ANGEWANDTE MATHEMATIK UND INFORMATIK
UNIVERSITÄT ZU KÖLN

Report No. 95.212

Tight Approximations for Resource Constrained Scheduling and Bin Packing

by

Anand Srivastav, Peter Stangier

1995

Parts of this paper appeared as extended abstract in: J. van Leeuwen (ed.), *Proceedings of the Second Annual European Symposium on Algorithms (ESA'94)*, pages 307 – 318, *Lecture Notes in Computer Science*, Vol 855, Springer Verlag, 1994.

ANAND SRIVASTAV

Institut für Informatik — Lehrstuhl Algorithmen und Komplexität
Humboldt Universität zu Berlin
Unter den Linden 6
10099 Berlin
Germany
`srivasta@informatik.hu-berlin.de`

PETER STANGIER

Institut für Informatik
Universität zu Köln
Pohligstr.1
50969 Köln
Germany
`stangier@informatik.uni-koeln.de`

1991 Mathematics Subject Classification: 60C05, 60E15, 68Q25, 90C10, 90B35
Keywords: resource constrained scheduling, multidimensional bin packing, randomized algorithms, derandomization, approximation algorithms, chromatic index.

Tight Approximations for Resource Constrained Scheduling and Bin Packing

Anand Srivastav Peter Stangier

September 1995

Abstract

We consider the following resource constrained scheduling problem. Given m identical processors, s resources with upper bounds, n independent tasks of unit length, where each task has a start time and requires one processor and a task-dependent amount of every resource. The optimization problem is to schedule all tasks at discrete times in $\mathbb{N}$, minimizing the latest completion time C_{max} subject to the processor, resource and start-time constraints. Multidimensional bin packing is a special case of this problem. The problem is NP-hard even under much simpler assumptions. In case of zero start times Röck and Schmidt (1983) showed an $(m/2)$-factor approximation algorithm and de la Vega and Lueker (1981), improving a classical result of Garey, Graham, Johnson and Yao (1976), gave for every $\epsilon > 0$ a linear time algorithm with an asymptotic approximation guarantee of $s+\epsilon$. The contribution of this paper is to break the $O(m)$ resp. $O(s)$ barrier, even in the case of zero start times, at least for problems where the number of processors and the resource bounds are in $\Omega(\log |I|)$, $|I|$ being the input size of the problem. The main results are constant factor approximation algorithms for such problems and the proof of the optimality of the achieved approximation under the hypothesis $P \neq NP$.

1 Introduction

Problem Definition and Complexity

Resource constrained scheduling with start times is the following problem: The input is

• a set $\mathcal{T} = \{T_1, \dots, T_n\}$ of independent tasks. Each task T_j needs one time unit for its completion and cannot be scheduled before its start time r_j, $r_j \in \mathbb{N}$.

• a set $\mathcal{P} = \{P_1, \dots, P_m\}$ of identical processors. Each task needs one processor.

• a set $\mathcal{R} = \{R_1, \dots, R_s\}$ of limited resources. This means that at any time all resources are available, but the available amount of each resource R_i is bounded by $b_i \in \mathbb{N}$.

• For $1 \leq i \leq s$, $1 \leq j \leq n$ let $R_i(j) \in [0,1]$ be rational resource requirements, indicating that every task T_j needs $R_i(j)$ amount of resource R_i in order to be processed.

The combinatorial optimization problem is:

Definition 1.1 *(Resource Constrained Scheduling) Find a schedule (or assignment) $\sigma : \mathcal{T} \mapsto \mathbb{N}$ of minimal time length subject to the start times, processor and resource constraints.*

□

According to the standard notation of scheduling problems the unweighted version of our problem ($R_i(j) = 0, 1$) can be formalized as

$$P|\text{res} \cdot\cdot 1, r_j, p_j = 1|C_{\max}.$$

This notation means that the number of identical processors is part of the input ($P|$) that resources are involved (res) that the number of resources and the amount of every resource are part of the input, too (res $\cdot\cdot$), that every task needs at most 1 unit of a resource (res $\cdot\cdot 1$), that start times are involved (r_j) and that the processing time of all tasks is equal ($p_j = 1$) and that the optimization problem is to schedule the tasks as soon as possible ($|C_{max}$).

The problem is NP-hard in the strong sense, even if $r_j = 0$ for all $j = 1,\dots,n$, $s = 1$ and $m = 3$ [GaJo79].

An interesting special case of resource constrained scheduling is the following generalized version of the multidimensional bin packing problem.

Definition 1.2 *(Bin Packing Problem $BIN(l,d)$) Let d,l,n be non negative integers. Given vectors $\vec{v}_1,\dots,\vec{v}_n \in [0,1]^d$, pack all vectors in a minimum number of bins such that in each bin B and for each coordinate i, $i = 1,\dots,d$,*

$$\textstyle\sum_{\vec{v}_j \in B} v_{ij} \leq l.$$

□

Observe that $BIN(1,d)$ is the multidimensional bin packing problem, and $BIN(1,1)$ is the classical bin packing problem. The intention behind the formulation with a parameter $l \geq 1$ is to analyse the relationship between bin sizes and polynomial-time approximability of the problem.

Previous Work

The known approximation algorithms for the problem class $P|\text{res} \cdots, r_j = 0, p_j = 1|C_{\max}$ are due to Garey, Graham, Johnson, Yao [GGJY76] and Röck and Schmidt [RS83]. Garey et al. constructed with the First-Fit-Decreasing heuristic a schedule of length C_{FFD} which asymptotically is a $(s+\frac{1}{3})$-factor approximation, i.e. there is an non negative integer N such that for all $C_{opt} \geq N$

$$C_{FFD} \leq C_{opt}(s + \frac{1}{3}).$$

de la Vega and Lueker [VeLu81] improved this result presenting for every $\epsilon > 0$ a linear-time algorithm with asymptotic approximation factor $d + \epsilon$. Röck and Schmidt showed, employing the polynomial-time solvability of the simpler problem $P2|\text{res} \cdot\cdot\cdot, r_j = 0, p_j = 1|C_{\max}$ with 2 processors, a $\lceil \frac{m}{2} \rceil$ factor approximation algorithm. Thus for problems with small optimal schedules or many resource constraints resp. processors these algorithms have a weak performance. Note that all these results are based on the assumption that no start-times are given, i.e. $r_j = 0$ for all tasks $T_j \in \mathcal{T}$. For example, Röck and Schmidt's algorithm cannot be used, when start-times are given, because the problem $P2|\text{res} \cdot\cdot 1, r_j, p_j = 1|C_{\max}$ is also NP-complete, so their basis solution cannot be constructed in polynomial time.

In [SrSt94] we showed a 2-factor approximation algorithm for resource constrained scheduling problems, when the resource bounds and the number of processors are in $\Omega(\log(Cs))$, where C is the length of an optimal fractional schedule, and proved that even for this class a polynomial-time ρ-approximation algorithm for $\rho < 1.5$ cannot exist, unless $P = NP$. Since this non-approximability result was derived with a reduction to the NP-complete problem of deciding if a schedule of length 2 does exist or not, it was conjectured that for other instances - exept this pathological one - better approximations might be possible.

We will show that this conjecture is true in a comprehensive sense:

The Results

Let C_{opt} be the minimum schedule length and let the integer C denote the minimum length of a schedule, if we consider the LP relaxation, where fractional assignments of the tasks to scheduling times are allowed. We briefly call solutions to the LP relaxation "fractional schedules" and solutions to the original integer problem "integral schedules"[1]. As the one main result we will present a polynomial-time approximation algorithm for the problem class $P|\text{res} \cdot \cdot 1, r_j, p_j = 1|C_{\max}$, including its rational weighted version $(0 \leq R_i(j) \leq 1)$:

(a) For every $\epsilon > 0$, $(1/\epsilon) \in I\!N$, we can find in strongly polynomial time an integral schedule of size at most $\lceil (1+\epsilon)C_{opt} \rceil$, provided that all resource bounds b_i are at least $\frac{3(1+\epsilon)}{\epsilon^2}\log(4Cs)$ and the number of processors is at least $\frac{3(1+\epsilon)}{\epsilon^2}\log(4C)$.

