Journal of Combinatorial Optimization, 8, 307–328, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Pipage Rounding: A New Method of Constructing Algorithms with Proven Performance Guarantee*

A.A. AGEEV ageev@math.nsc.ru
Sobolev Institute of Mathematics, pr. Koptyuga 4, 630090, Novosibirsk, Russia

M.I. SVIRIDENKO sviri@us.ibm.com
IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY

Received December 6, 2000; Accepted July 24, 2002

Abstract. The paper presents a general method of designing constant-factor approximation algorithms for some discrete optimization problems with assignment-type constraints. The core of the method is a simple deterministic procedure of rounding of linear relaxations (referred to as pipage rounding). With the help of the method we design approximation algorithms with better performance guarantees for some well-known problems including MAXIMUM COVERAGE, MAX CUT with given sizes of parts and some of their generalizations.

Keywords: approximation algorithm, performance guarantee, linear relaxation, rounding technique, maximum coverage, max cut

1. Introduction

Rounding of linear relaxations is one of the most effective techniques in designing approximation algorithms with proven performance guarantees. Two points are most important for those who aim at success when applying this technique: a good choice of integer program formulation and a good choice of rounding scheme for its linear relaxation. Selecting a good integer program formulation highly depends on the nature of the problem and and is often a matter of intuition. As for rounding methods, there are two basic different approaches—deterministic and random. When applying random rounding methods one encounters the additional problem of derandomization. This problem may prove to be extremely difficult or quite intractable. For example, a difficult point in implementation of random roundings is that of producing feasible solutions in the case of some simple equality constraints like a cardinality or a budget one. Though by using some tricks this difficulty could be overcome, the resulting algorithm will be too sophisticated to admit derandomization.

The main result of this paper is a simple deterministic rounding method—we call it the pipage rounding—especially oriented to tackle some problems with assignment- and budget-type constraints.

*Parts of this paper appeared in preliminary form in Proceedings of IPCO'99 (Ageev and Sviridenko, 1999) and ESA'00 (Ageev and Sviridenko, 2000).

The paper is organized as follows. In Section 2 we give a general description of the pipage method.

In Section 3 we demonstrate relatively simple but effective applications of the method. We apply the method to the maximum coverage problem. The result is the $(1-(1-1/k)^k)$-approximation algorithm for the maximum coverage problem where k is the maximum size of the subsets. The performance guarantee of $(1-(1-1/k)^k)$ improves on that of $1-e^{-1}$ in each case of bounded k, established by Cornuejols et al. (1977) for the greedy algorithm.

In Section 4 we consider the maximum k-cut problem in hypergraphs with the additional constraint that the sizes of the parts (sides) of the partition must be equal to given numbers (hypergraph max k-cut with given sizes of parts or, for short, HYP MAX k-CUT WITH GSP). By applying the pipage rounding method, we prove that HYP MAX k-CUT WITH GSP can be approximated within a factor of $\min\{\lambda_{|S|} : S \in E\}$ of the optimum, where E is the edge set of the hypergraph and $\lambda_r = 1-(1-1/r)^r-(1/r)^r$. This gives a 0.5-approximation for the case of graphs and a $(1-e^{-1})$-approximation for the case of hypergraphs with edges of size at least 3. We also show that the performance guarantee of our algorithm in the later case is best possible unless $P = NP$.

In Section 5 we present an essentially more sophisticated application of our method. We combine the pipage rounding with the partial enumeration method of Sahni (1975) to design an algorithm with a better approximation ratio for a knapsack generalization of MAXIMUM COVERAGE.

In the last section we apply the pipage rounding to a scheduling problem. In this problem, for a set of unrelated parallel machines, it is required to find a schedule minimizing the total weighted completion time provided that for each machine the number of jobs that it can process is bounded by a given number. The version without restrictions on the numbers of jobs was studied earlier by Skutella (1998, 1999) who proved that it can be approximated within a factor of 3/2 of the optimum. Applying pipage rounding we obtain that the same approximation bound also holds for the (more general) problem with the restrictions on the numbers of jobs. A distinctive feature of this application is that it deals with a minimization problem while the previous applications treated maximization ones.

2. Pipage rounding: A general description

In this section we present a general description of the rounding procedure, that underlies all the algorithms presented in this paper. The description, being far from most general, still covers the most important particular cases treated so far. We will also dwell on specific features of the scheme corresponding to some important special cases.

We start with formulating a general problem. Suppose we are given a bipartite graph $G = (U, W; E)$, a function $F(x)$ defined on the rational points $x = (x_e : e \in E)$ of the $|E|$-dimensional cube $[0, 1]^{|E|}$ and computable in polynomial time, and $p : U \cup W \to \mathbb{Z}_+$.

Consider the binary program

$$\begin{aligned} \max \quad & F(x) && (1)\\ \text{s.t.} \quad & \sum_{e\in\delta(v)} x_e \le p_v, \quad v \in U \cup W, && (2)\\ & 0 \le x_e \le 1, \quad e \in E, && (3)\\ & x_e \in \{0, 1\}, \quad e \in E. && (4) \end{aligned}$$

Denote by F^* the optimal value of (1)–(4). We call a solution x satisfying (2)–(3) *fractional* if some of its components are non-integral. Notice that the fractional relaxation (1)–(3) is not assumed to be polynomially solvable.

Let $\check{x}$ be a fractional solution to (1)–(3). We further describe a rounding algorithm, called PIPAGE, that converts $\check{x}$ into an integral vector $\bar{x}$ satisfying (2)–(4). After that we will formulate some sufficient conditions on F guaranteeing $F(\bar{x}) \ge F(\check{x})$.

Algorithm PIPAGE. PIPAGE consists of uniform steps at each of which a current fractional solution x transforms into a new solution x' with smaller number of non-integral components. We proceed to description of the general step. Let x be a current vector satisfying (2)–(4). If x is integral, then the algorithm terminates and outputs x. Suppose now that x is fractional. Construct the subgraph H_x of G with the same vertex set and the edge set E_x defined by the condition that $e \in E_x$ if and only if x_e is non-integral. If H_x contains cycles, then we set R to be such a cycle. If H_x is a forest, then we set R to be a path of H_x whose endpoints have degree 1. Since H_x is bipartite, in both cases R can be uniquely represented as the union of two matchings. Let M_1 and M_2 denote those matchings. Define a new solution $x(\varepsilon, R)$ by the following rule: if $e \in E \setminus R$, then $x_e(\varepsilon, R)$ coincides with x_e, otherwise $x_e(\varepsilon, R) = x_e + \varepsilon$ when $e \in M_1$, and $x_e(\varepsilon, R) = x_e - \varepsilon$ when $e \in M_2$. Set

$$\varepsilon_1 = \min\left\{ \min_{e\in M_1} x_e, \min_{e\in M_2}(1 - x_e)\right\}$$

and

$$\varepsilon_2 = \min\left\{ \min_{e\in M_1}(1 - x_e), \min_{e\in M_2} x_e\right\}.$$

Let $x_1 = x(-\varepsilon_1, R)$, $x_2 = x(\varepsilon_2, R)$. Set $x' = x_1$ if $F(x_2) < F(x_1)$ and $x' = x_2$ otherwise.

