Approximation Algorithms for Maximum Coverage and Max Cut with Given Sizes of Parts*

A. A. Ageev and M. I. Sviridenko

Sobolev Institute of Mathematics
pr. Koptyuga 4, 630090, Novosibirsk, Russia
{ageev,svir}@math.nsc.ru

Abstract. In this paper we demonstrate a general method of designing constant-factor approximation algorithms for some discrete optimization problems with cardinality constraints. The core of the method is a simple deterministic ("pipage") procedure of rounding of linear relaxations. By using the method we design a $(1-(1-1/k)^k)$-approximation algorithm for the maximum coverage problem where k is the maximum size of the subsets that are covered, and a $1/2$-approximation algorithm for the maximum cut problem with given sizes of parts in the vertex set bipartition. The performance guarantee of the former improves on that of the well-known $(1-e^{-1})$-greedy algorithm due to Cornuejols, Fisher and Nemhauser in each case of bounded k. The latter is, to the best of our knowledge, the first constant-factor algorithm for that version of the maximum cut problem.

1 Introduction

It is a fact of the present day that rounding of linear relaxations is one of the most effective techniques in designing approximation algorithms with proven worst-case bounds for discrete optimization problems. The quality characteristics of a rounding-based algorithm are highly dependent on the choice of an integer program reformulation and a rounding scheme. When applying popular random roundings one encounters the additional problem of derandomization. This problem may prove to be extremely difficult or quite intractable. For example, the widely known derandomization method of conditional probabilities [1] succeeds, as is easily seen, only under some very special conditions; in particular, if the relaxation has a subset of active variables that determine the optimal values of the remaining ones and the optimization problem with respect to these active variables is unconstrained. If one adds even a single cardinality constraint connecting the active variables, the method fails. In this paper we present a simple deterministic ("pipage") rounding method to tackle some problems of this sort. So far the method has happened to be applicable to two well-known

* This research was partially supported by the Russian Foundation for Basic Research, grant 97-01-00890

problems. By using it we design a $(1-(1-1/k)^k)$-approximation algorithm for the maximum coverage problem where k is the maximum size of the subsets, and a 1/2-approximation algorithm for the maximum cut problem with given sizes of parts in the vertex set bipartition. The performance guarantee of the former improves on that of the well-known $(1-e^{-1})$- greedy algorithm due to Cornuejols, Fisher and Nemhauser [2] in each case of bounded k. The latter is, to the best of our knowledge, the first constant-factor algorithm for that version of the maximum cut problem. In Sections 4 and 5 we show that both algorithms can be extended to apply to more general problems — the maximum k-cut problem with given sizes of parts and the maximum coverage problem with a knapsack constraint — with preservation of the same performance guarantees in both cases.

To elucidate the key ideas behind the method we describe it informally under some general assumptions (of course, these are far from most general).

Assume that the problem under consideration can be formulated as the following nonlinear binary program:

$$\max F(x_1, \dots, x_n) \tag{1}$$

$$\text{s. t.} \quad \sum_{i=1}^{n} x_i = p \tag{2}$$

$$x_i \in \{0,1\}, \quad i = 1, \dots, n \tag{3}$$

where p is a positive integer, $F(x_1, \dots, x_n)$ is a function defined on the rational points of the n-dimensional cube $[0,1]^n$ and computable in polynomial time. Assume further that one can associate with $F(x_1, \dots, x_n)$ another function $L(x_1, \dots, x_n)$ which is defined and polynomially computable on the same set, coincides with $F(x_1, \dots, x_n)$ on binary vectors, and the problem of maximizing $L(x_1, \dots, x_n)$ over all rational points of $[0,1]^n$ subject to (2) is polynomially solvable. We shall call such a problem a *nice relaxation*. In our algorithms each nice relaxation will be in fact a slight reformulation of a linear program. Assume that additionally the following properties hold:

(A) there exists $C > 0$ such that $F(x_1, \dots, x_n) \geq CL(x_1, \dots, x_n)$ for each $x \in [0,1]^n$;

(B) the function $\varphi(\varepsilon, x, i, j) = F(x_1, \dots, x_i + \varepsilon, \dots, x_j - \varepsilon, \dots, x_n)$ is convex with respect to $\varepsilon \in [-\min\{x_i, 1-x_j\}, \min\{1-x_i, x_j\}]$ for each pair of indices i and j and each $x \in [0,1]^n$.

We claim that under the above assumptions one can find in polynomial time a feasible solution $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_n)$ to the problem (1)–(3), satisfying $F(\tilde{x}) \geq CF(x^*)$ where x^* is an optimal solution to (1)–(3). Let x' be an optimal solution to the nice relaxation. Indeed, if the vector x' is not binary, then due to (2) it has at least two different components x_i and x_j with values lying strictly between 0 and 1. By property (B), $\varphi(\varepsilon, x', i, j) \geq F(x')$ either for $\varepsilon = \min\{1 - x_i, x_j\}$, or for $\varepsilon = -\min\{x_i, 1-x_j\}$. Thus we obtain a new feasible solution $x'' = (x'_1, \dots, x'_i + \varepsilon, \dots, x'_j - \varepsilon, \dots, x'_n)$ with smaller number of noninteger components and such that $F(x'') \geq F(x')$. After repeating this "pipage" step at most n times we arrive at a binary feasible solution $\tilde{x}$ with $F(\tilde{x}) \geq CL(x') \geq CL(x^*) = CF(x^*)$, as required. Since an optimal solution to the nice relaxation can be found in polynomial time, the overall running time of the above described C-approximation algorithm is polynomially bounded.