As a surprising consequence a schedule of length $C_{opt}+1$ can be constructed in polynomial-time, [2] whenever $b_i \geq 3C(C+1)\log(Cs))$ for all resource bounds b_i and $m \geq 3C(C+1)\log C$. This approximation guarantee is independent of the number of processors or resources and can be used also for small schedules, for example $C_{\max} = 2, 3$ or 4. The results can be extended to the case of integer resource requirements ($R_i(j) \in \mathbb{N}$) by scaling and exploiting the fact that we can handle rational weights, thus we obtain polynomial-time approximation algorithm for problems of the form $P|\text{res} \cdot \cdot \cdot, r_j, p_j = 1|C_{\max}$.

One might wonder, if under the assumptions $b_i \in \Omega(\epsilon^{-2}\log(Cs))$ for all i and $m \in \Omega(\epsilon^{-2}\log C)$ the scheduling problem is interesting enough. In other words, can it be that such problems are easily solvable in polynomial time ? The answer is that under the hypothesis $P \neq NP$ our approximation is best possible:

(b) For the simpler problem with zero start-times and $C \geq 3$ it is NP-complete [3] to decide the question "Is $C_{opt} = C$?", even if

- *a fractional schedule of size C is known,*
- *an integral schedule of size $C+1$ is known,*
- *$b_i \in \Omega(C^2\log(Cs))$ for all resource bounds b_i.*
- *$m \in \Omega(C^2 \log C)$.*

In conclusion, both results together precisely determine the border of approximability of resource constrained scheduling and sheed light on its complexity.

An interesting special case of resource constrained scheduling is the multidimensional bin packing problem and the above results imply

(c) Let $\epsilon > 0$, $(1/\epsilon) \in I\!N$. If $l \geq \frac{3(1+\epsilon)}{\epsilon^2}\lceil \log(4nd)) \rceil$, then we can find in $O(dn^3\log(nd))$ time a bin packing with L bins such that $L \leq \lceil (1+\epsilon)L_{opt} \rceil$. But even if $l = \Omega(L_R^2\log(nd))$, then it is NP-complete to decide whether $L_{opt} = L_R$ or $L_{opt} = L_R + 1$.

Furthermore, extending the method of $\log^c n$-wise independence from the case of $0-1$ to multivalued random variables, we can parallelize our algorithm in special cases.

Let $\tau \geq \frac{1}{\log n}$ and suppose that $m, b_i \geq 2n^{\frac{1}{2}+\tau}\sqrt{\log 2n(s+1)}$ for all i. Then there is a NC-algorithms that runs on $O(n^2(ns)^{1+\frac{1}{\tau}})$ parallel processors and finds in $O(\log n \log^3(ns))$ time a schedule of size at most $2C_{opt}$.

Observe that there is a gap of a $n^{\frac{1}{2}+\tau}$ factor between the lower bounds of the resource bounds and number of processors in the sequential and in the parallel algorithm. This factor comes into picture due to the estimation of higher moments required by the method of $\log^c n$-wise independence.

[1]This should not cause any confusion: C is always an integer, only the assignments of tasks to times in the discrete interval $\{1, \ldots, C\}$ are fractional numbers.

[2]Note that C is at most the sum of n and the maximal start time, hence the factor $\log(Cs)$ is within the size of the problem input.

[3]For $C = 2$ see [SrSt94]

We leave open the question, if there is a parallel 2-factor approximation algorithm for problems with $m, b_i \in \Omega(\log(Cs))$.

The paper is organized as follows. In section 2 we study a general system of integer inequalities related to resource constrained scheduling, show under which circumstances an integer solution can be constructed via derandomization and apply the results to resource constrained scheduling. In section 3 it is proved that the achieved approximation is optimal, unless $P = NP$. In section 4, as an example, the multidimensional bin packing problem is discussed and in section 5 we give for some special cases parallel counterparts of our scheduling and bin packing algorithms based on an extension of the method of $\log^c n$-wise independence to multivalued random variables (appendix section 6).

2 Integer Inequalities

Resource constrained scheduling can be modelled as a special case of a system of integral inequalities with equality constraints:

Let l, n, N be non negative integers. For $k = 1, \ldots, N$ let $A^{(k)}$ be rational $l \times n$ matrices with $0 \leq a_{ij}^{(k)} \leq 1$ and $b^{(k)} \in \mathbb{Q}_+^l$. For a vector $y_j \in \{0,1\}^N$ let y_{jk} be its k-th component, $k = 1, \ldots, N$. Let $y^{(k)}$ be the vector of all the k-th components, i.e. $y^{(k)} = (y_{1k}, \ldots, y_{nk})$. Consider the following system of linear inequalities

Inequality System (IS)

$$\begin{array}{rcl} A^{(k)}y^{(k)} & \leq & b^{(k)} \ \forall k = 1, \ldots, N \\ ||y_j||_1 & = & 1 \ \forall j = 1, \ldots, n \\ y_j & \in & \{0,1\}^N \\ y^{(k)} & = & (y_{1k}, \ldots, y_{nk}). \end{array}$$

Resource constrained scheduling fits in this scheme as follows: consider the processors as a resource R_{s+1} with requirement $R_{s+1}(j) = 1$ for all j and bound m. Let $R = (R_i(j))_{ij}$ be the resource constraint matrix, set $A^{(k)} = R$ and $b^{(k)} = (b, m)$ for all $k = 1, \ldots, N$. Then, the problem of finding a minimum N such that a so defined inequality system (IS) has a solution is equivalent to the problem of finding a schedule of minimal length.

In general, one cannot expect to solve (IS) in polynomial time, even if a fractional solution exists, because then we would be able to decide the solvability of resource constrained scheduling.

The key observation is that (IS) can be solved, if the same system with a tighter right hand side, say $(1+\epsilon)^{-1}b^{(k)}$ instead of $b^{(k)}$, is fractionally solvable.

Define the ϵ-version of (IS) for $0 < \epsilon \leq 1$ as

ϵ-Inequality System ($IS(\epsilon)$)

$$\begin{array}{rcl} A^{(k)}y^{(k)} & \leq & (1+\epsilon)^{-1}b^{(k)} \ \forall k = 1, \ldots, N \\ ||y_j||_1 & = & 1 \ \forall j = 1, \ldots, n \\ y_j & \in & \{0,1\}^N \\ y^{(k)} & = & (y_{1k}, \ldots, y_{nk}). \end{array}$$

We need the following parameters. Let l_1 be an integer with $l_1 \leq l$ and define

$$b_{\epsilon,1} = \frac{3(1+\epsilon)}{\epsilon^2}\log(4l_1N) \ and \ b_{\epsilon,2} = \frac{3(1+\epsilon)}{\epsilon^2}\log(4(l-l_1)N). \tag{1}$$

Theorem 2.1 *(Rounding Theorem) Let $0 < \epsilon \leq 1$, l_1 and $b_{\epsilon,i}$ as in (1). Suppose that $b_i^{(k)} \geq b_{\epsilon,1}$ for all $1 \leq i \leq l_1$ and $b_i^{(k)} \geq b_{\epsilon,2}$ for all $i > l_1$. Suppose that $u = (u_1, \ldots, u_n)$ with $u_j \in [0,1]^N$ is a fractional solution for the ϵ-scaled system IS(ϵ). Then a vector $x = (x_1, \ldots, x_n)$ with $x_j \in \{0,1\}^N$ satisfying the inequalities of system IS, i.e.*

$$A^{(k)}x^{(k)} \leq b^{(k)}, \ \forall k \quad and \quad ||x_j||_1 = 1 \ \forall j$$

can be constructed in $O(Nln^2 \log(Nln))$ *time.*

$\square$

For the proof we need the following derandomization result which is an algorithmic version of the Angluin-Valiant inequality for multi-valued random variables. Let l, n, N be non-negative integers. We are given n mutually independent random variables X_j with values in $\{1, \ldots, N\}$ and probability distribution $Prob(X_j = k) = \tilde{x}_{jk}$ for all $j = 1, \ldots, n$, $k = 1, \ldots, N$ and $\sum_{k=1}^{N} \tilde{x}_{jk} = 1$. Suppose that the $\tilde{x}_{jk}$ are rational numbers with $0 \leq \tilde{x}_{jk} \leq 1$. Let X_{jk} denote the random variable which is 1, if $X_j = k$ and is 0 else. For $1 \leq k \leq N$, $1 \leq i \leq l$, $1 \leq j \leq n$ let $w_{ij}^{(k)}$ be rational weights with $0 \leq w_{ij}^{(k)} \leq 1$. For $i = 1, \ldots, l$ and $k = 1, \ldots, N$ define the sums ψ_{ik} by

$$\psi_{ik} = \sum_{j=1}^{n} w_{ij}^{(k)} X_{jk}. \tag{2}$$

Let $0 < \beta_{ik} \leq 1$ be rational numbers.