We call the above transformation the *rounding of x over R* .

Define $\varphi(\varepsilon, x, R) = F(x(\varepsilon, R))$.

Correctness and analysis. Consider an arbitrary step of PIPAGE. Note first that $E_{x'}$ is a proper subset of E_x and hence x' has smaller number of fractional components than x. This means that PIPAGE terminates after at most $|E|$ steps, and its output, $\bar{x}$, is integral. Recall now that by assumption each p_v is an integer and by description, if R is a path, the endpoints of R have degree 1 in H. It follows that $x(\varepsilon)$ satisfies (2)–(3) for every $\varepsilon \in [-\varepsilon_1, \varepsilon_2]$.

Let $v \in U \cup W$ and $q_v = \sum_{e\in\delta(v)} x_e$. Observe that if v is not an endpoint of R, then $\sum_{e\in\delta(v)} x_e(\varepsilon) = q_v$ for all ε. Assume that q_v is an integer. Then v cannot have degree 1 in H_x and therefore it cannot be an endpoint of R when R is a path. Consequently, $\sum_{e\in\delta(v)} x_e(\varepsilon) = q_v$ whenever $\varepsilon \in [-\varepsilon_1, \varepsilon_2]$. Assume now that q_v is non-integral. Then, by the definition of ε_1 and ε_2, $\lfloor q_v \rfloor \le \sum_{e\in\delta(v)} x_e(\varepsilon) \le \lfloor q_v \rfloor + 1$ whenever $\varepsilon \in [-\varepsilon_1, \varepsilon_2]$. Thus we have established the following relations between $\bar{x}$ and $\check{x}$.

Property 1. *If* $\sum_{e\in\delta(v)} \check{x}_e$ *is integral, then* $\sum_{e\in\delta(v)} \bar{x}_e = \sum_{e\in\delta(v)} \check{x}_e$.

Property 2. *If* $\sum_{e\in\delta(v)} \check{x}_e$ *is non-integral, then*

$$\left\lfloor \sum_{e\in\delta(v)} \check{x}_e \right\rfloor \le \sum_{e\in\delta(v)} \bar{x}_e \le \left\lfloor \sum_{e\in\delta(v)} \check{x}_e \right\rfloor + 1.$$

Since all p_v are integers, Properties 2 and 2 imply that $\bar{x}$ satisfies (2). Thus, after performing at most $|E|$ steps PIPAGE transforms $\check{x}$ into a solution $\bar{x}$ obeying (1)–(4). Since each step can be clearly implemented in polynomial time, it follows that the overall running time of the algorithm is polynomially bounded.

The pipage rounding scheme. Assume now that there is a polynomial-time algorithm for finding a fractional feasible solution $\check{x}$ to (1)–(3) such that

$$F(\check{x}) \ge CF^*, \tag{5}$$

where C is a positive constant. If, in addition, $\bar{x}$—the output of PIPAGE—satisfied

$$F(\bar{x}) \ge F(\check{x}), \tag{6}$$

we would obtain a *C-approximation algorithm* for solving (1)–(4). The inequality (6) clearly holds if at each step of PIPAGE $F(x') \ge F(x)$, which in turn is the case if $\varphi(\varepsilon, x, R)$, treated as a function of ε, attains its maximum at an endpoint of the interval $[-\varepsilon_1, \varepsilon_2]$. Thus we arrive at the following condition that guarantees (6).

Definition 1. We say that the program (1)–(4) satisfies the *ε-convexity condition* if the function $\varphi(\varepsilon, x, R)$ is convex in ε for every feasible solution x to (1)–(3) and for all paths and cycles R of the graph H_x.

We describe now a general way of constructing algorithms satisfying (5). Assume that one can associate with $F(x)$ another function $L(x)$ that is defined and polynomially computable on the same set, coincides with $F(x)$ on binary x satisfying (2), and such that the program

$$\max \quad L(x) \tag{7}$$

$$\text{s.t.} \quad \sum_{e\in\delta(v)} x_e \le p_v, \quad v \in U \cup W, \tag{8}$$

$$0 \le x_e \le 1, \quad e \in E \tag{9}$$

(henceforth called the *nice relaxation*) is polynomially solvable.

Definition 2. We say that the functions F and L satisfy the *F/L lower bound condition* if

$$F(\check{x}) \geq CL(\check{x}), \tag{10}$$

where $\check{x}$ is an optimal solution to (7)–(9) and C is a positive constant.

Since $L(\check{x}) \geq F^*$, this condition implies (5). Thus if both of the above conditions hold (we will further refer to them as the *main conditions*) one can construct a C-approximation algorithm for solving (1)–(4).

Remark 1. The pipage rounding is essentially based on Properties 1 and 2. It is easy to observe that both properties hold for some modifications of the problem (1)–(4), which expose the actual range of applicability of our method. In particular, $\leq$ in (2) can be replaced by $=$. Moreover, after a few obvious alterations in the description (namely, the *ε-convexity* and *F/L lower* bound conditions should be replaced by the *ε-concavity* and *F/L upper* bound conditions, respectively) the method is applicable to the problem obtained from the above by converting max in (1) into min and $\leq$ in (2) into $\geq$ or $=$.