2 The maximum coverage problem

In the maximum coverage problem (MCP for short), given a family $\mathcal{F} = \{S_j : j \in J\}$ of subsets of a set $I = \{1, \ldots, n\}$ with associated nonnegative weights w_j and a positive integer p, it is required to find a subset $X \subseteq I$ with $|X| = p$ so as to maximize the total weight of the sets in $\mathcal{F}$ having nonempty intersections with X. The polynomial-time solvability of MCP clearly implies that of the set cover problem and so it is NP-hard. In a sense MCP can be treated as an inverse of the set cover problem and like the latter has numerous applications (see, e. g. [10]). It is well known that a simple greedy algorithm solves MCP approximately within a factor of $1-(1-1/p)^p$ of the optimum (Cornuejols, Fisher and Nemhauser [2]). Feige [4] proves that no polynomial algorithm can have better performance guarantee provided that P$\neq$NP. Another result concerning MCP is due to Cornuejols, Nemhauser and Wolsey [3] who prove that the greedy algorithm almost always finds an optimal solution to MCP in the case of two-element sets. We show below that MCP can be solved in polynomial time approximately within a factor of $1-(1-1/k)^k$ of the optimum, where $k = \max\{|S_j| : j \in J\}$. Although $1-(1-1/k)^k$ like $1-(1-1/p)^p$ can be arbitrary close to $1-e^{-1}$, the parameter k looks more interesting: for each fixed k ($k = 2, 3, \ldots$) MCP still remains NP-hard. E. g., in the case when $k = 2$, which is the converse of the vertex cover problem, the performance guarantee of the greedy algorithm has the same value of $1-e^{-1}$ [3], whereas our algorithm finds a solution within a factor of 3/4 of the optimum. Ultimately, the performance guarantee of our algorithm beats the performance guarantee of the greedy algorithm in each case of bounded k and coincides with that when k is unbounded. Note also that our result is similar in a sense to the well-known result [9] that the set cover problem can be approximated in polynomial time within a factor of r of the optimum, where r is the maximum number of sets containing an element.

Let $J = \{1, \ldots m\}$. MCP can be equivalently reformulated as a constrained version of MAX SAT over variables $y_1, \ldots, y_n$ with m clauses $C_1, \ldots, C_m$ such that C_j is the collection of y_i with $i \in S_j$ and has weight w_j. It is required to assign "true" values to exactly p variables so as to maximize the total weight of satisfied clauses. Furthermore, analogously to MAX SAT (see, e. g. [6]), MCP can be stated as the following integer program:

$$\max \sum_{j=1}^{m} w_j z_j \tag{4}$$

$$\text{s. t.} \quad \sum_{i \in S_j} x_i \geq z_j, \quad j = 1, \ldots, m, \tag{5}$$

$$\sum_{i=1}^{n} x_i = p, \tag{6}$$

$$x_i \in \{0,1\}, \quad i = 1, \dots, n, \tag{7}$$

$$0 \le z_i \le 1, \quad i = 1, \dots, m. \tag{8}$$

It is easy to see that the relation "$x_i = 1$ if $i \in X$, and $x_i = 0$ otherwise" establishes a 1-1 correspondence between the optimal sets in MCP and the optimal solutions to (4)–(8). Note that the variables x_i determine the optimal values of z_j in any optimal solution and represent active variables in the sense above. Moreover, it is clear that MCP is equivalent to maximizing the function $F(x) = \sum_{j=1}^{m} w_j \big(1 - \prod_{i \in S_j}(1 - x_i)\big)$ over all binary vectors x satisfying (6). Observe also that the objective function (4) can be replaced by the function

$$L(x) = \sum_{j=1}^{m} w_j \min\{1, \sum_{i \in S_j} x_i\}$$

of the active variables $x_1, \dots, x_n$, thus providing a nice relaxation.

We now show that the functions F and L just defined satisfy properties (A) and (B).

Property (A) holds with $C = (1 - (1 - 1/k)^k)$ where $k = \max\{|S_j| : j \in J\}$, which is implied by the following inequality (used first by Goemans and Williamson [6] in a similar context):

$$1 - \prod_{i=1}^{k}(1 - y_i) \ge (1 - (1 - 1/k)^k) \min\{1, \sum_{i=1}^{k} y_i\}, \tag{9}$$

valid for all $0 \le y_i \le 1$, $i = 1, \dots, k$. To make the paper self-contained we derive it below. By using the arithmetic-geometric mean inequality we have that

$$1 - \prod_{i=1}^{k}(1 - y_i) \ge 1 - \left(1 - \frac{z}{k}\right)^k,$$

where $z = \min\{1, \sum_{i=1}^{k} y_i\}$. Since the function $g(z) = 1 - (1 - z/k)^k$ is concave on the segment $[0, 1]$ and $g(0) = 0$, $g(1) = 1 - (1 - 1/k)^k$, we finally obtain

$$g(z) \ge \left(1 - (1 - 1/k)^k\right) z,$$

as desired.