Denote by E_{ik} the event

$$\psi_{ik} \leq \mathbb{E}(\psi_{ik})(1 + \beta_{ik}). \tag{3}$$

Let (E_{ik}) be a collection of lN such events. Define

$$f(\beta_{ik}) = \exp(-\tfrac{\beta_{ik}^2 \mathbb{E}(\psi_{ik})}{3}).$$

By the Angluin-Valiant inequality ([McD89], Theorem 5.7) we have for all k

Proposition 2.2 $\mathbb{P}[E_{ik}^c] \leq f(\beta_{ik})$.

$\square$

Thus the probability that there is some (i, k) so that E_{ik}^c holds is at most $\sum_{i=1}^{m} \sum_{k=1}^{N} f(\beta_{ik})$. Let us assume that this is bounded away from one, i.e.

$$\sum_{i=1}^{m} \sum_{k=1}^{N} f(\beta_{ik}) \leq 1 - \delta. \tag{4}$$

for some $0 < \delta < 1$. The following theorem is a special case of the algorithmic version of the Angluin-Valiant inequality for multivalued random variables (Theorem 2.13 in [SrSt94]).

Theorem 2.3 *([SrSt94] Let* $0 < \delta < 1$ *and* E_{ik} *be as above satisfying (4). Then*

$$\mathbb{P}(\bigcap_{i=1}^{m} \bigcap_{k=1}^{N} E_{ik}) \geq \delta$$

and a vector $x \in \bigcap_{i=1}^{m} \bigcap_{k=1}^{N} E_{ik}$ *can be constructed in* $O\left(Nln^2 \log \frac{Nln}{\delta}]\right)$*-time.*

$\square$

We are ready to prove Theorem 2.1.

Proof of Theorem 2.1: Set $[N] = \{1, \ldots, N\}$. Let $X_1, \ldots, X_n$ be mutually independent random variables with values in $[N]$ defined by $\mathbb{P}[X_j = k] = u_{jk}$, $k \in [N]$. For j, k let X_{jk} be the $0 - 1$ random variable which is 1 if $X_j = k$ and 0 else. Furthermore, for each $k \in [N]$ let $X^{(k)}$ be the vector $(X_{1k}, \ldots, X_{nk})$. For $i = 1, \ldots, l$ define sums ψ_{ik} by

$$\psi_{ik} = (A^{(k)} X^{(k)})_i = \sum_{j=1}^{n} a_{ij}^{(k)} X_{jk}.$$

Let E_{ik} be the event

$$\psi_{ik} \leq (1+\epsilon)(1+\epsilon)^{-1} b_i^{(k)}.$$

By the Angluin-Valiant inequality and the assumption $b_i^{(k)} \geq b_{\epsilon,1}$ for $1 \leq i \leq l_1$ and $b_i^{(k)} \geq b_{\epsilon,2}$ for $l_1 + 1 \leq i \leq l$

$$\mathbb{P}[E_{ik}^c] \leq \exp\left(-\frac{\epsilon^2 b_i^{(k)}}{3(1+\epsilon)}\right) \leq \frac{1}{2N\mu},$$

where $\mu = l_1$ or $\mu = l - l_1$ resp.. Hence

$$\mathbb{P}[\bigcup_{ik} E_{ik}^c] \leq \frac{1}{2}.$$

With $\delta = \frac{1}{2}$ we can invoke the algorithmic Angluin-Valiant inequality for multi-valued random variables (Theorem 2.3) and can construct vectors $x_1, \ldots, x_n$ with $x_j \in \{0,1\}^N$ and $||x_j||_1 = 1$ such that

$$A^{(k)} x^{(k)} \leq b^{(k)} \quad \forall k$$

in $O(Nln^2 \log(Nln))$ time.

□

In order to show the claimed approximation guarantee for resource constrained scheduling we proceed as follows. In the first step we solve the linear programming relaxation of the integer program associated to resource constrained scheduling. Now an integer schedule can be generated in principle with our rounding theorem (Theorem 2.1). Since the rounding theorem can be applied only if a fractional solution within the smaller resource bound vector $(1+\epsilon)^{-1}b$ is available, in the second step we must show the existence of such a solution.

Fractional Solutions

Let $r_{max} := \max_{j=1,\ldots,n} r_j$ and $D = r_{max} + n$. Thus

$$C \leq C_{opt} \leq D.$$

The following integer linear program is equivalent to resource constrained scheduling with start times.

$$\begin{array}{rcll} & \min S & & \\ \sum_j R_i(j) x_{jz} & \leq & b_i & \forall\, R_i \in \mathcal{R}, \\ & & & z \in \{1, \ldots, D\} \\ \sum_z x_{jz} & = & 1 & \forall\, T_j \in \mathcal{T} \\ x_{jz} & = & 0 & \forall\, T_j \in \mathcal{T}, z < r_j \text{ and} \\ & & & \forall\, T_j \in \mathcal{T}, z > S \\ x_{jz} & \in & \{0,1\}. & \end{array}$$

The fractional optimal schedule C can be found in a standard way (see for example [LST90]): Start with an overall integer deadline $\widetilde{C} \leq D$ and check whether the LP

$$\begin{array}{rcll} \sum_j R_i(j) x_{jz} & \leq & b_i & \forall\, R_i \in \mathcal{R}, \\ & & & z \in \{1, \ldots, D\} \\ \sum_z x_{jz} & = & 1 & \forall\, T_j \in \mathcal{T} \\ x_{jz} & = & 0 & \forall\, T_j \in \mathcal{T}, z < r_j \text{ and} \\ & & & \forall\, T_j \in \mathcal{T}, z > \widetilde{C} \\ x_{jz} & \in & [0,1] & \end{array}$$

has a solution. Using binary search we can find C along with fractional assignments $(\widetilde{x}_{jz})$ solving at most $\log D$ such LPs. W.l.o.g we can assume that $D = O(n)$. Hence this procedure is a polynomial-time algorithm, if we use standard polynomial-time LP algorithms.

Our goal is to find an integer solution using the rounding theorem (Theorem 2.1). But at this moment we cannot apply it, because the rounding theorem requires a fractional solution within the tighter resource bound $\frac{b}{1+\epsilon}$, while C and the assignments $(\tilde{x}_{ij})$ are feasible only within the bound b. It should be intuitively clear that given a fractional solution within the resource bound b and a fractional schedule of length C, a new fractional solution within $\frac{b}{1+\epsilon}$ might be constructable enlarging the length of the schedule to some $C_\epsilon > C$.

But how can we choose an appropriate C_ϵ ? Let $\epsilon > 0$ with $(1/\epsilon) \in \mathbb{N}$. Consider the time interval $\{1, \ldots, \lceil (1+\epsilon)C \rceil\}$. Define a new fractional solution as follows. Call $\{C+1, \ldots, \lceil (1+\epsilon)C \rceil\}$ the ϵ-compressed image of $\{1, \ldots, C\}$ and put $\delta = \frac{1}{1+\epsilon}$.