3. Maximum coverage

In the maximum coverage problem (MAXIMUM COVERAGE 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 MAXIMUM COVERAGE clearly implies that of the set cover problem and so it is *NP*-hard. In a sense MAXIMUM COVERAGE can be treated as an inverse of the set cover problem and like the latter has numerous applications (see, e.g. Hochbaum, 1997). It is well known that a simple greedy algorithm solves MAXIMUM COVERAGE approximately within a factor of $1-(1-1/p)^p$ of the optimum (Cornuejols et al., 1977). Feige (1998) proves that no polynomial algorithm can have better performance guarantee provided that P$\neq$NP. Another result concerning MAXIMUM COVERAGE is due to Cornuejols et al. (1980) who prove that the greedy algorithm almost always finds an optimal solution to MAXIMUM COVERAGE in the case of two-element sets. We show below that MAXIMUM COVERAGE 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 ($k = 2, 3, \ldots$) MAXIMUM COVERAGE 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 $1 - e^{-1}$ (Cornuejols et al., 1980), 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 (Bar-Yehuda and Even, 1981; Hochbaum, 1982) 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 $J = \{1, \ldots m\}$. MAXIMUM COVERAGE 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. Goemans and Williamson, 1994), MAXIMUM COVERAGE can be stated as the following integer program:

$$\max \quad \sum_{j=1}^{m} w_j z_j \tag{11}$$

$$\text{s.t.} \quad \sum_{i \in S_j} x_i \geq z_j, \quad j = 1, \ldots, m, \tag{12}$$

$$\sum_{i=1}^{n} x_i = p, \tag{13}$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, n, \tag{14}$$

$$0 \leq z_i \leq 1, \quad i = 1, \ldots, m. \tag{15}$$

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 MAXIMUM COVERAGE and the optimal solutions to (11)–(15). Note that the variables x_i determine the optimal values of z_j in any optimal solution. Moreover, it is clear that MAXIMUM COVERAGE is equivalent to maximizing the function $F(x) = \sum_{j=1}^{m} w_j (1 - \prod_{i \in S_j} (1 - x_i))$ over all binary vectors x satisfying (13). To see that that this non-linear problem is a special case of of the problem (2)–(4) we define $U = \{u\}$ and $W = \{1, \ldots, n\}$ where u is a dummy element corresponding to the cardinality constraint. Observe also that the objective function (11) can be replaced by the function

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

of the variables $x_1, \ldots, x_n$, thus providing a nice relaxation of the form (7)–(9).

We now show that the functions F and L just defined satisfy the ε-convexity and F/L lower bound conditions.

The F/L lower bound condition 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, 1994 in a similar context):

$$1 - \prod_{i=1}^{k} (1 - y_i) \geq (1 - (1 - 1/k)^k) \min\left\{1, \sum_{i=1}^{k} y_i\right\}, \tag{16}$$

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

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

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

$$g(z) \geq (1 - (1 - 1/k)^k)z,$$

as desired.

To check the ε-convexity condition it suffices to observe that in this case φ as a function in ε 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 (11)–(15) can be arbitrarily close to $(1 - (1 - 1/k)^k)$ and thus the rounding scheme described above is best possible for the integer program (11)–(15). 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 (13)–(14) 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 (13) and $0 \leq x_i \leq 1$ for all i. Now it is easy to derive an upper bound on the ratio

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

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

Remark 2. 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 (13) is replaced by the constraints

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

where $\{I_t : t = 1, \dots, r\}$ 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 3. 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:

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

where S_j and R_t are arbitrary subsets of $\{1, \dots, n\}$, 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.

4. Hypergraph Max k-Cut with given sizes of parts

In this section we consider Hypergraph Max k-Cut with given sizes of parts or, for short, HYP MAX k-CUT WITH GSP. An instance of HYP MAX k-CUT WITH GSP consists of a hypergraph $H = (V, E)$, nonnegative weights w_S on its edges S, and k positive integers $p_1, \dots, p_k$ such that $\sum_{i=1}^{k} p_i = |V|$. It is required to partition the vertex set V into k parts $X_1, X_2, \dots, X_k$, with each part X_i having size p_i, so as to maximize the total weight of the edges of H, not lying entirely in any part of the partition (i.e., to maximize the total weight of $S \in E$ satisfying $S \not\subseteq X_i$ for all i).

Several closely related versions of HYP MAX k-CUT WITH GSP were studied in the literature but few results have been obtained. Andersson and Engebretsen (1998) presented an 0.72-approximation algorithm for the ordinary HYP MAX CUT problem (i.e., for the version without any restrictions on the sizes of parts). Arora et al. (1999) designed a PTAS for dense instances of this problem or, more precisely, for the case when the hypergraph H is restricted to have $\Theta(|V|^d)$ edges, under the assumption that $|S| \leq d$ for each edge S and some constant d. Another known special case of HYP MAX k-CUT WITH GSP is MAX BISECTION. In this problem we should split a regular graph into two equal parts and maximize a total weight of the cut between them. Frieze and Jerrum (1997) obtained 0.64-approximation algorithm based on semidefinite programming relaxation.

By applying the pipage rounding method, we prove that HYP MAX k-CUT WITH GSP can be approximated within a factor of $\min\{\lambda_{|S|} : S \in E\}$ of the optimum, where $\lambda_r = 1 - (1 - 1/r)^r - (1/r)^r$. By direct calculations it easy to get some specific values of λ_r: $\lambda_2 = 1/2 = 0.5$, $\lambda_3 = 2/3 \approx 0.666$, $\lambda_4 = 87/128 \approx 0.679$, $\lambda_5 = 84/125 = 0.672$, $\lambda_6 \approx 0.665$ and so on. It is clear that λ_r tends to $1 - e^{-1} \approx 0.632$ as $r \to \infty$. A bit less trivial fact is that $\lambda_r > 1 - e^{-1}$ for each $r \geq 3$ (Lemma 2 in this paper). Summing up we arrive at the following conclusion: our algorithm finds a feasible cut of weight within a factor of 0.5 on general hypergraphs, i.e., in the case when each edge of the hypergraph has size at least 2, and within a factor of $1 - e^{-1} \approx 0.632$ in the case when each edge has size at least 3. Finally, we show that in the case of hypergraphs with each edge of size at least 3 the bound of $1 - e^{-1}$ cannot be improved, unless $P = NP$.

It is easy to see that an instance of HYP MAX k-CUT WITH GSP can be equivalently formulated as the following (nonlinear) integer program:

$$\max \quad F(x) = \sum_{S \in E} w_S \left(1 - \sum_{t=1}^{k} \prod_{i \in S} x_{it}\right) \tag{17}$$

$$\text{s.t.} \quad \sum_{t=1}^{k} x_{it} = 1 \quad \text{for all } i, \tag{18}$$

$$\sum_{i=1}^{n} x_{it} = p_t \quad \text{for all } t, \tag{19}$$

$$x_{it} \in \{0, 1\} \quad \text{for all } i \text{ and } t. \tag{20}$$

The equivalence is shown by the one-to-one correspondence between optimal solutions to the above program and optimal k-cuts $\{X_1, \ldots, X_k\}$ of instance of HYP MAX k-CUT WITH GSP defined by the relation "$x_{it} = 1$ if and only if $i \in X_t$".