To check property (B) it suffices to observe that in this case the function $\varphi(\varepsilon, x, i, j)$ is convex because it is a quadratic polynomial in ε, whose main coefficient is nonnegative for each pair of indices i and j and each $x \in [0, 1]^n$. Thus by concretizing the general scheme described in the introduction we obtain a $(1 - (1 - 1/k)^k)$-approximation algorithm for the maximum coverage problem.

We now demonstrate that the integrality gap of (4)–(8) can be arbitrarily close to $(1 - (1 - 1/k)^k)$ and thus the rounding scheme described above is best possible for the integer program (4)–(8). Set $n = kp$, $w_j = 1$ for all j and let $\mathcal{F}$ be the collection of all subsets of $\{1, \ldots, n\}$ with cardinality k. Then, by symmetry, any binary vector with exactly p units maximizes $L(x)$ subject to (6)–(7) and so the optimal value of this problem is equal to $L^* = C_n^k - C_{n-p}^k$. On the other hand, the vector with all components equal to $1/k$ provides an optimal solution of weight $L' = C_n^k$ to the linear relaxation in which the objective is to maximize $L(x)$ subject to (6) and $0 \leq x_i \leq 1$ for all i. Now it is easy to derive an upper bound on the ratio

$$\begin{aligned}
\frac{L^*}{L'} &= \frac{C_n^k - C_{n-p}^k}{C_n^k} \\
&= 1 - \frac{(n-p)!}{k!(n-p-k)!} \frac{k!(n-k)!}{n!} \\
&= 1 - \Big(\frac{n-p}{n}\Big)\Big(\frac{n-p-1}{n-1}\Big) \ldots \Big(\frac{n-p-k+1}{n-k+1}\Big) \\
&\leq 1 - \Big(\frac{n-p}{n}\Big)\Big(\frac{n-p-1}{n}\Big) \ldots \Big(\frac{n-p-k+1}{n}\Big) \\
&= 1 - \Big(1 - \frac{1}{k}\Big)\Big(1 - \frac{1}{k} - \frac{1}{n}\Big)\Big(1 - \frac{1}{k} - \frac{2}{n}\Big) \ldots \Big(1 - \frac{1}{k} - \frac{k+1}{n}\Big) \\
&\leq 1 - \Big(1 - \frac{1}{k} - \frac{k+1}{n}\Big)^k ,
\end{aligned}$$

which tends to $(1 - (1 - 1/k)^k)$ when k is fixed and $n \to \infty$.

Remark 1. The algorithm and the argument above can be easily adopted to yield the same performance guarantee in the case of the more general problem in which the constraint (6) is replaced by the constraints

$$\sum_{i \in I_t} x_i = p_t, \quad t = 1, \ldots, r$$

where $\{I_t : t = 1, \ldots, r\}$ is a partition of the ground set I and p_t are positive integers. It can be shown, on the other hand, that the worst-case ratio of the straightforward extension of the greedy algorithm cannot be lower bounded by any absolute constant. So, our algorithm is the only algorithm with constant performance guarantee among those known for this generalization.

Remark 2. It can be easily observed that from the very beginning (and with the same ultimate result) we could consider objective functions of the following more general form:

$$F(x) = \sum_{j=1}^{m} w_j \big(1 - \prod_{i \in S_j} (1 - x_i)\big) + \sum_{t=1}^{l} u_t \big(1 - \prod_{i \in R_t} x_i\big),$$

where S_j and R_t are arbitrary subsets of $\{1, \ldots, n\}$, and u_t, w_j are nonnegative weights. The problem with such objective functions can be reformulated as the constrained MAX SAT in which each clause either contains no negations or contains nothing but negations.

3 The maximum cut problem with given sizes of parts

Let G be a complete undirected graph with vertex set $V(G) = \{1, \dots, n\}$. Any nonempty vertex subset $X \subseteq V(G)$ determines a cut $\delta(X)$ of G which, by definition, consists of the set of edges having exactly one end in X. Assume that to each edge $e = ij$ of G is assigned a nonnegative weight w_{ij}. In the maximum cut problem with given sizes of parts (MCGS for short), given a complete graph G, nonnegative edge weights w_{ij} and a positive integer $p \leq n/2$, it is required to find a cut $\delta(X)$ having maximum total weight over all cuts with $|X| = p$. In the special case of $p = n/2$, also known as the max bisection problem, an approximate solution of weight within a factor of $1/2$ of the optimum can be found in polynomial time by a simple random sampling algorithm. However, natural extensions of this algorithm to the case of arbitrary p do not admit any fixed constant performance guarantee. Frieze and Jerrum [5] prove that the max bisection problem can be approximately solved within a factor of 0.65 of the optimum by a randomized rounding of an optimal solution to a semidefinite program but their algorithm does not admit straightforward extensions. We present an approximation algorithm which finds a feasible solution to MCGS of weight within a factor of $1/2$ of the optimum in the case of arbitrary p.

