can be generalized for the general problem. We consider MAXIMUM CAPACITATED STAR PACKING PROBLEM IN BIPARTITE GRAPHS. An instance of this problem consists of a complete bipartite graph $G = (V_1, V_2, E)$ with non-negative edge weights w. Let $p = |V_1|$. The problem is to locate p centers of stars of cardinality $c_1 \geq \cdots \geq c_p$ at the vertices of V_1 and assign vertices of V_2, $|V_2| = \sum_{i=1}^{p} c_i$, to star centers, satisfying the cardinality constraints on sizes of stars.

2.1. *Greedy Algorithm.* We consider a greedy algorithm that modifies the weights of edges, algorithm GR in Figure 1. Note that GR selects in each iteration a maximum weight star with respect to modified weights which reflect a deletion of an edge from the partial solution, when a new edge entering the same vertex is selected. Algorithm GR outputs a partial solution, i.e., a set of stars $S_1, \ldots, S_p$ such that $|S_i| \leq c_i$. Since all original edge weights w are non-negative, this solution can be completed to a feasible capacitated star packing without decreasing its value. Note that the modified edge weights may be negative. However, such edges are never chosen to the star S_i in Step i since all modified edges are incident to $\bigcup_{k=1}^{i-1} X_i$ and therefore any vertex $v \in V_1$ has at least c_i non-modified (and therefore non-negative) edges incident to it in Step i.

THEOREM 1. *Let APX be an arbitrary completion of the partial solution returned by GR, and let OPT be an optimal solution. Then $w(APX) \geq w(OPT)/2$ and the bound is asymptotically achievable.*

GR
input
A weighted bipartite graph $G = (V_1, V_2, E, w)$ *and integers* $c_1 \geq \cdots \geq c_p$.
returns
A partial solution.
begin
Let $\bar{w}$ *be a weight function on edges such that* $\bar{w} = w$.
for $i = 1, \ldots, p$
Let $S_i = (v_i, X_i)$ *be a star of maximum weight with respect to* $\bar{w}$
such that $v_i \in V_1$, $X_i \subseteq V_2$, $|X_i| = c_i$.
Delete from $S_1, \ldots, S_{i-1}$ *the edges which enter* X_i, *delete* v_i *from* V_1.
for every $x \in X_i$ *and* $y \in V_1$
$\bar{w}_{yx} := w_{yx} - w_{v_i x}$.
return $S_1, \ldots, S_p$.
end *GR*

Fig. 1. Algorithm GR.

PROOF. In Step i of GR we have in our approximate solution stars $S_1, \ldots, S_i$, where $|S_k| \leq c_k, k = 1, \ldots, i$. The size of S_k is exactly c_k during the kth step of the algorithm but it may decrease in future steps. In each step the set V_1 and the weight function $\bar{w}$ are modified, so at the end of Step i of the algorithm we have an instance I_i of the problem on the bipartite graph (V_1, V_2, E) and weight function $\bar{w}_i$. Let OPT_i be an optimal solution of the STAR PACKING IN BIPARTITE GRAPHS with star sizes $c_{i+1}, \ldots, c_p$ on the graph $(V_1, V_2, E, \bar{w}_i)$. We will prove that for $i = 1, \ldots, p$,

$$2\bar{w}_{i-1}(S_i) \geq \bar{w}_{i-1}(OPT_{i-1}) - \bar{w}_i(OPT_i). \tag{1}$$

Note that by convention we assume that $\bar{w}_p(OPT_p) = 0$ since we must allocate the empty set of stars in I_p. By summing (1) over $i = 1, \ldots, p$ we get

$$\begin{aligned} 2\sum_{i=1}^{p} \bar{w}_{i-1}(S_i) &\geq \sum_{i=1}^{p} (\bar{w}_{i-1}(OPT_{i-1}) - \bar{w}_i(OPT_i)) \\ &= \bar{w}_0(OPT_0) - \bar{w}_p(OPT_p) = w(OPT). \end{aligned}$$

Note that $\bar{w}_{i-1}(S_i)$ is the weight of star S_i in Step i. We emphasize this since some edges of S_i could be removed in the future but their weight is compensated by changing the weight function, i.e., the sum of original edge weights of final stars is at least $\sum_{i=1}^{p} \bar{w}_{i-1}(S_i)$ and therefore, $w(APX) \geq \sum_{i=1}^{p} \bar{w}_{i-1}(S_i) \geq w(OPT)/2$. To prove (1) we construct a solution SOL_i to I_i such that

$$2\bar{w}_{i-1}(S_i) \geq \bar{w}_{i-1}(OPT_{i-1}) - \bar{w}_i(SOL_i). \tag{2}$$