By defining $U = \{1, \ldots, k\}$ and $W = \{1, \ldots, n\}$ we can see that the problem (17)–(20) is a special case of the general non-linear problem (2)–(4). As a nice relaxation we consider the following linear program:

$$\max \quad \sum_{S \in E} w_S z_S \tag{21}$$

$$\text{s.t.} \quad z_S \leq |S| - \sum_{i \in S} x_{it} \quad \text{for all } S \in E, \tag{22}$$

$$\sum_{t=1}^{k} x_{it} = 1 \quad \text{for all } i, \tag{23}$$

$$\sum_{i=1}^{n} x_{it} = p_t \quad \text{for all } t, \tag{24}$$

$$0 \leq x_{it} \leq 1 \quad \text{for all } i \text{ and } t, \tag{25}$$

$$0 \leq z_S \leq 1 \quad \text{for all } S \in E. \tag{26}$$

It is easy to see that, given a feasible matrix x, the optimal values of z_S in the above program can be uniquely determined by simple formulas. Using this observation we can exclude the variables z_S and rewrite (21)–(26) in the following equivalent way:

$$\max \quad \mathrm{L}(x) = \sum_{S \in E} w_S \min\left\{1, \min_t \left(|S| - \sum_{i \in S} x_{it}\right)\right\} \tag{27}$$

subject to (23)–(25). Note that $F(x) = L(x)$ for each x satisfying (18)–(20).

We claim that, for every feasible x and every cycle D in the graph H_x (for definitions, see Section 2), the function $\varphi(\varepsilon) = F(x(\varepsilon))$ is a quadratic polynomial with a nonnegative leading coefficient. Indeed, observe that each product $\prod_{i \in S} x_{it}(\varepsilon)$ contains at most two modified variables. Assume that a product $\prod_{i \in S} x_{it}(\varepsilon)$ contains exactly two such variables $x_{i_1t}(\varepsilon)$ and $x_{i_2t}(\varepsilon)$. Then they can have only one of the following forms: either $x_{i_1t} + \varepsilon$ and $x_{i_2t} - \varepsilon$ or $x_{i_1t} - \varepsilon$ and $x_{i_2t} + \varepsilon$, respectively. In either case ε^2 has a nonnegative coefficient in the term corresponding to the product. This proves that the ε-convexity condition does hold.

For any $r \geq 1$, set $\lambda_r = 1 - (1 - 1/r)^r - (1/r)^r$.

Lemma 1. *Let $x = (x_{it})$ be a feasible solution to* (27), (23)–(25) *and $S \in E$. Then*

$$\left(1 - \sum_{t=1}^{k} \prod_{i \in S} x_{it}\right) \geq \lambda_{|S|} \min\left\{1, \min_t \left(|S| - \sum_{i \in S} x_{it}\right)\right\}.$$

Proof: Let $z_S = \min\{1, \min_t(|S| - \sum_{i \in S} x_{it})\}$. Define q_S and t' by the equalities

$$q_S = \max_t \sum_{i \in S} x_{it} = \sum_{i \in S} x_{it'}.$$

Note that

$$z_S = \min\{1, |S| - q_S\}. \tag{28}$$

Using the arithmetic-geometric mean inequality and the fact that

$$\sum_{t=1}^{k} \sum_{i \in S} x_{it} = |S|$$

we obtain that

$$\begin{aligned}
1-\sum_{t=1}^{k}\prod_{i\in S}x_{it} &= 1-\prod_{i\in S}x_{it'}-\sum_{t\neq t'}\prod_{i\in S}x_{it}\\
&\geq 1-\left(\frac{\sum_{i\in S}x_{it'}}{|S|}\right)^{|S|}-\sum_{t\neq t'}\left(\frac{\sum_{i\in S}x_{it}}{|S|}\right)^{|S|}\\
&\geq 1-\left(\frac{q_S}{|S|}\right)^{|S|}-\left(\frac{\sum_{t\neq t'}\sum_{i\in S}x_{it}}{|S|}\right)^{|S|}\\
&= 1-\left(\frac{q_S}{|S|}\right)^{|S|}-\left(\frac{|S|-\sum_{i\in S}x_{it'}}{|S|}\right)^{|S|}\\
&= 1-\left(\frac{q_S}{|S|}\right)^{|S|}-\left(1-\frac{q_S}{|S|}\right)^{|S|}. \qquad (29)
\end{aligned}$$

Let

$$\psi(y)=1-\left(1-\frac{y}{|S|}\right)^{|S|}-\left(\frac{y}{|S|}\right)^{|S|}.$$

Case 1. $|S|-1\leq q_S\leq |S|$. Then by (28), $z_S=|S|-q_S$, and hence by (29),

$$1-\sum_{t=1}^{k}\prod_{i\in S}x_{it}\geq 1-\left(1-\frac{z_S}{|S|}\right)^{|S|}-\left(\frac{z_S}{|S|}\right)^{|S|}=\psi(z_S).$$

Since the function ψ is concave and $\psi(0)=0$, $\psi(1)=\lambda_{|S|}$, it follows that

$$1-\sum_{t=1}^{k}\prod_{i\in S}x_{it}\geq \lambda_{|S|}z_S.$$

Case 2. $1\leq q_S\leq |S|-1$. Here $z_S=1$. Since $\psi(y)$ is concave and $\psi(1)=\psi(|S|-1)=\lambda_{|S|}$,

$$1-\sum_{t=1}^{k}\prod_{i\in S}x_{it}\geq \lambda_{|S|}.$$

Case 3. $0\leq q_S\leq 1$. Again, $z_S=1$. For every t, set $\mu_t=\sum_{i\in S}x_{it}$. Note that, by the assumption of the case,

$$0\leq \mu_t\leq 1, \qquad (30)$$

and, moreover,

$$\sum_{t=1}^{k} \mu_t = |S|. \tag{31}$$

By the arithmetic-geometric mean inequality it follows that

$$\begin{aligned} \sum_{t=1}^{k} \prod_{i \in S} x_{it} &\leq \sum_{t=1}^{k} \left(\frac{\mu_t}{|S|} \right)^{|S|} \\ \text{(by (30))} \quad &\leq |S|^{-|S|} \sum_{t=1}^{k} \mu_t \\ \text{(by (31))} \quad &= |S|^{-|S|} |S|. \end{aligned}$$

Consequently,

$$\begin{aligned} 1 - \sum_{t=1}^{k} \prod_{i \in S} x_{it} &\geq 1 - |S| \left(\frac{1}{|S|} \right)^{|S|} \\ &= 1 - \left(\frac{1}{|S|} \right)^{|S|} - (|S| - 1) \left(\frac{1}{|S|} \right)^{|S|} \\ &\geq 1 - \left(\frac{1}{|S|} \right)^{|S|} - (|S| - 1)^{|S|} \left(\frac{1}{|S|} \right)^{|S|} \\ &= \lambda_{|S|}. \end{aligned}$$

□

Corollary 1. *Let $x = (x_{it})$ be a feasible solution to* (27), (23)–(25). *Then*

$$F(x) \geq \Big(\min_{S \in E} \lambda_{|S|} \Big) L(x).$$

The corollary states that the F/L lower bound condition holds with

$$C = \min_{S \in E} \lambda_{|S|}.$$

Hence the pipage rounding provides an algorithm that finds a feasible k-cut whose weight is within a factor of $\min_{S \in E} \lambda_{|S|}$ of the optimum.