Observe first that MCGS is equivalent to maximizing the function

$$F(x) = \sum_{i<j} w_{ij}(x_i + x_j - 2x_i x_j)$$

over all binary vectors x with exactly p units. Notice first that the function F has property (B) because of the same reason as in the above section. Consider the following integer program:

$$\max \sum_{i<j} w_{ij} z_{ij} \tag{10}$$

$$\text{s. t.} \quad z_{ij} \leq x_i + x_j, \quad i < j, \tag{11}$$

$$z_{ij} \leq 2 - x_i - x_j, \quad i < j, \tag{12}$$

$$\sum_{i \in V(G)} x_i = p, \tag{13}$$

$$x_i, z_{ij} \in \{0, 1\}, \quad i \in V(G),\ i < j. \tag{14}$$

It is easy to check that $X^* \subseteq V(G)$ provides an optimal cut in MCGS if and only if the vector x defined by $x_i^* = 1$ if $i \in X^*$ and 0 otherwise, is an optimal solution to (10)–(14). Note that again, variables x_i can be considered as active in the sense that the program (10)–(14) is in fact equivalent to

$$\max \quad L(x) = \sum_{i<j} w_{ij} \min\{x_i + x_j, 2 - x_i - x_j\} \tag{15}$$

$$\text{s. t.} \quad \sum_{i \in V(G)} x_i = p, \tag{16}$$

$$x_i \in \{0, 1\}, \quad i \in V(G). \tag{17}$$

Here we take as a nice relaxation the continuous relaxation of (15)–(17) (in which (17) is replaced by $0 \le x_i \le 1$ for all $i \in V(G)$). Let x be an optimal solution to the nice relaxation. We claim that $F(x) \ge 1/2L(x)$. Indeed, it suffices to check that

$$x_i + x_j - 2x_i x_j \ge 1/2 \min\{x_i + x_j, 2 - x_i - x_j\} \tag{18}$$

for all pairs i, j with $i < j$. Fix some $i < j$ and assume first that $x_i + x_j \le 1$. Then (18) is equivalent to $x_i + x_j \ge 4x_i x_j$, which follows from

$$x_i + x_j \ge (x_i + x_j)^2 \ge 4x_i x_j.$$

The case when $x_i + x_j \ge 1$ is quite symmetric to the above, which is easily shown by the substitution $x_i = 1 - y_i$, $x_j = 1 - y_j$. Thus property (A) with $C = 1/2$ does hold and we obtain an 1/2-approximation algorithm for MCGS.

Consider an instance of MCGS in which $2p \le n$, $w_{ij} = 1$ if both i and j lie in a fixed subset of $V(G)$ having cardinality $2p$, and $w_{ij} = 0$ otherwise. The optimal value of this instance is obviously equal to p^2, whereas the optimal value of the linear relaxation is $p(p-1)/2$. Hence

$$\frac{p^2}{p(p-1)/2} = \frac{2p^2}{p^2 - p} = \frac{2}{1 - 1/p}$$

which can be arbitrarily close to 2. Thus, again, our rounding scheme is best possible for the integer program (10)–(14).

4 The maximum k-cut problem with given sizes of parts

In this section we illustrate how the "pipage" technique can be extended to apply to a natural generalization of MCGS.

In the maximum k-cut problem with given sizes of parts (k-MCGS), given a complete undirected graph G with $V(G) = \{1, \dots, n\}$, nonnegative edge weights w_{ij}, and a collection of nonnegative numbers $\{p_1, \dots, p_k\}$, it is required to find a partition $\{P_1, \dots, P_k\}$ of $V(G)$ with $|P_t| = p_t$ for each t, maximizing the function

$$\sum_{r<s} \sum_{i \in P_r} \sum_{j \in P_s} w_{ij}.$$

Two similar problems have been considered earlier by Guttmann-Beck and Hassin in [7] and [8]. The minimization version of k-MCGS, which is also NP-hard, is studied in [8]. No constant-factor algorithm for this problem is known. Guttmann-Beck and Hassin present a polynomial-time 3-approximation algorithm for the special case when the edge weights satisfy triangle inequality and the number of parts is fixed. The other paper [7] considers the min-sum p-clustering problem which can be treated as a complement to k-MCGS. The authors suggest a 2-approximation algorithm for the similar special case of this problem.

We present further an approximation algorithm which produces a feasible solution to k-MCGS of weight within a factor of $1/2$ of the optimum. The problem can be reformulated as the following nonlinear integer program:

$$\max \quad F(x) = \sum_{i<j} w_{ij}(1 - \sum_{t=1}^{k} x_{it}x_{jt}) \tag{19}$$

$$\text{s. t.} \quad \sum_{i=1}^{n} x_{it} = p_t, \quad t = 1, \ldots k, \tag{20}$$

$$\sum_{t=1}^{k} x_{it} = 1, \quad i \in V(G), \tag{21}$$

$$x_{it} \in \{0,1\}, \quad i \in V(G). \tag{22}$$

It is easy to check that x^*_{it} is an optimal solution to (19)–(22) if and only if $\{P_1, \ldots, P_k\}$ with $P_t = \{i : x_{it} = 1\}$ for each t, is an optimal partition in k-MCGS.