Set

$$I = \{1, \ldots, \lceil (1+\epsilon)C \rceil\},$$

$$I_0 = \{1, \ldots, C\},$$

$$I_1^\epsilon = \{C+1, \ldots, C + \lfloor \epsilon C \rfloor\}$$

and

$$I_2^\epsilon = \begin{cases} \emptyset, & \text{if } \epsilon C \in \mathbb{N} \\ \{\lceil (1+\epsilon)C \rceil\} & \text{else} \end{cases}$$

Then $I = I_0 \cup I_1^\epsilon \cup I_2^\epsilon$. For integers l let g, f be functions defined by

$$f(l) = \frac{l-C}{\epsilon} \text{ and } g(l) = \frac{l-C}{\epsilon} + 1 - \frac{1}{\epsilon}.$$

The new fractional assignments $\widehat{x}_{jl}$ are

$$\widehat{x}_{jl} := \begin{cases} \delta \widetilde{x}_{j,l} & \text{for} \quad l \in I_0 \\ \sum_{t=g(l)}^{f(l)} (1-\delta)\widetilde{x}_{jt} & \text{for} \quad l \in I_1^\epsilon \\ \sum_{t=g(l)}^{C} (1-\delta)\widetilde{x}_{jt} & \text{for} \quad l \in I_2^\epsilon. \end{cases} \tag{5}$$

It is instructive to state the randomized rounding algorithm behind the rounding theorem (Theorem 2.1) for resource constrained scheduling explicitly. Here is the algorithm:

Algorithm *RANDOM-SCHEDULE*

Schedule the tasks at times selected by the following randomized procedure:

(a) Cast n mutually independent dice each having $N = \lceil (1+\epsilon)C \rceil$ faces where the z-th face of the j-th die corresponding to task j appears with probability $\widetilde{x}_{jz}$. (The faces stand for the scheduling times)

(b) Schedule task T_j at the time selected in (a).

□

The following examples illustrate this algorithm.

For $\epsilon = 1$ we double the interval I_0 and get $\{1, \ldots, C, C+1, \ldots, 2C\}$. Then each task T_j is randomly scheduled at time $z \in I_0$ with probability $\frac{1}{2}\widetilde{x}_{jz}$ and is scheduled at time $z + C$ with the same probability. While this is quite clear, the idea of ϵ-compressed image becomes more transparent looking at $\epsilon = \frac{1}{2}$.

For $\epsilon = \frac{1}{2}$ we have the enlarged interval $\{1, \ldots, C, C+1, \ldots, \lceil \frac{3}{2}C \rceil\}$. Given a task T_j we randomly assign it to a time $z \in \{1, \ldots, C\}$ with probability $\frac{2}{3}\tilde{x}_{jz}$, assign it to $z = C+1$ with probability $\frac{1}{3}\tilde{x}_{j1} + \frac{1}{3}\tilde{x}_{j2}$, to $z = C+2$ with probability $\frac{1}{3}\tilde{x}_{j3} + \frac{1}{3}\tilde{x}_{j4}$ and so on up to the assignment to $z = \lfloor \frac{3}{2}C \rfloor$. Now there are two cases:

If $3C$ is even, we assign T_j to $\lfloor \frac{3}{2}C \rfloor$ with probability $\frac{1}{3}\tilde{x}_{j,C-1} + \frac{1}{3}\tilde{x}_{j,C}$ and since $\lceil \frac{3}{2}C \rceil = \lfloor \frac{3}{2}C \rfloor$, we are done. If $3C$ is odd, then we assign T_j to $\lfloor \frac{3}{2}C \rfloor$ with probability $\frac{1}{3}\tilde{x}_{j,C-2} + \frac{1}{3}\tilde{x}_{j,C-1}$ and to $\lfloor \frac{3}{2}C \rfloor + 1$ with probability $\frac{1}{3}\tilde{x}_{jC}$.

The following figures illustrate this schedule enlargment. Given a fractional schedule as indicated in figure 1 we generate the schedule as shown in figure 2.

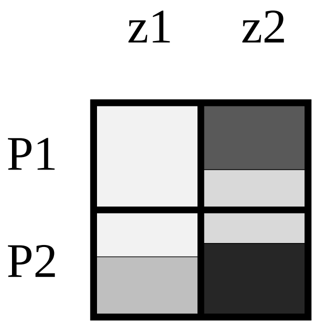

Figure 1: A fractional schedule

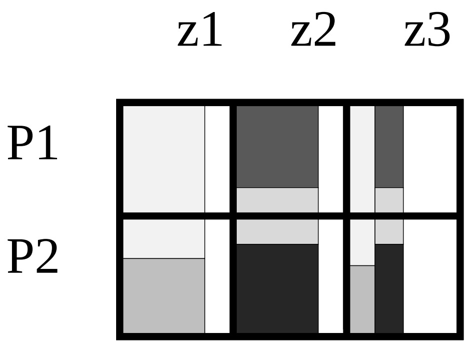

Figure 2: The new fractional schedule

The following lemma shows that the new fractional schedule is indeed feasible.

Lemma 2.4 *The vectors $\hat{x}_j = (\hat{x}_{j1}, \ldots, \hat{x}_{jN})$, $j = 1, \ldots, n$ form a fractional solution for resource constrained scheduling with $(1+\epsilon)^{-1}m$ processors and resource bound vector $(1+\epsilon)^{-1}b$.*

□

Proof: Let $\delta = \frac{1}{1+\epsilon}$ and consider an arbitrary resource R_i. (We can regard the processor constraint as an additional resource, thus the following arguments also apply to the processor constraint)

The resource constraints are satisfied, because for $l \in I_0$ we have

$$\sum_{j=1}^{n} R_i(j)\hat{x}_{jl} \leq \delta b_i,$$

for $l \in I_1^\epsilon$ (using $f(l) - (g(l) - 1) = \frac{1}{\epsilon}$)

$$\begin{aligned}\sum_{j=1}^{n} R_i(j)\widehat{x}_{jl} &= \sum_{j=1}^{n}\sum_{t=g(l)}^{f(l)} (1-\delta)R_i(j)\widetilde{x}_{jt} \\ &\leq \frac{1}{\epsilon}(1-\delta)b_i = \delta b_i,\end{aligned}$$

and for $l \in I_2^\epsilon$ (using $C - (g(l) - 1) \leq \frac{1}{\epsilon}$)

$$\begin{aligned}\sum_{j=1}^{n} R_i(j)\widehat{x}_{jl} &= \sum_{j=1}^{n}\sum_{t=g(l)}^{C} (1-\delta)R_i(j)\widetilde{x}_{jt} \\ &\leq \frac{1}{\epsilon}(1-\delta)b_i = \delta b_i.\end{aligned}$$

Furthermore for all tasks T_j,

$$\begin{aligned}\sum_{l\in I} \widehat{x}_{jl} &= \sum_{l\in I_0} \delta\widetilde{x}_{jl} + \sum_{l\in I_1^\epsilon}\sum_{t=g(l)}^{f(l)} (1-\delta)\widetilde{x}_{jt} + \sum_{l\in I_2^\epsilon}\sum_{t=g(l)}^{C} (1-\delta)\widetilde{x}_{jt} \\ &= \sum_{l=1}^{C} \delta\widetilde{x}_{jl} + \sum_{l=1}^{C}(1-\delta)\widetilde{x}_{jl} \\ &= 1.\end{aligned}$$

□

Using the rounding theorem (Theorem 2.1) the derandomized version of the algorithm *RANDOM SCHEDULE* simply is:

Algorithm *SCHEDULE*

Determine by the rounding theorem for each task its scheduling time.

□

Theorem 2.5 *For every $\epsilon > 0$ with $(1/\epsilon) \in I\!N$, the algorithm* SCHEDULE *finds a feasible integral schedule of size at most $\lceil (1+\epsilon)C \rceil$ in polynomial time, provided that $m \geq \frac{3(1+\epsilon)}{\epsilon^2}\lceil \log(4C)) \rceil$ and $b_i \geq \frac{3(1+\epsilon)}{\epsilon^2}\lceil \log(4Cs)) \rceil$ for all $i = 1, \ldots, s$.*

□

Proof: As above let $N = \lceil (1+\epsilon)C \rceil$. Let $A = (R_i(j))_{ij}$ be the extended $(s+1) \times n$ resource constraint matrix where the resource R_{s+1} represents the processor requirements and thus is defined by $R_{s+1}(j) = 1$ for all j. For $k = 1, \ldots, N$ let $b^{(k)} := (b, m)$ be the resource bound vector. The theorem is equivalent to the problem of finding a solution to the following inequality system

$$\begin{aligned} Ay^{(k)} &\leq b^{(k)}\ \forall k = 1, \ldots, N \\ ||y_j||_1 &= 1\ \forall j = 1, \ldots, n \\ y_j &\in \{0,1\}^N \\ y^{(k)} &= (y_{1k}, \ldots, y_{nk}). \end{aligned} \tag{6}$$

(The vector y_j and the condition $||y_j||_1 = 1$ stand for the assignment of task T_j to exactly one time in $\{1, \ldots, N\}$.)