Note that $\lambda_2 = 1/2$. We now establish a lower bound on λ_r for all $r \geq 3$.

Lemma 2. *For any $r \geq 3$,*

$$\lambda_r > 1 - e^{-1}.$$

Proof: We first deduce it from the following stronger inequality:

$$\left(1-\frac{1}{r}\right)^r < e^{-1}\left(1-\frac{1}{2r}\right) \quad \text{for all} \quad r \geq 1. \tag{32}$$

Indeed, for any $r \geq 3$,

$$\begin{aligned}
\lambda_r &= 1-\frac{1}{r^r}-\left(1-\frac{1}{r}\right)^r \\
&> 1-\frac{1}{r^r}-e^{-1}\left(1-\frac{1}{2r}\right) \\
&= 1-e^{-1}+\frac{1}{r}\left(\frac{e^{-1}}{2}-\frac{1}{r^{r-1}}\right) \\
&> 1-e^{-1}.
\end{aligned}$$

To prove (32), by taking natural logarithm of both sides of (32) rewrite it in the following equivalent form:

$$1+r\ln\left(1-\frac{1}{r}\right) < \ln\left(1-\frac{1}{2r}\right) \quad \text{for all} \quad r \geq 1.$$

Using the Taylor series expansion

$$\ln(1-\sigma) = -\sum_{i=1}^{\infty}\frac{\sigma^i}{i}$$

we obtain that for each $r = 1, 2, \ldots,$

$$\begin{aligned}
1+r\ln\left(1-\frac{1}{r}\right) &= 1+r\left(-\frac{1}{r}-\frac{1}{2r^2}-\frac{1}{3r^3}-\cdots\right) \\
&= -\frac{1}{2r}-\frac{1}{3r^2}-\frac{1}{4r^3}\cdots \\
&< -\frac{1}{2r}-\frac{1}{2(2r)^2}-\frac{1}{3(2r)^3}\cdots \\
&= \ln\left(1-\frac{1}{2r}\right),
\end{aligned}$$

as required. □

We now show that in the case of r-uniform hypergraphs the integrality gap for the relaxation (21)–(26) can be arbitrarily close to λ_r. It follows that no other rounding of this relaxation can provide an algorithm with a better performance guarantee.

Indeed, consider the following instance: the complete r-uniform hypergraph on $n = rq$ vertices, $k = 2$, $w_S = 1$ for all $S \in E$, $p_1 = q$ and $p_2 = n - q$. It is clear that any feasible cut in this hypergraph has weight

$$C_n^r - C_q^r - C_{n-q}^r.$$

Consider the feasible solution to (23)–(26) in which

$$x_{i1} = 1/r \quad \text{and} \quad x_{i2} = 1 - 1/r \quad \text{for each } i.$$

The weight of this solution is equal to C_n^r, since for each edge S we have

$$r - \sum_{i \in S} x_{i1} \geq r - \sum_{i \in S} x_{i2} = 1$$

and therefore $z_S = 1$ for all $S \in E$. Thus the integrality gap for this instance is at most

$$\begin{aligned}
\frac{C_n^r - C_q^r - C_{n-q}^r}{C_n^r} &= 1 - \frac{q!(n-r)!}{(q-r)!n!} - \frac{(n-q)!(n-r)!}{(n-q-r)!n!} \\
&\leq 1 - \frac{q!}{(q-r)!n^r} - \frac{(n-q)!}{(n-q-r)!n^r} \\
&\leq 1 - \frac{(q-r)^r}{n^r} - \frac{(n-q-r)^r}{n^r} \\
&= 1 - \left(\frac{1}{r} - \frac{1}{q}\right)^r - \left(1 - \frac{1}{r} - \frac{1}{q}\right)^r,
\end{aligned}$$

which tends to λ_r as $q \to \infty$.

We conclude the paper with a proof that the performance bound of $1 - e^{-1}$, our algorithm provides on hypergraphs with each edge of size at least 3, cannot be improved, unless $P = NP$.

In the Maximum Coverage problem (MAXIMUM COVERAGE 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 $X \subseteq I$ (called *coverage*) with $|X| = p$ so as to maximize the total weight of the sets in $\mathcal{F}$ having nonempty intersections with X. It is well known that a simple greedy algorithm solves MAXIMUM COVERAGE approximately within a factor of $1 - e^{-1}$ of the optimum (Cornuejols et al., 1977). Feige (1998) proved that no polynomial algorithm can have better performance guarantee, unless $P = NP$.

Our proof consists in constructing an approximation preserving reduction from MAXIMUM COVERAGE to HYP MAX k-CUT WITH GSP. Let a set I, a collection $S_1, \ldots, S_m \subseteq I$, nonnegative weights (w_j), and a positive number p form an instance A of MAXIMUM COVERAGE. Construct an instance B of HYP MAX k-CUT WITH GSP as follows: $I' = I \cup \{u_1, \ldots, u_m\}$ (assuming that $I \cap \{u_1, \ldots, u_m\} = \emptyset$), ($S_1' = S_1 \cup \{u_1\}, \ldots, S_m' = S_m \cup \{u_m\}$), the same weights w_j, and $p_1 = p$, $p_2 = |I'| - p$. Let $(X, I' \setminus X)$ be a maximum weight cut in B with the sizes of parts p_1 and p_2. It is clear that its weight is at least the weight of a maximum coverage in A. Thus it remains to transform $(X, I' \setminus X)$ into a coverage of A with the same weight. If $X \subseteq I$, we are done. Assume that X contains u_j for some j. Then successively, for each such j, replace u_j in X by an arbitrary element in S_j that is not a member of X, or if $S_j \subseteq X$, by an arbitrary element of I that is not a member of X. After this transformation and after possibly including a few more elements from I to get exactly p elements, we arrive at a coverage $Y \subseteq I$ in A whose weight is at least the weight of the cut $(X, I' \setminus X)$ in.