Consider now the following linear integer program also equivalent to k-MCGS and virtually generalizing (10)–(14):

$$\max \quad \frac{1}{2} \sum_{i<j} w_{ij} \sum_{t=1}^{k} z^t_{ij} \tag{23}$$

$$\text{s. t.} \quad z^t_{ij} \le x_{it} + x_{jt}, \quad i < j, \quad t = 1, \ldots, k, \tag{24}$$

$$z^t_{ij} \le 2 - x_{it} - x_{jt}, \quad i < j, \quad t = 1, \ldots, k, \tag{25}$$

$$\sum_{i=1}^{n} x_{it} = p_t, \quad t = 1, \ldots, k, \tag{26}$$

$$\sum_{t=1}^{k} x_{it} = 1, \quad i \in V(G), \tag{27}$$

$$x_{it}, z^t_{ij} \in \{0,1\}, \quad i,j \in V(G), \quad t = 1, \ldots, k. \tag{28}$$

As in the previous section the objective function (23) can be replaced by

$$L(x) = \frac{1}{2} \sum_{i<j} w_{ij} \sum_{t=1}^{k} \min\{x_{it} + x_{jt}, 2 - x_{it} - x_{jt}\}.$$

We claim that $F(x) \ge 1/2L(x)$ for all feasible $x = (x_{it})$. Fix a pair of indices i and j with $i < j$. It suffices to show that

$$1 - \sum_{t=1}^{k} x_{it}x_{jt} \ge 1/4 \sum_{t=1}^{k} \min\{x_{it} + x_{jt}, 2 - x_{it} - x_{jt}\}. \tag{29}$$

Indeed, we have already proved in the above section (see (18)) that for each t

$$x_{it} + x_{jt} - 2x_{it}x_{jt} \ge 1/2 \min\{x_{it} + x_{jt}, 2 - x_{it} - x_{jt}\}. \tag{30}$$

Adding together (30) over t from 1 to k and taking into account (27) we obtain that

$$2 - 2\sum_{t=1}^{k} x_{it}x_{jt} \geq 1/2\sum_{t=1}^{k} \min\{x_{it} + x_{jt}, 2 - x_{it} - x_{jt}\},$$

which implies (29).

We describe now the "pipage" step. Let x be an optimal solution to the linear relaxation of (23)–(28) (that is, with (28) replaced by $0 \leq x_{jt} \leq 1$, $0 \leq z_{ij}^{t} \leq 1$). Define the bipartite graph H with the bipartition $(\{1,\dots,n\},\{1,\dots,k\})$ so that $jt \in E(H)$ if and only if x_{jt} is fractional. Note that (26) and (27) imply that each vertex of H is either isolated or has degree at least 2. Assume that x has fractional components. Since H is bipartite it follows that H has a circuit C of even length. Let M_1 and M_2 be the matchings of H whose union is the circuit C. Define a new solution $x(\varepsilon)$ by the following rule: if jt is not an edge of C, then $x_{jt}(\varepsilon)$ coincides with x_{jt}, otherwise, $x_{jt}(\varepsilon) = x_{jt} + \varepsilon$ if $jt \in M_1$, and $x_{jt}(\varepsilon) = x_{jt} - \varepsilon$ if $jt \in M_2$.

By definition $x(\varepsilon)$ is a feasible solution to the linear relaxation of (23)–(28) for all $\varepsilon \in [-\varepsilon_1, \varepsilon_2]$ where

$$\varepsilon_1 = \min\{\min_{jt\in M_1} x_{jt}, \min_{jt\in M_2}(1 - x_{jt})\}$$

and

$$\varepsilon_2 = \min\{\min_{jt\in M_1}(1 - x_{jt}), \min_{jt\in M_2} x_{jt}\}.$$

Moreover, $F(x(\varepsilon))$ is a quadratic polynomial in ε with a nonnegative main coefficient and therefore

$$F(x(\varepsilon^*)) \geq F(x) \geq 1/2L(x)$$

for some $\varepsilon^* \in \{-\varepsilon_1, \varepsilon_2\}$. The new solution $x(\varepsilon^*)$, being feasible for the linear relaxation of (23)–(28), has a smaller number of fractional components. Set $x' = x(\varepsilon^*)$ and, if x' has fractional components, apply to x' the above described "pipage" step and so on. Ultimately, after at most nk steps, we arrive at a solution $\tilde{x}$ which is feasible for (19)–(22) and satisfies

$$F(\tilde{x}) \geq 1/2L(x) \geq 1/2F^*,$$

where F^* is an optimal value of (19)–(22) (and of the original instance of k-MCGS). Thus we have designed an 1/2-approximation algorithm for k-MCGS.

5 The maximum coverage problem with a knapsack constraint.

In this section we show that the "pipage" rounding of linear relaxations in conjunction with the partial enumeration method developed by Sahni [12] for the knapsack problem can yield good constant-factor approximation algorithms even in the case of a knapsack constraint.