Assume that algorithm GR chooses the star S_i with the center v_i in Step i. If OPT_{i-1} has the star with the same index (i.e., the star with cardinality c_i) located at v_i, then we define SOL_i from OPT_{i-1} by deleting vertex v_i and the star of cardinality c_i located at

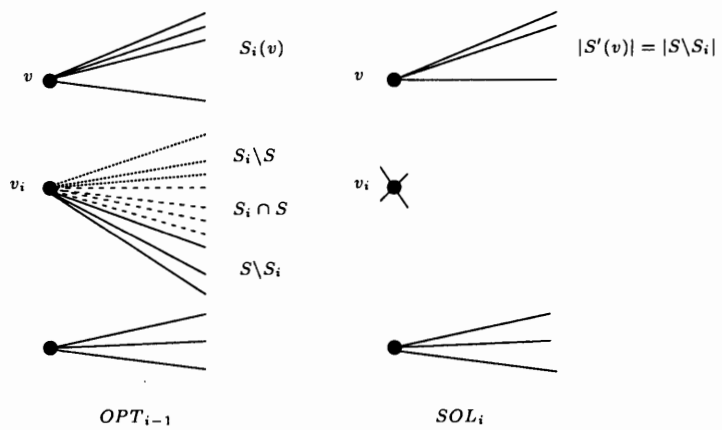

Fig. 2. The solutions OPT_{i-1} and SOL_i.

this vertex from OPT_{i-1} (Figure 2). In this case

$$\bar{w}_{i-1}(S_i) = \bar{w}_{i-1}(OPT_{i-1}) - \bar{w}_{i-1}(SOL_i).$$

We obtain (2) by noticing that $\bar{w}_i(SOL_i) = \bar{w}_{i-1}(SOL_i)$ since edges belonging to stars in SOL_i are not incident to edges from S_i and therefore their weight does not change.

However, in general OPT_{i-1} can have another star S located at v_i. In this case OPT_{i-1} has the star of cardinality c_i located at another vertex, say v, we call this star $S_i(v)$. Since $c_1 \geq \cdots \geq c_p$, we know that $|S_i| \geq |S| \geq |S \backslash S_i|$. We construct SOL_i from OPT_{i-1} as follows. First we delete stars $S_i(v)$ and S from OPT_{i-1}. We do that since we cannot have a star with index i in SOL_i and since vertex v_i is in S_i. After that we try to recover the weight we lost when we deleted $S \backslash S_i$ (generally speaking, we did not loose $S \cap S_i$ since these edges are used in the greedy solution) by moving the center of S from v_i to v and using $|S \backslash S_i|$ heaviest edges with respect to the weight function $\bar{w}_{i-1}$ from the $|S_i|$ available edges at v. We call this new star $S'(v)$. We define SOL_i from OPT_{i-1} by removing vertex v_i from OPT_{i-1} and by defining new star $S'(v)$ at the vertex v instead of $S_i(v)$.

Denote by $av(C)$ the average weight, with respect to $\bar{w}_{i-1}$, of an edge in the star C. By the greedy property, S_i uses the heaviest edges touching vertex v_i. Also their total weight is at least the weight of $S_i(v)$. Therefore, $av(S_i) \geq av(S \backslash S_i)$ and also $av(S_i) \geq av(S_i(v))$. Since $S'(v)$ consists of the heaviest edges in $S_i(v)$, it also follows that $av(S_i) \geq av(S_i(v)) \geq av(S_i(v) \backslash S'(v))$. Since $|S \backslash S_i| = |S'(v)|$ and $S'(v) \subseteq S_i(v)$, $|S_i| = |S \backslash S_i| + |S_i(v) \backslash S'(v)|$. Therefore,

$$\begin{aligned}
\bar{w}_{i-1}(S_i) &= av(S_i)\,|S_i| \\
&\geq av(S')|S \backslash S_i| + av(S_i(v) \backslash S'(v))|S_i(v) \backslash S'(v)| \\
&= \bar{w}_{i-1}(S \backslash S_i) + \bar{w}_{i-1}(S_i(v) \backslash S'(v)) \\
&= \bar{w}_{i-1}(S \backslash S_i) + \bar{w}_{i-1}(S_i(v)) - \bar{w}_{i-1}(S'(v)),
\end{aligned}$$

