ELSEVIER

Operations Research Letters 21 (1997) 133–137

operations research letters

Approximation algorithms for maximum dispersion

Refael Hassin*, Shlomi Rubinstein, Arie Tamir

Department of Statistics and Operations Research, School of Mathematical Sciences, Tel-Aviv University, Ramat Aviv, Tel-Aviv 69978, Israel

Received 1 November 1995; revised 1 May 1997

Abstract

We describe approximation algorithms with bounded performance guarantees for the following problem: A graph is given with edge weights satisfying the triangle inequality, together with two numbers k and p. Find k disjoint subsets of p vertices each, so that the total weight of edges within subsets is maximized. © 1997 Elsevier Science B.V.

Keywords: Approximation algorithms; Maximum dispersion; Facility location

1. Introduction

Given is an undirected complete graph $G=(V,E)$, with a vertex set $V=\{v_1,v_2,\ldots,v_n\}$. Each edge (v_i,v_j) is associated with a nonnegative weight $w(v_i,v_j)$. It is assumed that the edge weights satisfy the triangle inequality. For a subset $V'\subset V$ we denote by $E(V')$ the set of edges in the complete subgraph induced by V'.

Given $p\in\{2,\ldots,n\}$, and $k\in 1,\ldots,\lfloor n/p\rfloor$, we consider the problem of finding k disjoint subsets $P_1,\ldots,P_k$ of V, with $|P_i|=p$, $i=1,\ldots,k$, such that $\sum_{i=1}^{k}\sum_{(x,y)\in E(P_i)} w(x,y)$ is maximized.

This problem and some variations of it, with $k=1$, are the subject of recent papers by Tamir [11, 12], Ravi et al. [10] and Ghosh [7]. In particular, [10] contains an algorithm for obtaining approximations for the case $k=1$. The algorithm can be executed in $O(n^2)$ time. The authors provide a relatively complex proof to show that the approximate solution's weight is at least $\frac{1}{4}$ of the maximum possible weight. They also provide an example where the actual ratio is asymptotically $\frac{1}{2}$. The question of whether the bound of $\frac{1}{4}$ is tight or not remains open. Feo and Khellaf [4] presented an approximation algorithm with a performance guarantee of $\frac{1}{2}$ for the case in which $|V|=kp$, i.e., the problem is of partitioning V into sets of size p.

In this note, we generalize the results of Feo and Khellaf [4]. We present an algorithm with a performance guarantee of $\frac{1}{2}$ for any $k\in\{1,\ldots,\lfloor|V|/p\rfloor\}$. For the case $k=1$ we present a second algorithm that achieves the same bound but with a lower time complexity.

Chandra and Halldórsson [3] provide approximation algorithms for several dispersion problems. In particular, they consider the *remote star problem* in which one wants to find a subset of vertices P, $|P|=p$, maximizing $\min_{x\in P}\sum_{y\in P} w(x,y)$. They prove that our first algorithm, with $k=1$, has a $\frac{1}{2}$ performance guarantee also for the remote star problem.

* Corresponding author. E-mail: hassin@math.tau.ac.il.

PII S0167-6377(97)00034-5

Kortsarz and Peleg [9] consider the case $k = 1$ without the triangle inequality, and present a polynomial algorithm with a performance guarantee of $O(n^{0.3885})$. More recently, Asahiro et al. [1] analyze an algorithm for the case $k = 1$ without the triangle inequality, that greedily removes vertices from the graph. Its performance is particularly good when p is large, e.g., $\frac{4}{9}$ when $p = n/2$.

2. The first algorithm

For a subset $V' \subset V$ let $W(V') = \sum_{(v_i,v_j)\in E(V')} w(v_i, v_j)$ be the total edge weight in the subgraph induced by V' and let $\bar{W}(V') = 2W(V')/(|V'|(|V'|-1))$ be the average edge weight in this subgraph. For a subset of edges E' let $W(E') = \sum_{(v_i,v_j)\in E'} w(v_i, v_j)$ and let $\bar{W}(E') = W(E')/|E'|$ be the average edge weight in E'.

A *q-matching* is a set of q vertex-disjoint edges. A *maximum q-matching* has maximum total weight.

The first algorithm:

- Compute a maximum q-matching M^* in G, where $q = k\lfloor p/2 \rfloor$. Let V^* be the set of vertices of the edges in M^*.
- Arbitrarily partition M^* into k subsets, $M_1, \ldots, M_k$ with $\lfloor p/2 \rfloor$ edges each. Let $P_1, \ldots, P_k$ be the vertices incident with these sets, respectively.
- If p is odd, complete the solution by adding an arbitrary (distinct) vertex from $V \setminus V^*$ to each subset.