5. Maximum coverage with knapsack constraint

In this section we show that the pipage rounding in conjunction with the partial enumeration method by Sahni (1975) can produce good constant-factor approximations 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 $I = \{1, \dots, n\}$ with associated nonnegative weights w_j and costs c_j, and a positive integer B, to find a subset $X \subseteq I$ with $\sum_{j \in X} c_j \le B$ so as to maximize the total weight of the sets in $\mathcal{F}$ having nonempty intersections with X. MCKP includes both MAXIMUM COVERAGE and the knapsack problem as special cases. Wolsey (1982) 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 et al. (1999) have designed a $(1 - e^{-1})$-approximation algorithm for MCKP by combining the partial enumeration method of Sahni for the knapsack problem (Sahni, 1975) 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 of finding a greedy solution, solves a linear relaxation and then rounds the fractional solution by a bit more general "pipage" procedure.

Generalizing (11)–(15) rewrite MCKP as the following integer program:

$$\max \quad \sum_{j=1}^{m} w_j z_j \tag{33}$$

$$\text{s.t.} \quad \sum_{i \in S_j} x_i \ge z_j, \quad j \in J, \tag{34}$$

$$\sum_{i=1}^{n} c_i x_i \le B, \tag{35}$$

$$0 \le x_i, z_j \le 1, \quad i \in I, j \in J \tag{36}$$

$$x_i \in \{0, 1\}, \quad i \in I. \tag{37}$$

Note that one can exclude variables z_j by rewriting (33)–(37) as the following equivalent nonlinear program:

$$\max \quad F(x) = \sum_{j=1}^{m} w_j \left(1 - \prod_{i \in S_j}(1 - x_i)\right) \tag{38}$$

subject to (35)–(37).

Set $k = \max\{|S_j| : j \in J\}$. Denote by $IP[I_0, I_1]$ and $LP[I_0, I_1]$ the integer program (33)–(37) and its linear relaxation (33)–(36) 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

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

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 3) 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 (16), it follows that

$$\begin{aligned} F(x^{\mathcal{A}}) &\geq F(x^{LP}) \\ &\geq \sum_{j \in J_1} w_j + (1 - (1 - 1/k)^k) \sum_{j \in J \setminus J_1} w_j \min\left\{1, \sum_{i \in S_j} x^{LP}_i\right\} \end{aligned} \tag{39}$$