where the last equality follows since $S'(v) \subseteq S_i(v)$. Then

$$\begin{aligned}\bar{w}_{i-1}(SOL_i) &= w_{i-1}(OPT_{i-1}) - \bar{w}_{i-1}(S) - \bar{w}_{i-1}(S_i(v)) + \bar{w}_{i-1}(S'(v)) \\ &= \bar{w}_{i-1}(OPT_{i-1}) - \bar{w}_{i-1}(S \cap S_i) \\ &\quad - \bar{w}_{i-1}(S \backslash S_i) - \bar{w}_{i-1}(S_i(v)) + \bar{w}_{i-1}(S'(v)) \\ &\geq \bar{w}_{i-1}(OPT_{i-1}) - \bar{w}_{i-1}(S_i) - \bar{w}_{i-1}(S \cap S_i).\end{aligned}$$

Equation (2) follows now from the fact that $\bar{w}_i(SOL_i) \geq \bar{w}_{i-1}(SOL_i) - \bar{w}_{i-1}(S_i \backslash S)$ since all edges in SOL_i are also edges in OPT_{i-1}, therefore their weight cannot be influenced by deleting edges from $S \cap S_i$ and changing weights of edges incident to $S \cap S_i$ since edges of OPT_{i-1} do not touch edges from S because S belongs to OPT_{i-1} and, therefore, only those edge of SOL_i decrease their weights which touch vertices from $S_i \backslash S$ and their total decrease is at most $\bar{w}_{i-1}(S_i \backslash S)$.

We now show that GR does not guarantee more than $\frac{1}{2}w(OPT)$. Let $V_1 = \{x, y\}$ and $|V_2| = c + 1$. V_2 has a special vertex z, and the edge weights are $w_{x,z} = c$, $w_{x,v} = 0$ for $v \in V_2 \backslash \{z\}$, $w_{y,v} = 1$ for $v \in V_2$. The capacities are $(c, 1)$. The optimal solution locates the center with capacity c at y and then $w(OPT) = 2c$, while GR may locate the center of capacity c at x in which case $w(APX) = c + 1$. □

We note that the same example shows that the more sophisticated algorithm GR1 (see Figure 4) also does not guarantee more than $\frac{1}{2}w(OPT)$.

2.2. *Local Search Algorithm.* In this section we consider a natural local search algorithm. We denote by *APX* (*OPT*) an approximate (optimal) solution. A *t-move* is defined as follows: Take a set of at most t centers of the current solution and permute their locations. To decide which leaves belong to which center, we solve a (max) transportation problem with supplies at the permuted centers each with supply equal to its capacity, and unit demands. The weight of the edge (v_i, j) between a center v_i and a leaf j, is modified to be the original weight $w(v_i, j)$ minus the weight of the single edge of the current approximate solution which touches vertex j.

A local search algorithm of depth t (Figure 3) is defined as follows: Start with some feasible solution, and check whether its weight can be increased by a t-move. Repeat until no further improvements are possible.

THEOREM 2. t-search *is a* $t/(1 + 2t)$*-approximation algorithm for the* MAXIMUM CAPACITATED STAR PACKING *problem in a bipartite graph.*

PROOF. Fix some optimal solution and let s_i^o be the center of the ith star in this optimal solution, and let s_i^a be the center of star S_i^a. Consider a directed graph $D = (V, A)$ with $(u, v) \in A$ if $u = s_i^a$ and $v = s_i^o$ for some $i = 1, \ldots, p$. Note that D consists of a collection of directed cycles (possibly loops). We call these *chains*. The *length* of a chain is the number of its vertices. We define a collection of subchains covering the arcs of the graph D, with at most t centers of *OPT* in each subchain, and such that each vertex appears in exactly t subchains.