By Lemma 2.4 the assignment vectors $\hat{x}_1, \ldots, \hat{x}_n$ with $\hat{x}_j \in [0,1]^N$ and $\sum_{k=1}^N \hat{x}_{jk} = 1$ are a fractional solution to the ϵ-scaled system

$$\begin{array}{rcl} Ay^{(k)} & \leq & (1+\epsilon)^{-1} b^{(k)} \ \forall k = 1, \ldots, N \\ \|y_j\|_1 & = & 1 \ \forall j = 1, \ldots, n \\ y_j & \in & \{0,1\}^N \\ y^{(k)} & = & (y_{1k}, \ldots, y_{nk}). \end{array}$$

We invoke the rounding theorem (Theorem 2.1) with $l_1 = s$: the assumptions of the theorem in the new notation read as

$$b_i^{(k)} \geq \frac{3(1+\epsilon)}{\epsilon^2} \lceil \log(4Cl_1)) \rceil \ \ \forall i = 1, \ldots, l_1$$

and

$$b_{l_1+1}^{(k)} \geq \frac{3(1+\epsilon)}{\epsilon^2} \lceil \log(4C)) \rceil,$$

thus the conditions required by the rounding theorem are satisfied. With the rounding theorem we can find in deterministic polynomial time a solution to the inequality system (6).

□

With $\epsilon = 1$ we get our result of [SrSt94]

Corollary 2.6 *If $m \geq 6\lceil \log(4C)) \rceil$ and $b_i \geq 6\lceil \log(4C(s+1)) \rceil$ for all i, then a schedule of size at most $2C$ can be found in deterministic polynomial time.*

□

And with $\epsilon = \frac{1}{C}$ we infer the - as it will be shown in the next section - optimal approximation.

Corollary 2.7 *If $m \geq 3C(C+1)\lceil \log(4C)) \rceil$ and $b_i \geq 3C(C+1)\lceil \log(4C(s+1)) \rceil$, for all i, then a schedule of size at most $C+1$ can be found in deterministic polynomial time.*

□

Note that in case of 0–1 weights (i.e. $R_i(j) \in \{0,1\}$) we have a *strongly* polynomial approximtion algorithm, because then the strongly polynomial LP algorithm of Tardos [Ta86] can be used to find the optimal fractional schedule. In any case the running time of the LP algorithm dominates clearly the running time of derandomization.

Finally, suppose that the resource requirements $R_i(j)$ are integers with bounds b_i, whereas in the problems above we assumed $0 \leq R_i(j) \leq 1$. Scaling the resource bounds reduces the problem to the 0–1 case with rational resource requirements and we may apply the results above. Scaling goes as follows: compute for every resource R_i the number $R_{\max}(i) = \max_{T_j \in \mathcal{T}} R_i(j)$, set $R_i(j)' = \frac{R_i(j)}{R_{\max}}$ and $b_i' = \frac{b_i}{R_{\max}}$. Identifying the processors with resource R_{s+1} we have

Corollary 2.8 *Consider the resource constrained scheduling problem with integer resource requirments, i.e. $R_i(j) \in I\!N$ for all $i = 1, \ldots, n$ and $i = 1, \ldots, s+1$. For every $\epsilon > 0$, $(1/\epsilon) \in I\!N$ an integral schedule of size at most $\lceil (1+\epsilon) C_{opt} \rceil$ can be found in polynomial time, provided that $b_i' \geq \frac{3(1+\epsilon)}{\epsilon^2} \lceil \log(4C(s+1)) \rceil$ for $i = 1, \ldots, s+1$.*

□

Proof: The proof is based on the observation that if a schedule for the scaled problem using b_i' and $R_i(j)'$ is feasible, then the rescaled schedule is feasible for the original problem. Now construct an integer schedule for the scaled problem with Theorem 2.5.

□

Furthermore, the choice of $\epsilon = 1$ yields a 2-factor approximation for problems of the general form $P|\text{res} \cdots, p_j = 1, r_j | C_{\max}$, and setting $\epsilon = \frac{1}{C}$ the optimum of an arbitrary resource constrained scheduling problem can be approximated up to *one* time-unit, provided the b_i's are as large as required above. Note that for any *constant* R_{max} this gives a tight approximation, since then b_i still grows logarithmically in the input size.

Remark 2.9 For scheduling of unrelated parallel machines results of similar flavour have been achieved by Lenstra, Shmoys and Tardos [LST90] and Lin and Vitter [LiVi92]. Lenstra, Shmoys and Tardos [LST90] gave a 2-factor approximation algorithm for the problem of scheduling independent jobs with different processing times on *unrelated* processors and also proved that there is no ρ-approximation algorithm for $\rho < 1.5$, unless $P = NP$. Lin and Vitter [LiVi92] considered the generalized assignment problem and the problem of scheduling of unrelated parallel machines. For the generalized assignment problem with resource constraint vector b they could show for every $\epsilon > 0$ an $1+\epsilon$ approximation of the minimum assignment cost, which if feasible within the enlarged packing constraint $(2 + \frac{1}{\epsilon})b$.

3 Non-Approximability

In this section we do not distinguish between the processor and the other resource constraints, but consider the processor requirement as an additional resource constraint.

Under the assumption $b_i = \Omega(C^2 \log(Cs))$ we have constructed an integral schedule of size at most $C + 1$. This is very close to the truth, because now C_{opt} is either C or $C + 1$. We will show for all fix $C \geq 3$ that even under the assumption $b_i = \Omega(C^2 \log(Cs))$ it is NP-complete to decide whether $C_{opt} = C + 1$ or $C_{opt} = C$. (For $C = 2$ we refer to [SrSt94])

In the remainder of this section we consider the "simpler" problem with zero start times. Then $C \leq n$ and the NP-completeness of scheduling problems with $b_i = \Omega(\log(ns))$ implies the NP-completeness of problems with $b_i = \Omega(\log(Cs))$. The following results show the NP-completeness of resource constrained scheduling under different conditions. In Theorem 3.1 we consider $b_i = 1$ and 0–1 resource requirements $R_i(j) \in \{0, 1\}$. Theorem 3.2 covers the case of $b_i = \Omega(\log(ns))$ (and still $R_i(j) \in \{0, 1\}$). In Theorem 3.3 we include the case of some $R_i(j)$ being fractional, i.e $R_i(j) \in \{0, 1, \frac{1}{2}\}$, and Theorem 3.4 covers the case $b_i = \Omega(C^2 \log(Cs))$. In particular, this shows that resource constrained scheduling is NP-complete, even if we ask for *small* schedules, i.e. "Is $C_{opt} = 3$?".

Theorem 3.1 *Under the assumptions that there exist a fractional schedule of size $C \geq 3$ and an integral schedule of size $C + 1$, and $b_i = 1$ for all resource bounds, it is NP-complete to decide whether or not there exists an integral schedule of size C.*

□

Proof: We give a reduction to the chromatic index problem which is NP-complete [Hol81]. The following is known about the chromatic index $\chi'(G)$ of a graph G. Let $\Delta(G)$ be the maximal vertex degree in G. Then by Vizing's theorem [Viz64] $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$ and an edge coloring with $\Delta(G) + 1$ colors can be constructed in polynomial time. But it is NP-complete to decide whether there exists a colouring that uses $\Delta(G)$ colours, even for cubic graphs, i.e. $\Delta(G) = 3$ [Hol81]. Therefore the edge colouring problem is NP-complete for any *fixed* $\Delta \geq 3$. This is an important fact which will be used in the proofs of Theorem 3.3 and Theorem 3.4.

Now to the reduction. Let $G = (V, E)$ be a graph with $|V| = \nu$, $|E| = \mu$ and $\deg(v) \leq \Delta$ for all $v \in V$. We construct a resource constrained scheduling problem asssociated to G as follows.

Introduce for every edge $e \in E$ exactly one task T_e and consider $\mu = |E|$ identical processors. We will freely call the edges tasks and vice versa. For every node $v \in V$ define a resource R_v with bound 1 and resource/task requirements

$$R_v(e) = \begin{cases} 1 & \text{if} \quad v \in e \\ 0 & \text{if} \quad v \notin e. \end{cases}$$

Claim 1: There exists a colouring that uses Δ colours if and only if there is a feasible integral schedule of size Δ.