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 (33)–(37);
Output: A feasible solution $\bar{x}$ 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;

```
$\bar{x} \leftarrow x^0$;
**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 $x_i^{\mathcal{A}}$ are integral
        **then begin** $t \leftarrow 1$; $\hat{x} \leftarrow x^{\mathcal{A}}$ **end**
        **otherwise**
          **begin**
            find $i'$ such that $x_{i'}^{\mathcal{A}}$ is fractional;
            $\hat{x}_{i'} \leftarrow 0$;
            **for all** $i \neq i'$ **do** $\hat{x}_i \leftarrow x_i^{\mathcal{A}}$;
            $I_0 \leftarrow I_0 \cup \{i'\}$
          **end**
        **if** $F(\hat{x}) > F(\bar{x})$ **then** $\bar{x} \leftarrow \hat{x}$
      **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 (38), (35)–(37). If $|X^*| \leq 3$, then the output of the algorithm is an optimal solution. So we may assume that $|X^*| \geq 4$. W.l.o.g. we may also assume that the set I is ordered in such a way that $X^* = \{1, \ldots, |X^*|\}$ and for each $i \in X^*$, the element i covers the sets in $\mathcal{F}$ not covered by the elements $1, \ldots, i-1$ of the maximum total weight among $j \in X^* \setminus \{1, \ldots, i-1\}$.

Consider now that iteration of the algorithm at which $I_1 = \{1, 2, 3, 4\}$. At this iteration the algorithm runs through q steps, $q \leq n - 4$. 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 $I_0^t = \{i_1, \ldots, i_{t-1}\}$. 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 $\hat{x}^1, \ldots, \hat{x}^q$ to (38), (35)–(37) or, equivalently, subsets $\hat{X}_1, \ldots, \hat{X}_q \subseteq I$ satisfying $\hat{X}_t \supseteq I_1$ and $\hat{X}_t \cap I_0^t = \emptyset$ where $I_0^t = \{i_1, \ldots, i_{t-1}\}$, $t = 1, \ldots, q$.

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 (39) 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 $I_0^1 = \emptyset, \ldots, I_0^q$, having nonempty intersection with X^*. In other words, i_s is the first index in the series $i_1, \ldots, i_{q-1}$ lying in X^*. We claim then that

$$F(\hat{x}^s) \geq (1-(1-1/k)^k)F(x^*). \tag{40}$$

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

$$F(X \cup \{i\}) - F(X) \geq F(X \cup Y \cup \{i\}) - F(X \cup Y), \tag{41}$$

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

$$\begin{aligned} 1/4F(I_1) = 1/4F(\{1,2,3,4\}) &= 1/4(F(\{1,2,3,4\}) - F(\{1,2,3\}) \\ &\quad + F(\{1,2,3\}) - F(\{1,2\}) \\ &\quad + F(\{1,2\}) - F(\{1\}) \\ &\quad + F(\{1\}) - F(\emptyset)) \\ \text{(by the choice of } I_1) & \\ &\geq 1/4(F(\{1,2,3,i\}) - F(\{1,2,3\}) \\ &\quad + F(\{1,2,i\}) - F(\{1,2\}) \\ &\quad + F(\{1,i\}) - F(\{1\}) \\ &\quad + F(\{i\}) - F(\emptyset)) \\ \text{(by (41))} & \\ &\geq \big(F(Y \cup \{i\}) - F(Y)\big). \end{aligned}$$

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

$$1/4F(I_1) \geq F(Y \cup \{i\}) - F(Y). \tag{42}$$

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 $x_{i_s}^s$. It follows that

$$F(\hat{X}_s \cup \{i_s\}) \geq F(x_s). \tag{43}$$

Let x^{LP}, z^{LP} denote an optimal solution to $LP[I_0^s, I_1]$. Using (37), (42) and (43) we finally obtain

$$\begin{aligned}
F(\hat{x}^s) &= F(\hat{X}_s)\\
&= F(\hat{X}_s \cup \{i_s\}) - (F(\hat{X}_s \cup \{i_s\}) - F(\hat{X}_s))\\
\text{(by (42))}\quad &\geq F(\hat{X}_s \cup \{i_s\}) - 1/4F(I_1)\\
\text{(by (43))}\quad &\geq F(x^s) - 1/4F(I_1)\\
&= \sum_{j\in J_1} w_j + \sum_{j\in J\setminus J_1} w_j \left(1 - \prod_{i\in S_j}\left(1 - x_i^s\right)\right) - 1/4\sum_{j\in J_1} w_j\\
\text{(by (39))}\quad &\geq 3/4\sum_{j\in J_1} w_j + (1-(1-1/k)^k)\sum_{j\in J\setminus J_1} w_j \min\left\{1, \sum_{i\in S_j} x_i^{LP}\right\}\\
&\geq (1-(1-1/k)^k)\left(\sum_{j\in J_1} w_j + \sum_{j\in J\setminus J_1} w_j \min\left\{1, \sum_{i\in S_j} x_i^{LP}\right\}\right)
\end{aligned}$$

(by the choice of s)

$$\begin{aligned}
&\geq (1-(1-1/k)^k)\left(\sum_{j\in J_1} w_j + \sum_{j\in J\setminus J_1} w_j \min\left\{1, \sum_{i\in S_j} x_i^{*}\right\}\right)\\
&= (1-(1-1/k)^k)F(x^*).
\end{aligned}$$

6. Scheduling unrelated parallel machines with prescribed numbers of jobs to be processed on each machine

We consider the problem of scheduling unrelated parallel machines with the total weighted completion time as the objective function and bounded numbers of jobs to be processed on each machine. An instance of the problem consists of a set $\mathcal{J} = \{J_1, \ldots, J_n\}$ of n jobs and a set $\mathcal{M} = \{M_1 \ldots, M_m\}$ of m machines, positive integers p_{ij}, R_i, and rational weights w_j for all $M_i \in \mathcal{M}$, $J_j \in \mathcal{J}$. Each positive integer p_{ij} is a processing requirement associated with the job J_j and depending on the machine M_i. Each job J_j must be processed without interruption for the respective amount of time on one of the m machines. Every machine can process at most one job at a time. The machine M_i can process at most R_i jobs, i. e., in any feasible schedule at most R_i jobs must be assigned to the machine M_i (the cardinality constraints). We denote the completion time of the job J_j in a feasible schedule by C_j. The objective is to find a feasible schedule that minimizes $\sum_{J_j\in\mathcal{J}} w_j C_j$ where $w_j \geq 0$ is a weight associated with each job J_j.

Polynomial algorithms for some special cases of the problem with general R_i were developed by Granot et al. (1997). For the problem of scheduling unrelated parallel machines in the absence of cardinality constraints (or when all $R_i = n$) Skutella (1998, 1999) and Sethuraman and Squillante (1999) independently suggested a convex programming relaxation that leads to a 3/2-approximation algorithm. In the following by using essentially the same convex relaxation in conjunction with the pipage rounding we obtain an approximation algorithm with the same performance guarantee for the problem with cardinality constraints.

Notice that the problem can be reformulated as an assignment problem of jobs to machines since for a given assignment of jobs to machines the sequencing of the assigned jobs can be done optimally on each machine M_i by applying the classical Smith's ratio rule (1956): schedule the jobs in order of nonincreasing ratios w_j/p_{ij}. For each machine M_i, define the corresponding total order $\prec_i$ on the set of jobs by setting $J_j \prec_i J_k$ if either $w_j/p_{ij} > w_k/p_{ik}$ or $w_j/p_{ij} = w_k/p_{ik}$ and $j < k$. Now the problem can be further reformulated as an integer quadratic program in nm assignment variables:

$$\min \quad \sum_{J_j\in\mathcal{J}} w_j C_j \tag{44}$$

$$\text{s.t.} \quad C_j = \sum_{M_i\in\mathcal{M}} x_{ij}\left(p_{ij} + \sum_{k\prec_i j} x_{ik}p_{ik}\right), \quad J_j \in \mathcal{J}, \tag{45}$$

$$\sum_{M_i\in\mathcal{M}} x_{ij} = 1, \quad J_j \in \mathcal{J}, \tag{46}$$

$$\sum_{J_j\in\mathcal{J}} x_{ij} \le R_i, \quad M_i \in \mathcal{M}, \tag{47}$$

$$x_{ij} \in \{0, 1\}, \quad J_j \in \mathcal{J}, M_i \in \mathcal{M}, \tag{48}$$