Let APX be the output of the algorithm. Let OPT be an optimal solution consisting of subsets $O_1, \ldots, O_k$ of size p each. Let $W(\text{OPT})$ and $W(\text{APX})$ be the respective solution values.

Lemma 2.1. *Let V' be a subset of vertices, $|V'| \geq 2q$, and let $M^*(V')$ be a maximum q-matching on V'. Then, $\bar{W}(V') \leq \bar{W}(M^*(V'))$.*

Proof. The average weight of a q-matching on V' is q times the average weight of an edge in the graph induced by V' since each such edge participates in the same number of q-matchings. Hence, the weight of a maximum q-matching is greater than or equal to $q\bar{W}(V')$ and the average weight of an edge in a maximum q-matching satisfies the claim. □

Theorem 2.2. $(2 - 1/\lceil p/2 \rceil)W(\text{APX}) \geq W(\text{OPT})$.

Proof. Consider an edge $(u, v) \in M^*$ whose vertices were assigned to a subset P_i in APX. Then, by the triangle inequality, for every $x \in P_i \setminus \{u, v\}$,

$$w(u, v) \leq w(u, x) + w(v, x).$$

Summing over all $x \in P_i \setminus \{u, v\}$ we get

$$(p-2)w(u,v) \leq \sum_{x\in P_i\setminus\{u,v\}} (w(u,x) + w(v,x)). \quad (1)$$

Suppose, first that, p is even. Then, each $x \in P_i$ is incident to an edge in M^* and every edge in $E(P_i)\setminus M^*$ is adjacent to two edges of M^*. Summing (1) over M^* we get

$$(p-2)W(M^*) \leq 2(W(\text{APX}) - W(M^*)),$$

or

$$\frac{2}{p} W(\text{APX}) \geq W(M^*). \quad (2)$$

Consider now the case where p is odd. Let y_i be the vertex added to P_i by the last step of the algorithm. Then, by the triangle inequality, for each $(u, v) \in M_i$,

$$w(y_i, u) + w(y_i, v) \geq w(u, v).$$

Summing over all $(u, v) \in M_i$ we get

$$\sum_{u\in P_i\setminus\{y_i\}} w(y_i, u) \geq W(M_i).$$

Summing (1) over M^* and noting that edges incident to y_i participate in this summation exactly once, we obtain

$$\begin{aligned}(p-2)W(M^*) &\leq 2(W(\text{APX}) - W(M^*)) \\ &\quad - \sum_{i=1}^{k} \sum_{u\in P_i\setminus\{y_i\}} w(y_i, u) \\ &\leq 2(W(\text{APX}) - W(M^*)) - W(M^*),\end{aligned}$$

or

$$\frac{2}{p+1} W(\text{APX}) \geq W(M^*). \quad (3)$$

Consider now OPT. For every subset O_i in OPT let N_i be a maximum weight $\lfloor p/2 \rfloor$-matching of its

vertices. Let $N = \bigcup_{i=1}^{k} N_i$. Note that N is a $k\lfloor p/2 \rfloor$-matching in G and, therefore,

$$W(N) \leqslant W(M^*).$$

By Lemma 2.1, $\bar{W}(N_i) \geqslant \bar{W}(O_i)$ for $i = 1, \ldots, k$. Since all the sets O_i have the same size p, it also follows that

$$\begin{aligned} W(\mathrm{OPT}) &= \sum_{i=1}^{k} W(O_i) \\ &= \sum_{i=1}^{k} \frac{p(p-1)}{2} \bar{W}(O_i) \\ &\leqslant \sum_{i=1}^{k} \frac{p(p-1)}{2} \bar{W}(N_i) \\ &= \sum_{i=1}^{k} \frac{p(p-1)}{2\lfloor p/2 \rfloor} W(N_i) \\ &= \frac{W(N)}{\lfloor p/2 \rfloor} \frac{p(p-1)}{2} \\ &\leqslant \frac{W(M^*)}{\lfloor p/2 \rfloor} \frac{p(p-1)}{2}. \end{aligned}$$

If p is even, we use (2) to obtain

$$W(\mathrm{OPT}) \leqslant 2\left(1 - \frac{1}{p}\right) W(\mathrm{APX}).$$

If p is odd, we use (3) to obtain

$$W(\mathrm{OPT}) \leqslant 2\left(1 - \frac{1}{p+1}\right) W(\mathrm{APX}). \qquad \square$$

Remark 2.3. The bound is tight even for $k = 1$. For every even integer p consider an instance with $V = V_1 \cup V_2$ such that $|V_1| = |V_2| = p$, all the edges induced by V_1 have weight 2, V_2 contains a perfect matching of edges of weight 2, and all the other edges are of weight 1. An optimal solution is V_1 with average weight of 2. The perfect matching of edges in V_2 with weight 2 could be selected by the algorithm, and in this case APX $= V_2$ has an average weight of $p/(p-1)$.