The maximum coverage problem with a knapsack constraint (MCKP) is, given a family $\mathcal{F} = \{S_j : j \in J\}$ of subsets of a set $I = \{1, \ldots, n\}$ with associated nonnegative weights w_j and costs c_j, and a positive integer B, to find a subset $X \subseteq I$ with $\sum_{j \in X} c_j \leq B$ so as to maximize the total weight of the sets in $\mathcal{F}$ having nonempty intersections with X. MCKP includes both MCP and the knapsack problem as special cases. Wolsey [13] appears to be the first who succeeded in constructing a constant-factor approximation algorithm for MCKP even in a more general setting with an arbitrary nondecreasing submodular objective function. His algorithm is of greedy type and has performance guarantee of $1 - e^{-\beta} \approx 0.35$ where β is the root of the equation $e^{\beta} = 2 - \beta$. Recently, Khuller, Moss and Naor [11] have designed a $(1 - e^{-1})$-approximation algorithm for MCKP by combining the partial enumeration method of Sahni for the knapsack problem [12] with a simple greedy procedure. In this section we present an $1 - (1 - 1/k)^k$-approximation algorithm for MCKP where k is the maximum size of sets in the instance. Our algorithm exploits the same idea of partial enumeration but instead finding a greedy solution, solves a linear relaxation and then rounds the fractional solution by a bit more general "pipage" procedure.

Generalizing (4)–(8) rewrite MCKP as the following integer program:

$$\max \sum_{j=1}^{m} w_j z_j \tag{31}$$

$$\text{s. t.} \quad \sum_{i \in S_j} x_i \geq z_j, \quad j \in J, \tag{32}$$

$$\sum_{i=1}^{n} c_i x_i \leq B, \tag{33}$$

$$0 \leq x_i, z_j \leq 1, \quad i \in I, j \in J \tag{34}$$

$$x_i \in \{0, 1\}, \quad i \in I. \tag{35}$$

Note that one can exclude variables z_j by rewriting (31)–(35) as the following equivalent nonlinear program:

$$\max \quad F(x) = \sum_{j=1}^{m} w_j \Big(1 - \prod_{i \in S_j} (1 - x_i)\Big) \tag{36}$$

subject to (33)–(35).

Set $k = \max\{|S_j| : j \in J\}$. Denote by $IP[I_0, I_1]$ and $LP[I_0, I_1]$ the integer program (31)–(35) and its linear relaxation (31)–(34) respectively, subject to the additional constraints: $x_i = 0$ for $i \in I_0$ and $x_i = 1$ for $i \in I_1$ where I_0 and I_1 are disjoint subsets of I. By a solution to $IP[I_0, I_1]$ and $LP[I_0, I_1]$ we shall mean only a vector x, since the optimal values of z_j are trivially computed if x is fixed.

We first describe an auxiliary algorithm $\mathcal{A}$ which finds a feasible solution $x^{\mathcal{A}}$ to the linear program $LP[I_0, I_1]$. The algorithm is divided into two phases.

The first phase consists in finding an optimal solution x^{LP} to $LP(I_0, I_1)$ by application one of the known polynomial-time algorithms. The second phase of $\mathcal{A}$ transforms x^{LP} into $x^{\mathcal{A}}$ through a series of "pipage" steps. We now describe the general step. Set $x^{\mathcal{A}} \leftarrow x^{LP}$. If at most one component of $x^{\mathcal{A}}$ is fractional, stop. Otherwise, choose two indices i' and i'' such that $0 < x^{\mathcal{A}}_{i'} < 1$ and $0 < x^{\mathcal{A}}_{i''} < 1$. Set $x^{\mathcal{A}}_{i'}(\varepsilon) \leftarrow x^{\mathcal{A}}_{i'} + \varepsilon$, $x^{\mathcal{A}}_{i''}(\varepsilon) \leftarrow x^{\mathcal{A}}_{i''} - \varepsilon c_{i'}/c_{i''}$ and $x^{\mathcal{A}}_k(\varepsilon) \leftarrow x^{\mathcal{A}}_k$ for all $k \neq i', i''$. Find an endpoint ε^* of the interval

$$[-\min\{x_{i'}, (1 - x_{i''})\frac{c_{i''}}{c_{i'}}\}, \min\{1 - x_{i'}, x_{i''}\frac{c_{i''}}{c_{i'}}\}].$$

such that $F(x(\varepsilon^*)) \geq F(x^{\mathcal{A}})$. Set $x^{\mathcal{A}} \leftarrow x(\varepsilon^*)$. Go to the next step.

The correctness of the second phase follows from the fact that the vector $x(\varepsilon)$ is feasible for each ε in the above interval, and from the earlier observation (see Section 2) that the function $F(x(\varepsilon))$ is convex.

Each "pipage" step of the second phase of $\mathcal{A}$ reduces the number of fractional components of the current vector $x^{\mathcal{A}}$ at least by one. So, finally, $\mathcal{A}$ outputs an "almost integral" feasible vector $x^{\mathcal{A}}$ having at most one fractional component.

By construction, $F(x^{\mathcal{A}}) \geq F(x^{LP})$. By (9), it follows that

$$\begin{aligned} F(x^{\mathcal{A}}) &\geq F(x^{LP}) \\ &\geq \sum_{j \in J_1} w_j + \left(1 - (1 - 1/k)^k\right) \sum_{j \in J \setminus J_1} w_j \min\{1, \sum_{i \in S_j} x^{LP}_i\} \end{aligned} \tag{37}$$

where and henceforth $J_1 = \{j : S_j \cap I_1 \neq \emptyset\}$.