Chains of length at most t, we duplicate t times. Chains of length greater than t we decompose into subchains of length t. If the chain is a cycle $v_1, v_2, \ldots, v_j, v_1$ $(j > t)$

t-search
input
(G, w, c) *and integers* $c_1 \geq \cdots \geq c_p$
returns
Vertex disjoint stars with sizes c.
begin
Start with a feasible solution $S_1^a, \ldots, S_p^a$.
while $\exists$ *an improving t-move*
Perform an improving t-move.
end while
return *stars* $S_1^a, \ldots, S_p^a$.
end *t-search*

Fig. 3. A depth t local search algorithm.

we use subchains $v_i, v_{i+1}, \ldots, v_{i+t}$ for $i = 1, \ldots, j$ where subscripts are modulo j. We consider such a subchain to contain all the approximate centers in it, and all except the first optimal center located at v_i, in it. We thus have j subchains, each containing t centers of OPT and $t + 1$ centers of APX. Each center of OPT appears in exactly t subchains, and each center of APX appears in $t + 1$ subchains.

Consider each of these subchains. Each corresponds to a potential t-move which does not give an improvement (otherwise the algorithm would have executed this move). Define $touch(S_i^o)$ to be the weight of all edges of the approximate solution touching a leaf of the star S_i^o. Let S^a (S^o) be the set of centers s_i^a (s_i^o) in such a chain. We have

$$\sum_{i \in S^a} w(S_i^a) \geq \sum_{i \in S^o} w(S_i^o) - \sum_{i \in S^o} touch(S_i^o).$$

Next, we sum the above inequality over the collection of subchains we generated from graph D. Since each center of APX appears in at most $t + 1$ chains, the sum of the first terms $\sum_{i \in S^a} w(S_i^a)$ is at most $(t + 1)w(APX)$. Recall that the graph G is bipartite, therefore each edge of APX cannot touch two leaves of optimal stars. Thus in summing $\sum_{i \in S^o} touch(S_i^o)$, each optimal star appears t times, and we get that each edge of APX appears at most t times. Finally, the sum of $\sum_{i \in S^o} w(S_i^o)$ is exactly $t \cdot w(OPT)$. Therefore, the summation yields

$$(t + 1)w(APX) \geq t \cdot w(OPT) - t \cdot w(APX)$$

or $w(APX)/w(OPT) \geq t/(1 + 2t)$, completing the proof. □

Thus an approximation factor arbitrarily close to $\frac{1}{2}$ can be achieved by increasing t. We remark that the local search algorithm can be carried out in polynomial time while maintaining the bound up to any desired level of proximity by scaling the weights (see [2] for a detailed description).

3. Maximum Transportation with Permutable Supply. The problem can be reformulated as a MAXIMUM CAPACITATED STAR PACKING problem in bipartite graphs, by

duplicating each vertex $i \in V_2$ b_i times, and then applying the algorithms in Figures 1 or 3. However, this approach yields only a pseudopolynomial time algorithms.

The local search algorithm can be modified to accommodate the demands of vertices in V_2. The step which needs to be modified is the one in which we calculate a new assignment of leaf vertices to centers, which was done using a transportation problem with modified weights. Here the weights must be modified in a more complex way. If we wish to "send" x_{ij} units from supply vertex $i \in V_1$ to demand vertex $j \in V_2$ we calculate the weight as x_{ij} original weights $w(i, j)x_{ij}$ minus the cheapest edges touching j whose total shipment is x_{ij}. (In the earlier case the $x_{ij} = 1$ so we subtracted only the cost of the cheapest edge touching vertex j.) This yields a maximum transportation problem with piecewise linear concave costs. See the textbook [1] which describes a polynomial time algorithm for the equivalent minimum cost piecewise convex problem.

Algorithm GR can also be implemented in polynomial time by a similar observation. Instead of duplicating demand vertices, in each step the algorithm finds the star of maximum (modified) weight. The star can have multiple edges between vertices, and we define the flow x_{ij} from vertex i to j to be equal to the number of edges between vertices i and j. The profit from using x_{ij} edges is defined in the same way, it is $w_{ij}x_{ij}$ minus a total cost of x_{ij} cheapest edges touching demand vertex j. Note that originally we assume that there are b_j edges of zero profit touching j. Therefore, in each step we need to solve a maximum transportation problem with piecewise linear concave costs and just one supply vertex. Actually, in this case the algorithm can be implemented directly without using [1].

4. Star Packing in General Graphs. We now consider star packing in general graphs, and suggest similar algorithms to those introduced for bipartite graphs. As we will see, the greedy approach does not guarantee a constant factor approximation, whereas a low depth local search guarantees a $\frac{1}{3}$-approximation.