Suppose there is an integral schedule of size Δ: colour each edge with the "colour" equal to its scheduling time. Now all edges incident to the same node have different colours. If this is not true, then there would be a node $v \in V$ covered by two edges $e, e' \in E$ of the same colour, which means that the tasks corresponding to the edges are scheduled at the same time z. So

$$x_{e'z} = x_{ez} = 1$$

and

$$R_v(e')x_{e'z} + R_v(e)x_{ez} = 1 + 1 = 2,$$

contradicting our assumption of a feasible schedule.

Suppose there is an edge colouring $\chi' : E \mapsto \{1, \ldots, \Delta\}$. Then schedule task T_e at the time $\chi'(e)$. It is easily verified that this is an feasible integral schedule.

Furthermore, there is a fractional schedule of size $C = \Delta$: Simply set

$$x_{ez} = \frac{1}{\Delta} \qquad \forall\ 1 \le z \le \Delta.$$

This schedule has size Δ and we have

$$\sum_{z=1}^{\Delta} x_{ez} = 1 \quad \forall\ T_e$$

and

$$\sum_{e} R_v(e)x_{ez} \le 1 = b_v \quad \forall\ z.$$

□

In Corollary 2.6 we assumed that $b_i \ge 6\lceil \log(4C(s+1)\rceil$ which we did not respect in the reduction above. In the next two theorems we show how this assumption can be included.

Theorem 3.2 *Under the assumption that there exist a fractional schedule of size $C \ge 3$ and an integral schedule of size $C+1$, $b_i = \Omega(\log(ns))$ for all resource bounds, and C is fixed, it is NP-complete to decide whether or not there exists an integral schedule of size C.*

Proof: We follow the proof of Theorem 3.1. But instead of respresenting each edge of the graph $G = (V, E)$ by one task, we consider $2K\Delta$ tasks where $K = \lceil \log \mu \rceil$ and $\mu = |E|$. For each $e \in E$ let us introduce $2K$ red tasks

$$T_1^r(e), \ldots, T_{2K}^r(e)$$

and $2K(\Delta - 1)$ blue tasks.

$$T_1^b(e), \ldots, T_{2K(\Delta-1)}^b(e).$$

We get rid of the processor constraint considering $\mu 2K\Delta$ identical processors, hence there are enough processors to schedule every task at any time. For every node $v \in V$ we define a resource R_v with bound $2K$ and introduce for every edge $e \in E$ a resource R_e also having bound $2K$. The requirements are: every red task $T_i^r(e)$ needs one unit of resource R_v, if e is incident with v. All the other tasks including the blue tasks $T_i^b(e)$ do not need any unit of the resource R_v. Every red or blue task needs one unit of its corresponding edge resource R_e. Hence a feasible schedule of size Δ has to schedule all tasks corresponding to an edge in packings of size at most $2K$. The crucial observation is:

If we can ensure that all red tasks corresponding to the same edge are scheduled at the same time, then we can define a scheduling time for this edge and can conclude arguing as in the proof of Theorem 3.1, where we showed that it is as hard to find a schedule of size Δ as to determine the existence of an edge colouring with Δ colors.

Figure 3 shows such a feasible schedule. To ensure that all red tasks corresponding to the same edge are scheduled at the same time we introduce a new resource type:

For every edge e, every red task $T_i^r(e)$ and every K-element subset S of blue tasks corresponding to e, define a resource $R_{T_i^r(e),S}$ with bound K whose requirements are defined as follows. Each task in $S \cup \{T_i^r(e)\}$ needs one unit of $R_{T_i^r(e),S}$ and all the other tasks do not need any unit of the resource $R_{T_i^r(e),S}$. Observe that the number of such resources is

$$2K\mu\binom{2K(\Delta-1)}{K}.$$

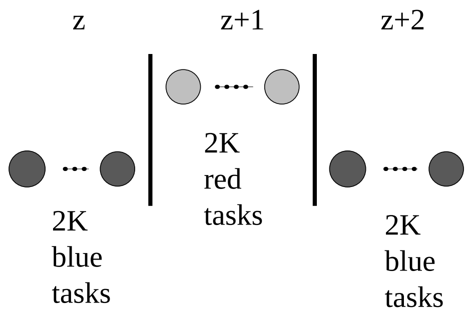

Figure 3: A part of a feasible schedule with tasks corresponding to one edge.

We are ready to show:

Claim 1: The edges of G can be colored with Δ colors, if and only if the above defined scheduling problem has a schedule of length Δ.

Here is a proof.

Suppose that the edges of G can be colored with Δ colors. Let Z be the time set $Z = \{1, \ldots, \Delta\}$. For each $e \in E$ schedule all red tasks $T_i^r(e)$ at the time corresponding to the color of e, say $z_e \in Z$ and schedule the blue tasks $T_i^b(e)$ in packets of $2K$ at the remaining times $z \neq z_e$, $z \in Z$. It is easily verified that this is a feasible schedule.

Suppose we are given a schedule of length Δ. Due to the bound K for the resources $R_{T_i^r(e),S}$ it is not hard to show that in such a schedule all red tasks $T_i^r(e)$ corresponding to the same edge e must be scheduled at the same time. This scheduling time defines a unique color for every edge and the proof of Theorem 3.1 shows that this is a desired coloring.

We are done, if we can show $b_i = \Omega(\log(ns))$. Let $\nu = |V|$ and let us assume that $\nu \leq \mu$. In total we introduced

$$\begin{aligned} s &= \nu + \mu + 2\mu\log\mu\binom{2(\Delta-1)\log\mu}{\log\mu} \\ &< 2\mu + 2\mu\log\mu 2^{2\Delta\log\mu} \\ &< \mu^3 2^{2\Delta} \\ &< \mu^{6\Delta} \end{aligned}$$

resources and we have $n = 2\Delta\log\mu$ tasks. It is straightforward to show the existence of a constant $\alpha \leq 7$ with $ns \leq \mu^{\alpha\Delta}$, hence

$$\log(ns) \leq \alpha\Delta\log\mu = \alpha\Delta K,$$

which together with the assumption that C, and consequently Δ is fix, imply

$$b_i = \Omega(K) = \Omega(\log(ns)).$$

It remains to show that there is a fractional schedule of size Δ and an integral schedule of size $\Delta + 1$. Since the chromatic index of G is $\Delta + 1$, we can define an integral schedule of this size as in the proof of Claim 1. Furthermore, it is easily checked that the setting $x_{Tz} = \frac{1}{\Delta}$ for all tasks T and times z defines a fractional schedule of size Δ. □

Finally, if we allow that some resource requirements are fractional numbers, we have:

Theorem 3.3 *Under the assumptions that there exist a fractional schedule of size $C \geq 3$ and an integral schedule of size $C+1$, $b_i \geq 6\lceil\log(4C(s+1))\rceil$ for all resource bounds, C is fixed and $R_i(j) \in \{0, \frac{1}{2}, 1\}$, it is NP-complete to decide whether or not there exists an integral schedule of size C.*

□

Proof: Once again we reduce the problem the the chromatic index problem and argue as in the proof of theorem 3.1. Let $G = (V, E)$ be a graph. Put $K = 100\log(\mu)$ and $\mu = |E|$. For every edge $e \in E$ we consider $2K$ red and $2K(\Delta - 1)$ blue tasks

$$T_1^r(e), \ldots, T_{2K}^r(e) \quad \text{and} \quad T_1^b(e), \ldots, T_{2(\Delta-1)K}^b(e).$$

Consider $2\mu K\Delta$ identical processors. For every node $v \in V$ let R_v be a resource with bound $2K$. Its resource requirement is

$$R_v(j) = \begin{cases} 1 & \text{if } T_j = T_j^r(e) \\ & \text{and } e \text{ contains } v \\ 0 & \text{else.} \end{cases}$$

We follow the argumentation of Theorem 3.2 and define one more resource type to enforce that all red task corresponding to the same edge are scheduled at the same time. Furthermore, we have to introduce some resource requirements of value $\frac{1}{2}$.

First we forbid that more than $2K$ tasks corresponding to the same edge e can be scheduled at the same time: Let R_e be a resource with bound $2K$ and resource requirement

$$R_e(j) = \begin{cases} 1 & \text{if } T_j = T_i^r(e) \text{ or } T_j = T_i^b(e) \\ 0 & \text{else.} \end{cases}$$

Let $\mathcal{T}$ denote the set of all tasks, $\mathcal{T}_r$ the blue tasks, $\mathcal{T}_b$ the blue task, $\mathcal{T}_r(e)$ the red tasks corresponding to an edge e, and $\mathcal{T}_b(e)$ the red tasks corresponding to an edge e.