We now present a description of the whole algorithm.

Input: An instance of the integer program (31)–(35);
Output: A feasible solution $\overline{x}$ to the instance;

Among all feasible solutions x to the instance satisfying $\sum_{i \in I} x_i \leq 3$, by complete enumeration find a solution x^0 of maximum weight;
$\overline{x} \leftarrow x^0$;
for all $I_1 \subset I$ such that $|I_1| = 4$ and $\sum_{i \in I_1} c_i \leq B$ **do**
  begin
    $I_0 \leftarrow \emptyset$;
    $t \leftarrow 0$;
    while $t = 0$ **do**
      begin
        apply $\mathcal{A}$ to $LP[I_0, I_1]$;
        if all $x^{\mathcal{A}}_i$ are integral
        then begin $t \leftarrow 1$; $\hat{x} \leftarrow x^{\mathcal{A}}$ **end**
        otherwise
          begin
            find i' such that $x^{\mathcal{A}}_{i'}$ is fractional;
            $\hat{x}_{i'} \leftarrow 0$;
            for all $i \neq i'$ **do** $\hat{x}_i \leftarrow x^{\mathcal{A}}_i$;
            $I_0 \leftarrow I_0 \cup \{i'\}$

end
if $F(\hat{x}) > F(\overline{x})$ **then** $\overline{x} \leftarrow \hat{x}$
end
end

We now prove that the performance guarantee of the described algorithm is indeed $(1-(1-1/k)^k$.

Let X^* be an optimal set of the given instance of MCKP. Denote by x^* the incidence vector of X^*. Recall that x^* is an optimal solution to the equivalent nonlinear program (36), (33)–(35). If $|X^*| \le 3$, then the output of the algorithm is an optimal solution. So we may assume that $|X^*| \ge 4$. W.l.o.g. we may also assume that the set I is ordered in such a way that $X^* = \{1, \dots, |X^*|\}$ and for each $i \in X^*$, the element i covers the sets in $\mathcal{F}$ not covered by the elements $1, \dots, i-1$ of the maximum total weight among $j \in X^* \setminus \{1, \dots, i-1\}$.

Consider now that iteration of the algorithm at which $I_1 = \{1, 2, 3, 4\}$. At this iteration the algorithm runs through q steps, $q \le n-4$. At step t it calls the algorithm $\mathcal{A}$ to find an "almost integral" feasible solution $x^t = x^{\mathcal{A}}$ to $IP[I_0^t, I_1]$ where $I_0^t = \{i_1, \dots, i_{t-1}\}$. If all components of x^t are integral then $t = q$ and the iteration ends up. Otherwise the vector x^t is replaced by the integral vector $\hat{x}^t$ which coincides with x^t in its integral components and equals zero in its single fractional component indexed by i_t. If the weight of $\hat{x}^t$ exceeds that of the record solution, the latter is updated. Then the algorithm sets $I_0^{t+1} = I_0^t \cup \{i_t\}$ and goes to the next step. Thus at the iteration the algorithm finds a series of feasible integral solutions $\hat{x}^1, \dots, \hat{x}^q$ to (36), (33)–(35) or, equivalently, subsets $\hat{X}_1, \dots, \hat{X}_q \subseteq I$ satisfying $\hat{X}_t \supseteq I_1$ and $\hat{X}_t \cap I_0^t = \emptyset$ where $I_0^t = \{i_1, \dots, i_{t-1}\}$, $t = 1, \dots, q$.

Assume first that $X^* \cap I_0^q = \emptyset$. Then x^* is an optimal solution to $IP[I_0^q, I_1]$. Recall that $\hat{x}^q = x^q$ and the latter is obtained from the optimal solution to $LP[I_0^q, I_1]$ by the "pipage" process. By (37) it follows that the solution $\hat{x}^q$, being not better than the output solution, has weight within a factor of $(1-(1-1/k)^k)$ of the optimum. Thus, in this case we are done.

Assume now that $X^* \cap I_0^q \neq \emptyset$ and let I_0^{s+1} be the first set in the series $I_0^1 = \emptyset, \dots, I_0^q$, having nonempty intersection with X^*. In other words, i_s is the first index in the series $i_1, \dots, i_{q-1}$ lying in X^*. We claim then that

$$F(\hat{x}^s) \ge \big(1-(1-1/k)^k\big)F(x^*). \tag{38}$$

Since the weight of $\hat{x}^s$ does not exceed the weight of the output vector of the algorithm this would prove that $(1-(1-1/k)^k)$ is the performance guarantee of the algorithm.

In the following argument for brevity we shall use alternately the sets and their incidence vectors as the arguments of F.