4.1. *Greedy Algorithm.* Algorithm GR1 (Figure 4) is a natural greedy algorithm for MAXIMUM CAPACITATED STAR PACKING. It chooses the next star center which maximizes the optimal value of a certain transportation problem between star centers and remaining vertices in the graph. The following example shows that it does not guarantee any constant bound for general graphs and identical capacities.

EXAMPLE 3. Consider a k-ary tree with depth L for some odd integer L. At level l, $l = 0, \ldots, L$, there are k^l vertices. Let the edges between level $l-1$ and l have weight α^l for $0 < \alpha < 1$ satisfying $k = 1/\alpha(1-\alpha)$. The graph consists of sufficiently many additional isolated vertices. All non-tree edges have zero weight.

Consider the star packing instance over the above graph, with capacities k. The number of centers is equal to the number of non-leaf vertices of the tree. GR1 starts by selecting the root. The value of α was selected so that the maximum addition of weight can now be obtained by selecting vertices from either the sons of the root (gaining $k\alpha^2 - \alpha$ per vertex) or their sons (gaining $k\alpha^3$ per vertex). Assume that the former choice is made. Continuing this way, GR1 selects all the non-leaf vertices of the tree as centers. Its value is thus equal to the total weight of the leaf edges which is $w(APX) = k^L\alpha^L$.

*GR*1
input
$(G, w, c), c_1 \geq \cdots \geq c_p$, *such that* $\sum_{i=1}^{p} c_i = |V| - p$.
returns
Vertex disjoint stars with sizes c.
begin
for $i = 1, \ldots, p$
$V_i := \{v_1, \ldots, v_i\}$
Let $v_i \notin \{v_1, \ldots, v_{i-1}\}$ *maximize the value of:*
$\max \sum_{v_l \in V_i} \sum_{j \notin V_i} w(v_l, j)x(v_l, j)$
$\sum_{j \notin V_i} x(v_l, j) = c_l, v_l \in V_i$
$\sum_{l=1}^{i} x(v_l, j) \leq 1, l \in V_i, j \notin V_i.$
$x(v_l, j) \in \{0, 1\}, v_l \in V_i, j \notin V_i.$
return *Stars* $S_1, \ldots, S_p$ *with centers* $v_1, \ldots, v_p$ *and leaves* $X_1, \ldots, X_p$, *respectively.*
end *GR*1

Fig. 4. Algorithm GR1.

The optimal solution selects for centers from the tree only at those vertices at an even distance from the root. The other centers are arbitrarily chosen from the non-tree vertices. The weight of an optimal solution is $w(OPT) = k\alpha + k^3\alpha^3 + \cdots + k^L\alpha^L$.

Let $R = k\alpha = 1/(1-\alpha) > 1$. Then $w(OPT) = R + R^3 + \cdots + R^L$ while $w(APX) = R^L$. Choosing smaller $\alpha > 0$, R becomes closer to 1, and the ratio $w(OPT)/w(APX)$ is about $L/2$. By increasing L we get the claimed result that GR1 does not guarantee any constant bound.

4.2. *Local Search.* Let S_i^a be the stars of an approximate solution, and let S_i^o be the stars of an optimal solution, such that the size of S_i^a is equal to the size of S_i^o, namely c_i. Denote by s_i^a (s_i^o) the center of S_i^a (S_i^o).

A *move*: Remove a star S_i^a and place its center at some v_1. If v_1 was a leaf of another star, we remove that edge. If v_1 is a center of the approximate solution, remove its star. Replace S_i^a by a star centered at v_1 of size c_i. The leaves of the new star are selected greedily, using modified weights of edges (v_1, v_j) which are equal to the original weight minus the weight of edges selected by other stars of the approximate solution that touch v_j. Stars that had leaves removed from them by this process get the appropriate number of new leaves arbitrarily. An improving move is one in which the weight of the approximate solution improves.

THEOREM 4. Local-search *returns a* $\frac{1}{3}$*-approximation for* MAXIMUM STAR PACKING IN GENERAL GRAPHS.