The resource with fractional requirements is:

For every red task $T_i^r(e)$, choose exactly one other red task $g(T_i^r(e))$ corresponding to e as follows.

$$g(T_i^r(e)) \quad = \quad T_{i+1}^r(e) \quad \text{for } i < 2K$$

and

$$g(T_{2K}^r(e)) \quad = \quad T_1^r(e).$$

Let us call $g(T_i^r(e))$ the *buddy* of $T_i^r(e)$. For every red task $T_i^r(e)$ let $R_{i,e}$ be a resource of bound K. The requirements are

$$R_{i,e}(j) = \begin{cases} 1 & \text{if } T_i^r(e) = T_j^r(e) \\ \frac{1}{2} & \text{if } T_j \text{ is a red task,} \\ & \quad \text{corresponding to the edge } e, \\ & \text{but is not the buddy of } T_i^r(e) \\ 0 & \textit{else} \end{cases}$$

Claim: G has an edge coloring with Δ colors, if and only if the above defined scheduling problem has a schedule of length Δ.

Suppose that the edges of G can be colored with Δ colors. Let Z be the time set $Z = \{1, \ldots, \Delta\}$. For each $e \in E$ schedule all red tasks $T_i^r(e)$ at the time corresponding to the color of e, say $z_e \in Z$ and schedule the blue tasks $T_i^b(e)$ in packets of $2K$ at the remaining times $z \neq z_e$, $z \in Z$. (See figure 4)

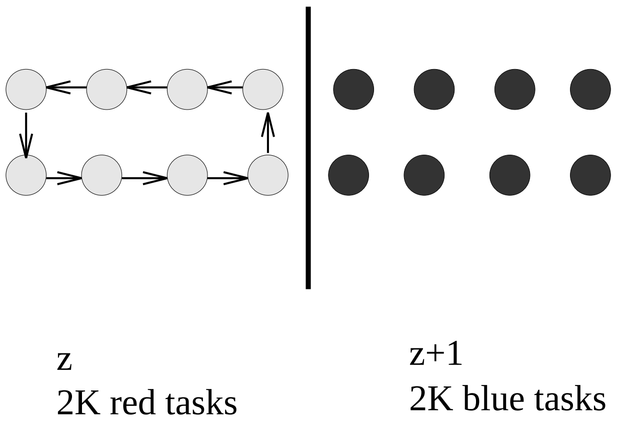

Figure 4: A way to schedule the red and blue tasks

The resource requirments are satisfied: set $x_{jz} = 1$, if the task T_j, $T_j \in \mathcal{T}$ is scheduled at time z, and 0 else. For the resources of type R_e we have

$$\sum_{T_j \in \mathcal{T}} R_e(j) x_{jz} = 2K.$$

Now fix an arbitrary $e \in E$. For every resource $R_{i,e}$ corresponding to a red task $T_{j_0}^r(e)$ we have for the scheduling time z_e

$$\sum_{T_j \in \mathcal{T}} R_{i,e}(j) x_{jz_e} = \sum_{T_j \in \mathcal{T}_r(e) - \{g(T_{j_0}^r(e))\}} R_{i,e}(j) x_{jz_e} = 1 + \frac{1}{2}(2K - 2) = K.$$

and for all other $z \neq z_e$, $z \in Z$

$$\sum_{T_j \in \mathcal{T}} R_{i,e}(j) x_{jz} = \sum_{T_j \in \mathcal{T}_b(e)} R_{i,e}(j) x_{jz} = \frac{1}{2}(2K) = K.$$

This proves the first part of the claim.

Suppose that we have a feasible schedule of length Δ. We show that it is impossible to schedule the red tasks corresponding to the same edge at different times. Then we can again argue as in the proof of Theorem 3.1 taking the scheduling time of the red tasks corresponding to an edge as its color and are done.

Assume for a moment that there is an edge $e_0 \in E$ and a time z_0 in a schedule of size Δ at which only $I < K$ red tasks corresponding to e_0. Then there must be exactly $2K - I$ blue tasks being scheduled at time z_0 in order *not* to violate the constraint of resource R_e at any time. Since there are strictly less than K red tasks in $\mathcal{T}_r(e)$ scheduled at time z_0, there is a task $T_{i_0}^r(e)$, whose buddy is *not* scheduled at time z_0. In fact there are at least two such tasks! Then we have for resource R_{i_0,e_0} at time z_0:

$$\sum_{T_j \in Te} R_{i_0,e_0}(j) x_{jz_0} = \sum_{T_j \in \mathcal{T}_r(e) \cup \mathcal{T}_b(e)} R_{i_0,e_0}(j) x_{jz_0}$$

$$\begin{aligned} &= 1 + \frac{1}{2}((2K - I) + (I - 1)) \\ &= K + \frac{1}{2} > K, \end{aligned}$$

and the schedule requires more than K units of resource R_{i_0,e_0} in contradiction to the feasibility assumption. Hence the red tasks corresponding to the edge must be scheduled at the same time.

We must show $b_i \geq 6\lceil \log(4C(s+1)) \rceil$ for all resource bounds. We introduced $n = 2K\mu\Delta$ tasks and $s = \nu + \mu + K\mu$ resources. Using $K = 48\log(\mu)$ the estimate $s+1 \leq \mu^6$ holds and with $C = \Delta$ we continue

$$\begin{aligned} 6\lceil \log(4C(s+1)) \rceil &\leq 6\log(4\Delta\mu^6) \\ &\leq 6\log(\mu^8) \\ &= K = b_i. \end{aligned}$$

Finally, one may check as in the proof of Theorem 3.2 that there is a fractional schedule of size Δ and an integral schedule of size $\Delta + 1$.

□

With $K = \Delta^c$ for a constant c satisfying $\Delta^c > 3\Delta(\Delta+1)\lceil \log(4\Delta(s+1)) \rceil$ the proof of Theorem 3.3 implies

Theorem 3.4 *Under the assumptions that there is a fractional schedule of size $C \geq 3$ and an integral schedule of size $C+1$, $b_i \geq 3C(C+1)\lceil \log(4C(s+1)) \rceil$ for all resource bounds, C is fixed and $R_i(j) \in \{0, \frac{1}{2}, 1\}$, it is NP-complete to decide whether or not there exists an integral schedule of size C.*

□

4 Multidimensional Bin Packing

Consider the general multidimensional bin packing problem $BIN(l,d)$ as defined in the introduction. Since $BIN(l,d)$ is nothing else than the resource constrained scheduling problem with d resources, resource bounds $b_i = l$ for all i and zero start times, all the approximation and non-approximability results proved for scheduling are valid for $BIN(l,d)$, too. To get a feeling for good approximations we briefly recall the known asymptotic approximation results. Let L_{opt} be the optimal bin number. The First-Fit heuristic gives a 2-factor approximation for $BIN(1,1)$, and using this result one can construct a solution for $BIN(1,d)$ within $2dL_{opt}$ in polynomial time [VeLu81]. A similar argumentation shows a $(1+\frac{1}{l})d$ factor approximation for $BIN(l,d)$. Thus, good approximations must beat these factors. The results of Garey, Graham, Johnson and Yao [GaJo79] for resource constrained scheduling (which we have already discussed in the introduction) applied to bin packing show the existence of an integer N_0 such that

$$L_{FFD} \leq (d + \frac{1}{3})L_{opt}$$

for all instances with $L_{opt} \geq N_0$. By de la Vega and Luecker [VeLu81] for all $0 < \epsilon < 1$ there is a linear time algorithm A_ϵ and an integer $N_\epsilon \geq 1$ such that

$$L_{A_\epsilon} \leq (d + \epsilon)L_{opt}$$

for all instances with $L_{opt} \geq N_\epsilon$. Our results imply for $BIN(l,d)$ an approximation within a factor independent of d and l: Let L_R be the minimum number of bins, if fractional packing is allowed, i.e.

$$L_R = \lceil \max_{1 \leq i \leq d} \frac{1}{l} \sum_{j=1}^{n} v_{ij} \rceil.$$

Theorem 2.5 implies

Theorem 4.1 *Let $\epsilon > 0$ with $(1/\epsilon) \in I\!N$. If $l \geq \frac{3(1+\epsilon)}{\epsilon^2}\lceil \log(4nd)) \rceil$, then we can find in $O(dn^3 \log(nd))$ time a bin packing with L bins such that $L \leq \lceil (1+\epsilon) L_{opt} \rceil$.*

□

Proof: Since we get the optimal fractional solution for free, we can apply Theorem 2.5 without using LP-algorithms. So, only the time for derandomization counts.