where $x_{ij} = 1$ means that job J_j is processed on the machine M_i. Plugging constraints (45) into the objective function we obtain the following quadratic integer problem:

$$\min \quad F(x) = c^T x + \frac{1}{2}x^T Dx \tag{49}$$

$$\text{s.t.} \quad \sum_{M_i\in\mathcal{M}} x_{ij} = 1, \quad J_j \in \mathcal{J}, \tag{50}$$

$$\sum_{J_j\in\mathcal{J}} x_{ij} \le R_i, \quad M_i \in \mathcal{M}, \tag{51}$$

$$x_{ij} \in \{0, 1\}, \quad J_j \in \mathcal{J}, M_i \in \mathcal{M}, \tag{52}$$

where x is a vector of length mn consisting of all variables x_{ij} lexicographically ordered with respect to the order $1, \dots, m$ of the machines and then, for each machine M_i, the jobs ordered according to $\prec_i$. The vector $c \in R^{mn}$ is given by $c_{ij} = w_j p_{ij}$ and $D = (d_{(ij)(hk)})$ is a symmetric $mn \times mn$ matrix given by

$$d_{(ij)(hk)} = \left\{ \begin{array}{ll} 0 & \text{if } i \neq h \text{ or } j = k, \\ w_j p_{ik} & \text{if } i = h \text{ and } k \prec_i j, \\ w_k p_{ij} & \text{if } i = h \text{ and } j \prec_i k. \end{array} \right\}$$

Observe that the function $F(x)$ satisfies the ε-concavity condition. Indeed, fix any feasible fractional solution to the program (49)–(52) and any path or cycle R in the graph H_x obtained by substituting $U = \mathcal{J}$ and $W = \mathcal{M}$. The function $\varphi(\varepsilon, x, R)$ is concave since for any machine $M_i \in \mathcal{M} = W$ there are at most two edges in R incident to M_i and if there are exactly two such edges in R then one edge is decreased by ε and another is increased by the same amount. Since $d_{(ij)(hk)}$ is positive only if $i = h$ (i.e. $x_{ij}(\varepsilon, R)x_{hk}(\varepsilon, R)$ is multiplied by a positive number only if $i = h$) the function $\varphi(\varepsilon, x, R)$ is a quadratic polynomial with nonpositive main coefficient.

Skutella (1998, 1999) observed that for any binary vector $x \in \{0, 1\}^{mn}$, the linear form $c^T x$ can be rewritten as $x^T \text{diag}(c)x$ where diag(c) denotes the diagonal matrix whose diagonal entries coincide with the entries of the vector c. It follows that the following nonlinear program is a relaxation of the problem (49)–(52):

$$\min \quad Z \tag{53}$$

$$\text{s.t.} \quad Z \geq \frac{1}{2}c^T x + \frac{1}{2}x^T(D + \text{diag}(c))x, \tag{54}$$

$$Z \geq c^T x, \tag{55}$$

$$\sum_{M_i \in \mathcal{M}} x_{ij} = 1, \quad J_j \in \mathcal{J}, \tag{56}$$

$$\sum_{J_j \in \mathcal{J}} x_{ij} \leq R_i, \quad M_i \in \mathcal{M}, \tag{57}$$

$$0 \leq x_{ij} \leq 1, \quad J_j \in \mathcal{J}, M_i \in \mathcal{M}. \tag{58}$$

Skutella shows (Lemma 2.4, Skutella, 1999) that the matrix $D + \text{diag}(c)$ is positive semidefinitive. Therefore (53)–(58) is a convex program and can be solved within an additive error ε in polynomial time (Grötschel et al., 1988) (notice that the running time of this algorithm is polynomial on $\log \frac{1}{\varepsilon}$). The objective function (53) can be replaced by

$$L(x) = \max \left\{ c^T x, \frac{1}{2}c^T x + \frac{1}{2}x^T(D + \text{diag}(c))x \right\}.$$

We consider the problem of minimizing $L(x)$ subject to (56)–(58) as a nice relaxation.

To get a 3/2-approximation using the pipage rounding we only need to show that $F(x) \leq 3/2L(x)$ for all feasible vectors x. Indeed,

$$F(x) = c^T x + \frac{1}{2}x^T Dx \leq \frac{1}{2}c^T x + \frac{1}{2}c^T x + \frac{1}{2}x^T(D + \text{diag}(c))x \leq \frac{3}{2}L(x).$$

Acknowledgment

Supported by the Russian Foundation for Basic Research, project codes 01-01-00786, 02-01-01153, by INTAS, project code 00-217, and by the Programme "Universities of Russia", project code UR.04.01.012.

References

A.A. Ageev and M.I. Sviridenko, "Approximation algorithms for maximum coverage and Max Cut with given sizes of parts," *Lecture Notes in Computer Science (Proceedings of IPCO'99)*, vol. 1610, pp. 17–30, 1999.

A.A. Ageev and M.I. Sviridenko, "An approximation algorithm for Hypergraph Max k-Cut with given sizes of parts," *Lecture Notes in Computer Science (Proceedings of ESA'2000)*, vol. 1879, pp. 32–41, 2000.

G. Andersson, "An approximation algorithm for Max p-Section," *Lecture Notes in Computer Science (Proceedings of STACS'99)*, vol. 1563, pp. 237–247, 1999.

G. Andersson and L. Engebretsen, "Better approximation algorithms for Set splitting and Not-All-Equal SAT," *Inform. Process. Letters*, vol. 65, pp. 305–311, 1998.

S. Arora, D. Karger, and M. Karpinski, "Polynomial time approximation schemes for dense instances of NP-hard problems," *Journal of Computer and System Science*, vol. 58, pp. 193–210, 1999.

R. Bar-Yehuda and S. Even, "A linear-time approximation algorithm for the weighted vertex cover problem," *J. Algorithms*, vol. 2, pp. 198–203, 1981.

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*, vol. 23, pp. 789–810, 1977.

G. Cornuejols, G.L. Nemhauser, and L.A. Wolsey, "Worst-case and probabilistic analysis of algorithms for a location problem," *Operations Research*, vol. 28, pp. 847–858, 1980.

U. Feige, "A threshold of ln n for approximating set cover," *J. of ACM.*, vol. 45, pp. 634–652, 1998.

A. Frieze and M. Jerrum, "Improved approximation algorithms for MAX k-CUT and MAX BISECTION," *Algorithmica*, vol. 18, pp. 67–81, 1997.

M. Grötschel, L. Lovasz, and A. Schrijver, "Geometric algorithms and combinatorial optimization, ser," *Algorithms and Combinatorics*, vol. 2, Springer-Verlag, 1988.

F. Granot, J. Skorin-Kapov, and A. Tamir, "Using quadratic programming to solve high multiplicity scheduling problems on parallel machines," *Algorithmica*, vol. 17, pp. 100–110, 1997.

M.X. Goemans and D.P. Williamson, "New 3/4-approximation algorithms for MAX SAT," *SIAM J. Discrete Math.*, vol. 7, pp. 656–666, 1994.

D.S. Hochbaum, "Approximation algorithms for the set covering and vertex cover problems," *SIAM J. on Computing*, vol. 11, pp. 555–556, 1982.

D.S. Hochbaum, "Approximating covering and packing problems: Set Cover, Vertex Cover, Independent Set, and related problems," in *Approximation algorithms for NP-hard problems*, D.S. Hochbaum (Ed.), PWS Publishing Company: New York, 1997, pp. 94–143.

S. Khuller, A. Moss, and J. Naor, "The budgeted maximum coverage problem," *Information Processing Letters*, vol. 70, pp. 39–45, 1999.

S. Sahni, "Approximate algorithms for the 0–1 knapsack problem," *J. of ACM*, vol. 22, pp. 115–124, 1975.

J. Sethuraman and M.S. Squillante, "Optimal scheduling of multiclass parallel machines," in *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1999, pp. 963–964.

M. Skutella, "Semidefinite relaxations for parallel machine scheduling," in *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, 1998, pp. 472–481.

M. Skutella, "Convex quadratic and semidefinite programming relaxations in scheduling," *J. of ACM*, vol. 48, pp. 206–242, 2001.

W.E. Smith, "Various optimizers for single-stage production," *Naval Research and Logistics Quarterly*, vol. 3, pp. 59–66, 1956.

L.A. Wolsey, "Maximizing real-valued submodular functions: Primal and dual heuristics for location problems," *Math. Oper. Res.*, vol. 7, pp. 410–425, 1956.