PROOF. Define *touch*(S_i^o) to be the weight of all edges of an approximate solution touching vertices of S_i^o. At the end of the algorithm, no move is an improving move and thus moving the center of the approximate solution to its location in the optimal solution

```
Local-search
  input
  (G, w, c)
  returns
  Vertex disjoint stars with sizes c.
  begin
  Start with a feasible solution S_1^a, ..., S_p^a.
  while ∃ an improving move
        Perform an improving move.
        end while
  return stars S_1^a, ..., S_p^a.
  end Local-search
```

Fig. 5. Local-search algorithm.

is non-improving. Hence,

$$w(S_i^a) \geq w(S_i^o) - touch(S_i^o).$$

Let $w(APX) = \sum_i w(S_i^a)$ be the value of the approximate solution while $w(OPT) = \sum_i w(S_i^o)$ is the optimal value. We now sum the above inequality over all centers $i = 1, \ldots, p$. Note that $\sum_i touch(S_i^o) \leq 2w(APX)$, since each edge of the approximate solution touches at most two stars of the optimal solutions, as the stars are vertex disjoint. The summation therefore yields $w(APX) \geq w(OPT) - 2w(APX)$, or $w(APX) \geq 1/3w(OPT)$. □

4.3. *Metric Case.* We now assume that the weights w_e satisfy the triangle inequality. We describe for this case a $\frac{1}{2}$-approximation algorithm.

A *greedy maximum matching* of size m is obtained by sorting the edges in non-increasing order of their weights and then scanning the list and selecting edges as long as they are vertex-disjoint to the previously selected edges and their number does not exceed m.

LEMMA 5. *Let M be a greedy maximum matching. Let M' be an arbitrary matching. Then, for $i \leq |M|$ and $2i - 1 \leq |M'|$, the weight of the ith largest edge in M is greater than or equal to the weight of the $2i - 1$ largest edge in M'.*

PROOF. Let $e_1, \ldots, e_m$ be the edges of M' in non-increasing order of weight. By the greedy construction, every edge of $e' \in M' \backslash M$ is incident to an edge of $e \in M$ with $w(e) \geq w(e')$. Since every edge of M can take the above role at most twice, it follows that for $e_1, \ldots, e_{2i-1}$ we use at least i edges of M, all of which are at least as large as $w(e_{2i-1})$. □

Assume that p is even. Our approximation algorithm first computes a greedy maximum matching of size $p/2$. The vertices incident to the matching will be the centers of

Metric
input
1. *A complete undirected graph* $G = (V, E)$ *with weights* w_e $e \in E$ *satisfying the triangle inequality.*
2. *Constants* $c_1 \geq \cdots \geq c_p \geq 1$ *such that* $\sum_i c_i = |V| - p$.
returns
Vertex disjoint stars $S_1, \ldots, S_p$ *such that* $|S_i| = c_i$.
begin
Greedily compute a matching $M = (e_1, \ldots, e_m)$ *where* $e_i = (a_i, b_i)$, *and* $m = \lceil p/2 \rceil$.
if p *is even*:
Arbitrarily choose from $V \backslash \{a_1, \ldots, a_m, b_1, \ldots, b_m\}$
disjoint subsets $V_1, \ldots, V_p$ *of sizes* $c_1, \ldots, c_p$ *respectively.*
for $i = 1, \ldots, p/2$
Set (r_{2i-1}, r_{2i}) *to* (a_i, b_i) *or* (b_i, a_i), *each with probability* 0.5.

elseif p *is odd*:
Arbitrarily choose from $V \backslash \{a_1, \ldots, a_m, b_1, \ldots, b_m\}$
subsets $V_1, \ldots, V_p$ *of sizes* $c_1, \ldots, c_{p-1}, c_p - 1$.
for $i = 1, \ldots, (p-1)/2$
Set (r_{2i-1}, r_{2i}) *to* (a_i, b_i) *or* (b_i, a_i), *each with probability* 0.5.
$r_p := a_m$ *or* $r_p := b_m$, *each with probability* 0.5.
$V_p := V_p \cup \{a_m, b_m\} \backslash r_p$.
end if
$S_i :=$ *the star with center* r_i *and leaves* V_i.
return $S_1, \ldots, S_p$.
end *Metric*

Fig. 6. Algorithm for the metric case.

the approximate star packing and the remaining vertices of the graphs will be the leaves. The algorithm takes edges of the greedy matching one by one by decreasing order of their values. It assigns two unassigned star centers of stars of the biggest cardinality to the ends of the edge considered in that step. Since there are two ways to do it, the algorithm chooses the way to assign centers each with probability $\frac{1}{2}$. After that the algorithm arbitrarily assigns leaves to centers. The algorithm for odd p works similarly (see Figure 6 for a precise description of the algorithm).