Remark 2.4. A maximum matching in a graph with n vertices and m edges can be computed in time $O(n^3)$ [5]. For sparse graphs a better bound of $O(nm + n^2 \log n)$ is achieved in [6].

The running time of the algorithm is dominated by the computation of a maximum q-matching during the first step. We now describe how this computation can be done using a maximum matching algorithm. We extend the graph by adding $n - 2q$ new vertices and connecting each of the new vertices to each of the original ones by a 'heavy edge' of weight $K = W(V) + 1$. The edges between new vertices have zero weight. Now compute a maximum matching. It will certainly use $n - 2q$ heavy edges of weight K each and in addition a maximum weight matching on the original graph of exactly q edges. The time complexity is $O(n^3)$.

A reduction in the complexity is also possible. Using the algorithm in [2], we can delete in $O(n^2)$ time all but the $2q - 1$ highest weight edges incident with each vertex. Clearly, there exists a maximum matching consisting of q edges from the remaining set. Let the new vertices be $u_1, \ldots, u_{n-2q}$. Generate heavy edges (v_i, u_i) for $i = 1, \ldots, n - 2q$ and also (v_i, u_j) for $i = n - 2q + 1, \ldots, n$ and $j = 1, \ldots, n - 2q$. Again, any matching of q edges in V can be completed to a perfect matching on the extended graph by adding $n - 2q$ heavy edges. Therefore, the structure of a maximum matching will be as before. Due to the smaller number of edges in the graph, the complexity reduces to $O(n^2(kp + \log n))$ (using Remark 2.4).

Remark 2.5. As Remark 2.3 shows, the bound of Theorem 2.2 is tight even when the edge weights have only two distinct values. However, in this case the approximation can be obtained through an application of a maximum cardinality matching algorithm.

3. The second algorithm

We now describe for the case $k = 1$, a faster algorithm with the same asymptotic performance guarantee.

- Set $S := \emptyset$.
- Repeat $\lfloor p/2 \rfloor$ times: Let $w(v_i, v_j) = \max\{w(v_k, v_\ell) | (v_k, v_\ell) \in E\}$. Set $S := S \cup \{v_i, v_j\}$. Delete from E the edges incident with v_i or with v_j.
- If p is odd, add to S an arbitrary vertex.

Let GR be the output of the algorithm and, as before, let OPT be an optimal solution.

Remark 3.1. The algorithm determines a greedy solution for the maximum weight matching problem on G. It is well known that the weight of this solution is at least half the weight of a maximum weight matching [8]. It easily follows from the proof of Theorem 2.2 that $W(\mathrm{GR}) \geqslant W(\mathrm{OPT})/4$. We will show, however, that a stronger result holds.

Theorem 3.2. $W(\mathrm{GR}) \geqslant W(\mathrm{OPT})/2$.

Proof. The proof is by induction on p. The theorem is trivially true when $p \leqslant 2$. Suppose it is true for $p-2$ and we will now prove it for p.

Let OPT_p (GR_p) be an optimal (approximate) solution for the graph G for a given value of p. Let $e=(v_i, v_j)$ be a maximum weight edge in G and let G' be the graph induced by $V\backslash\{v_i, v_j\}$. Let OPT'_{p-2} (GR'_{p-2}) be an optimal (approximate) solution on G' with $p-2$ vertices. We can assume that $\mathrm{GR}_p = \mathrm{GR}'_{p-2} \cup \{v_i, v_j\}$.

We decompose the edges induced by GR_p into three sets: $\{e\}$, $\{(v_i, v_k)\colon v_k \in \mathrm{GR}_p\backslash\{v_i, v_j\}\} \cup \{(v_j, v_k)\colon v_k \in \mathrm{GR}_p\backslash\{v_i, v_j\}\}$, and $\{(v_k, v_\ell)\colon v_k, v_\ell \in \mathrm{GR}_p\backslash\{v_i, v_j\}\}$.

We will compare the approximation to the optimal solution in three cases. In each case we distinguish an edge e' induced by OPT_p. By definition, $w(e) \geqslant w(e')$. By the triangle inequality, the $2(p-2)$ edges incident with e in GR_p contribute at least $(p-2)w(e)$ to $W(\mathrm{GR}_p)$ while the $2(p-2)$ edges incident with e' in OPT_p have total weight of at most $2(p-2)w(e)$. The proof will be completed by using the induction hypothesis to show that the other edges induced by the $p-2$ vertices of GR_p have total weight of at least half of the weight of the other edges in OPT_p.