Indeed, the function F can be also treated as the set function $F(X)$ defined on all subsets $X \subseteq I$. It is well known that $F(X)$ is submodular and, consequently, have the property that

$$F(X \cup \{i\}) - F(X) \ge F(X \cup Y \cup \{i\}) - F(X \cup Y), \tag{39}$$

for all $X, Y \subseteq I$ and $i \in I$. Let $i \in I$ and $Y \supseteq I_1$. Then

$$\begin{aligned}
1/4F(I_1) = 1/4F(\{1,2,3,4\}) &= 1/4\big(F(\{1,2,3,4\}) - F(\{1,2,3\}) + \\
&\quad F(\{1,2,3\}) - F(\{1,2\}) + \\
&\quad F(\{1,2\}) - F(\{1\}) + \\
&\quad F(\{1\}) - F(\emptyset)\big) \\
\big(\text{by the choice of } I_1\ \big) & \\
&\geq 1/4\big(F(\{1,2,3,i\}) - F(\{1,2,3\}) + \\
&\quad F(\{1,2,i\}) - F(\{1,2\}) + \\
&\quad F(\{1,i\}) - F(\{1\}) + \\
&\quad F(\{i\}) - F(\emptyset)\big) \\
\big(\text{by } (39)\big) & \\
&\geq \big(F(Y \cup \{i\}) - F(Y)\big).
\end{aligned}$$

Thus, we have proved that for any $Y \supseteq I_1$ and any $i \in I$,

$$1/4F(I_1) \geq F(Y \cup \{i\}) - F(Y). \tag{40}$$

Recall that the integral vector $\hat{x}^s$ is obtained from an "almost integral" vector x^s returned by the algorithm $\mathcal{A}$, by the replacement with zero its single fractional component $x^s_{i_s}$. It follows that

$$F(\hat{X}_s \cup \{i_s\}) \geq F(x_s). \tag{41}$$

Let x^{LP}, z^{LP} denote an optimal solution to $LP[I_0^s, I_1]$. Using (35), (40) and (41) we finally obtain

$$\begin{aligned}
F(\hat{x}^s) &= F(\hat{X}_s) \\
&= F(\hat{X}_s \cup \{i_s\}) - \big(F(\hat{X}_s \cup \{i_s\}) - F(\hat{X}_s)\big) \\
\big(\text{by } (40)\big) & \\
&\geq F(\hat{X}_s \cup \{i_s\}) - 1/4F(I_1) \\
\big(\text{by } (41)\big) & \\
&\geq\ F(x^s) - 1/4F(I_1) \\
&= \sum_{j \in J_1} w_j + \sum_{j \in J \setminus J_1} w_j \big(1 - \prod_{i \in S_j} (1 - x_i^s)\big) - 1/4 \sum_{j \in J_1} w_j \\
(\text{by } (37)) & \\
&\geq 3/4 \sum_{j \in J_1} w_j + \big(1 - (1 - 1/k)^k\big) \sum_{j \in J \setminus J_1} w_j \min\{1, \sum_{i \in S_j} x_i^{LP}\} \\
&\geq \big(1 - (1 - 1/k)^k\big) \Big(\sum_{j \in J_1} w_j + \sum_{j \in J \setminus J_1} w_j \min\{1, \sum_{i \in S_j} x_i^{LP}\} \Big)
\end{aligned}$$

(by the choice of s)

$$\geq \big(1-(1-1/k)^k\big)\Big(\sum_{j\in J_1} w_j + \sum_{j\in J\setminus J_1} w_j \min\{1, \sum_{i\in S_j} x_i^*\}\Big)$$
$$= \big(1-(1-1/k)^k\big)F(x^*).$$

Acknowledgement

The authors wish to thank Refael Hassin for helpful comments and for pointing out some references.

References

1. N. Alon and J. H. Spenser, The probabilistic method, John Wiley and Sons, New York, 1992.
2. G. Cornuejols, M. L. Fisher and G. L. Nemhauser, Location of bank accounts to optimize float: an analytic study exact and approximate algorithms, Management Science 23 (1977) 789–810.
3. G. Cornuejols, G. L. Nemhauser and L. A. Wolsey, Worst-case and probabilistic analysis of algorithms for a location problem, Operations Research 28 (1980) 847–858.
4. U. Feige, A threshold of ln n for approximating set cover, J. of ACM. 45 (1998) 634–652.
5. A. Frieze and M. Jerrum, Improved approximation algorithms for MAX k-CUT and MAX BISECTION, Algorithmica 18 (1997) 67–81.
6. M. X. Goemans and D. P. Williamson, New 3/4-approximation algorithms for MAX SAT, SIAM J. Discrete Math. 7 (1994) 656–666.
7. N. Guttmann-Beck and R. Hassin, Approximation algorithms for min-sum p-clustering, Discrete Appl. Math. 89 (1998) 125–142.
8. N. Guttmann-Beck and R. Hassin, Approximation algorithms for minimum K-cut, to appear in Algorithmica.
9. D. S. Hochbaum, Approximation algorithms for the set covering and vertex cover problems, SIAM J. on Computing 11 (1982) 555–556.
10. D. S. Hochbaum, Approximating covering and packing problems: Set Cover, Vertex Cover, Independent Set, and related problems, in: D. S. Hochbaum, ed., Approximation algorithms for NP-hard problems (PWS Publishing Company, New York, 1997) 94–143.
11. S. Khuller, A. Moss and J. Naor, The budgeted maximum coverage problem, to appear in Inform. Proc. Letters.
12. S. Sahni, Approximate algorithms for the 0–1 knapsack problem, J. of ACM 22 (1975) 115–124.
13. L. A. Wolsey, Maximizing real-valued submodular functions: primal and dual heuristics for location problems, Math. Oper. Res. 7 (1982) 410–425.