THEOREM 6. *Given that the edge weights satisfy the triangle inequality, algorithm Metric (Figure 6) is a* $\frac{1}{2}$*-approximation algorithm for* MAXIMUM CAPACITATED STAR PACKING.

PROOF. Let apx be the expected weight of the solution returned by Metric. Let g_i be the length of the ith largest edge in the greedy matching M computed by Metric. Let o_i be the length of the largest edge in the ith star in a given optimal solution of total weight $w(OPT)$. Note that the corresponding edges are a matching. Let $(l_1, \ldots, l_p)$ be the values of $\{o_1, \ldots, o_p\}$ sorted in a non-increasing order.

Suppose first that p is even. Then by the triangle inequality and the randomized way by which the ends of the greedy matching are assigned to centers of stars, the expected weight of each edge selected to the stars S_{2i-1} and S_{2i} is at least $g_i/2$. Therefore,

$$apx \geq \frac{1}{2}\sum_{i=1}^{p/2}(c_{2i-1}+c_{2i})g_i.$$

On the other hand,

$$w(OPT) \leq \sum_{i=1}^{p/2}(c_{2i-1}o_{2i-1}+c_{2i}o_{2i}) \leq \sum_{i=1}^{p/2}(c_{2i-1}l_{2i-1}+c_{2i}l_{2i}) \leq \sum_{i=1}^{p/2}(c_{2i-1}+c_{2i})g_i,$$

where the second inequality follows from Lemma 5. The theorem follows from the above two relations.

Suppose now that p is odd. We repeat the same proof but also use the fact that (by the same lemma) $w(e_m) \geq o_p$. □

We note that algorithm Metric can be easily derandomized. When assigning a_i and b_i to r_{2i-1} and r_{2i}, choose the one with maximum additional weight. This maximum is always bigger than the average, which is bigger than $(c_{2i-1}+c_{2i})g_i/2$.

REMARK 7. The bound given by Theorem 6 is tight, and this is still true even for the derandomized version. This fact is demonstrated by the following example. Let $V = A \cup B$ where $A = \{1, 2, 3\}$ and $B = \{4, 5, 6\}$. Let $w_e = 1$ if e connects A and B, and $w_e = 0$ otherwise. Let $p = 2$ and $c_1 = c_2 = 2$. Clearly, an optimal solution places one center in A and the other in B, and uses four unit-weight edges. Hence, $w(OPT) = 4$. Algorithm Metric will choose for M some unit-weight edge, say $(1, 4)$. It then partitions $V\backslash\{1, 4\}$ arbitrarily into two 2-sets. A possible choice is $V_1 = \{2, 5\}$ and $V_2 = \{3, 6\}$. For this choice, each of the resulting stars has one zero-weight edge and one unit-weight edge, so that $apx = 2$.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] E.M. Arkin and R. Hassin, On Local Search for weighted k-set packing, *Mathematics of Operations Research* **23** (1998), 640–648.

[3] E.M. Arkin, R. Hassin, and M. Sviridenko, Approximating the maximum quadratic assignment problem, *Information Processing Letters* **77** (2001), 13–16.

[4] B. Chandra and M.M. Halldórsson, Greedy local improvement and weighted set packing approximation, *Journal of Algorithms* **39** (2001), 223–240.

[5] M.S. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.

[6] M. Hujter, B. Klinz, and G. Woeginger, A note on the complexity of the transportation problem with a permutable demand vector, *Mathematical Methods of Operations Research* **50** (1999), 9–16.

[7] C.A.J. Hurkens, and 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 Journal on Discrete Mathematics* **2** (1989), 68–72.

[8] S. Meusel, Minimizing the placement-time on printed circuit boards, Ph.D. Thesis, Fakultät für Mathematik und Informatik, TU Bergakademie Freiberg, 1998.

[9] S. Meusel and R. Burkard, A transportation problem with a permuted demand vector, *Mathematical Methods of Operations Research* **50** (1999), 1–7.

[10] B. Steinbrecher, Ein Transportproblem mit permutiertem Bedarfsvektor, Master's thesis, Fakultät für Mathematik und Informatik, TU Bergakademie Freiberg, 1997.

[11] L.A. Wolsey, Maximizing real-valued submodular functions: primal and dual heuristics for location problems, *Mathematics of Operations Research* **7** (1982), 410–425.