Case 1: $v_i, v_j \in \mathrm{OPT}_p$. Set $e' = e$.

Case 2: $|\mathrm{OPT}_p \cap e| = 1$. Choose $v_x \in \mathrm{OPT}_p\backslash\mathrm{OPT}'_{p-2}$ such that $v_x \notin \{v_i, v_j\}$. If $v_i \in \mathrm{OPT}_p$, set $e' = (v_i, v_x)$. Otherwise set $e' = (v_j, v_x)$.

Case 3: $\mathrm{OPT}_p \cap e = \emptyset$. Choose $e' = (v_x, v_y)$ where $v_x, v_y \in \mathrm{OPT}_p\backslash\mathrm{OPT}'_{p-2}$.

In all of the three cases, the $p-2$ vertices of OPT_p which are not the vertices of the chosen edge e' are in G', so that the inductive assumption can be applied to conclude that

$$\begin{aligned} W(\mathrm{OPT}_p) &\leqslant w(e') + 2(p-2)w(e) + W(\mathrm{OPT}'_{p-2}) \\ &\leqslant w(e) + 2(p-2)w(e) + 2W(\mathrm{GR}'_{p-2}) \\ &\leqslant 2[w(e) + (p-2)w(e) + W(\mathrm{GR}'_{p-2})] \\ &\leqslant 2W(\mathrm{GR}_p). \qquad \square \end{aligned}$$

Remark 3.3. The bound is asymptotically tight as demonstrated again by the example of Remark 2.3.

Straightforward implementation of the algorithm takes $O(pn^2)$ time. We can reduce this bound as follows: For each $v \in V$ we define a subset $A_v \subset E$ of p highest weight edges incident with v. The above task can be performed in $O(n^2)$ time (again, using the algorithm in [2]). Then, for each vertex, using the weight as a key, maintain the remaining $O(p)$ edges in a heap. It is then easy to verify that each of the $\lfloor p/2 \rfloor$ steps of the above greedy algorithm can be executed in $O(p \log p + n)$ time. Therefore, the complexity bound of the algorithm is $O(n^2 + p^2 \log p + pn) = O(n^2 + p^2 \log p)$.

References

[1] Y. Asahiro, K. Iwama, H. Tamaki, T. Tokuyama, Greedily finding a dense graph, in: Proc. 5th Scandinavian Workshop on Algorithm Theory (SWAT), Lecture Notes in Computer Science, vol. 1097, Springer, Berlin, 1996, pp. 136–148.

[2] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest, R.E. Tarjan, Time bounds for selection, J. Comput. System Sci. 7 (1972) 448–461.

[3] B. Chandra, M. Halldórsson, Facility dispersion and remote subgraphs, in: Proc. 5th Scandinavian Workshop on Algorithm Theory (SWAT), Lecture Notes in Computer Science, vol. 1097, Springer, Berlin, 1996, pp. 53–65.

[4] T.A. Feo, M. Khellaf, A class of bounded approximation algorithms for graph partitioning, Networks 20 (1990) 181–195.

[5] H.N. Gabow, An efficient implementation of Edmond's algorithm for maximum matchings on graphs, J. ACM 23 (1975) 221–234.

[6] H.N. Gabow, Data structures for weighted matching and nearest common ancestors with linking, in: Proc. 1st Ann. ACM–SIAM Symp. on Discrete Algebra 1990, pp. 434–443.

[7] J.B. Ghosh, Computational aspects of the maximum diversity problem, Oper. Res. Lett. 19 (1996) 175–181.

[8] B. Korte, D. Hausmann, An analysis of the greedy heuristic for independence systems, Ann. Discrete Math. 2 (1978) 65–74.

[9] G. Kortsarz, D. Peleg, On choosing a dense subgraph, in: Proc. 34th IEEE Ann. Symp. on Foundations of Computer Science, Palo Alto, CA, 1993, pp. 692–701.

[10] S.S. Ravi, D.J. Rosenkrantz, G.K. Tayi, Heuristic and special case algorithms for dispersion problems, Oper. Res. 42 (1994) 299–310.

[11] A. Tamir, Obnoxious facility location on graphs, SIAM J. Discrete Math. 4 (1991) 550–567.

[12] A. Tamir, Comments on the paper: 'Heuristic and special case algorithms for dispersion problems' by S.S. Ravi, D.J. Rosenkrantz and G.K. Tayi, Oper. Res., to appear.