□

For $\epsilon = 1$ Theorem 4.1 implies

Corollary 4.2 *If $l \geq 6\lceil \log(4nd) \rceil$, then we can find in $O(dn^3 \log(nd))$ time a bin packing with L bins such that $L \leq 2L_{opt}$.*

□

and for $\epsilon = L_R^{-2}$ we get

Corollary 4.3 *If $l \geq 3L_R(L_R+1)\lceil \log(4nd)) \rceil$, then we can find in $O(dn^3 \log nd)$ time a bin packing with L bins such that $L \leq L_{opt} + 1$.*

□

Since we allow really large bin sizes, one might wonder if the class of problems is interesting enough. The following negative result, which is follows from the proof of Theorem 3.4, gives the answer.

Theorem 4.4 *Even if $l = \Omega(L_R^2 \log(nd))$, then it is NP-complete to decide whether $L_{opt} = L_R$ or $L_{opt} = L_R + 1$.*

□

5 Parallel Scheduling and Bin Packing

There is no obvious way to achieve the approximation guarantee of Theorem 2.5 in NC. In this section we will show that at least in some special cases there is a NC approximation algorithm. The algorithm is based on the method of $\log^c n$-wise independence. The important steps are:

1. Fractional Scheduling in Parallel. We wish to apply randomized rounding using $\log^c(n)$-wise independence and therefore first have to generate an appropriate probability distribution in NC. Sequentially this is easy: solve the linear programming relaxation of the integer programming formulation of our scheduling problem and the fractional assignments of tasks to times will define the right distribution. Unfortunately, linear programming is P-complete ! But fortunately, due to the fact that the start times are zero, we have (as for the bin packing problem) a formula for the optimal fractional schedule.

2. Schedule Enlargement. Having found the fractional optimal schedule of length say C, we enlarge the makespan to $2C$ and define fractional assignments as in Lemma 2.4.

3. Rounding in NC is performed with the parallel conditional probability method for multivalued random variables (Theorem 6.2).

First we generate a fractional, optimal schedule in NC. Let x_{jz} be a 0–1 variable which is 1 if T_j is scheduled at time t, and is 0 else ($z \in \{1, \ldots, n\}$). [4]

Lemma 5.1 *Define*

$$C' = \max_{1 \leq i \leq s+1} \left\{ \frac{1}{b_i} \sum_{j=1}^{n} R_i(j) \right\}$$

[4]Note that it suffice to deal with the time intervall $[1, \ldots, n]$, since any one-to-one assignment of the n tasks to the n times is feasible.

and $C = \lceil C' \rceil$. Then C is the length of an optimal fractional schedule and the corresponding task-time assignments $\widetilde{x}_{jt}$ are

$$\widetilde{x}_{jt} = \begin{cases} \frac{1}{C} & \forall\, T_j \in \mathcal{T}, 1 \le t \le C \\ 0 & \textit{else.} \end{cases}$$

$\square$

Proof: Straightforward.

$\square$

Proposition 5.2 *There is a NC-algorithm computing an optimal fractional schedule on $O(n+s)$ $EREW-PRAM$-processors in $O(\log s \log\log s + \log n)$ time.*

$\square$

Proof: Using $O(n)$ $EREW-PRAM$-processors for each resource R_i we can compute $\frac{1}{b_i}\sum_{j=1}^{n} R_i(j)$ in $O(\log n)$ time. With standard sorting algorithms ([JaJa92], Remark 4.4 and Corollary 4.2) we can find

$$C' = \max_{1\le i\le s+1}\left\{\frac{1}{b_i}\sum_{j=1}^{n} R_i(j)\right\}$$

in $O(\log s \log\log s)$ time on $O(s)$ $EREW-PRAM$-processors.

$\square$

The next step is the enlargement of the fractional makespan to $\{1,\ldots,2C\}$ as in Lemma 2.4. The fractional assignments are defined as follows. For every time $z' \in \{1,\ldots,C\}$ introduce a second time $z'' = C + z'$ and set:

$$\widetilde{x}_{jz} = \frac{1}{2C} \quad \text{for } 1 \le j \le n,\ 1 \le z \le 2C.$$

Clearly this new fractional schedule has size $2C$ and by Lemma 2.4 we have for every $z \in \{1,\ldots,2C\}$ and every resource $R_i \in \mathcal{R}$

$$\sum_{j=1}^{n} R_i(j)\widetilde{x}_{jz} \le \frac{b_i}{2}. \tag{7}$$

The main result of this section is

Theorem 5.3 *Let $\tau \ge \frac{1}{\log n}$ and suppose that $b_i \ge 2n^{\frac{1}{2}+\tau}\sqrt{\log 2n(s+1)}$ for all i. Then there is a NC-algorithms that runs on $O(n^2(ns)^{1+\frac{1}{\tau}})$ parallel processors and finds in $O(\log n \log^3(ns))$ time a schedule of size at most $2C_{opt}$.*

$\square$

Proof: For the moment let k be an arbitrary integer. We will fix k in the proof. In order to apply Theorem 6.2 (the NC version of the conditional probability method for *multivalued* random variables), put $M := s+1$, $c_{ij} := R_i(j)$ and $d = 2C$. Then $N = dMn^k$. Define k-wise independent, uniformly distributed random variables $X_1,\ldots,X_n$ with values in $\{1,\ldots,d\}$ as in Theorem 6.2 and with the notation there [5] we define for a resource R_i and a time $z \in \{1,\ldots,d\}$ the function f_{iz} by

$$f_{iz}(X_1,\ldots,X_n) := \sum_{j=1}^{n} R_i(j)(X_{jz} - \frac{1}{d})$$

[5] In Theorem 6.2 the X_j's take on values in the set $\{0,\ldots,d-1\}$ But this is only a convention.

and

$$F(X_1,\dots,X_n) := \sum_{iz} f_{iz}(X_1,\dots,X_n).$$

By Theorem 6.2 we can construct $x_1,\dots,x_n$, where $x_j \in \{1,\dots,d\}$ for all j such that

$$F(x_1,\dots,x_n) \le \mathbb{E}(F(X_1,\dots,X_n)) \tag{8}$$

holds, using

$$O(N) = O(dMn^k) = O(n^2(ns)^{1+\frac{1}{\tau}})$$

parallel processors in

$$O(k\log n \log N) = O(\log n \log^3(ns))$$

time. To complete the proof we must show that the assignment of task T_j to time x_j for all j indeed is a feasible schedule. By definition of the function F we have for all R_i and $z \in \{1,\dots,d\}$

$$\sum_{j=1}^{n} R_i(j)(x_{jz} - \frac{1}{d}) \le (F(x_1,\dots,x_n))^{\frac{1}{k}} \le (\mathbb{E}(F(X_1,\dots,X_n))^{\frac{1}{k}}. \tag{9}$$

Utilizing the fact that the random variables X_{jz} are binomially distributed and putting $k = \lceil 2\frac{\log 2n(s+1)}{\tau \log n}\rceil$ the proof of Corollary 2.6 in [BeRo91] shows

$$(\mathbb{E}(F(X_1,\dots,X_n))^{\frac{1}{k}} \le n^{\frac{1}{2}+\tau}\sqrt{\log 2n(s+1)}. \tag{10}$$

(9), (10) and (7) imply for all R_i and $z \in \{1,\dots,d\}$

$$\begin{aligned}\sum_{j=1}^{n} R_i(j)x_{jz} &= \sum_{j=1}^{n} R_i(j)(x_{jz} - \frac{1}{d}) + \sum_{j=1}^{n} R_i(j)\frac{1}{d}\\ &\le n^{\frac{1}{2}+\tau}\sqrt{\log 2n(s+1)} + \frac{b_i}{2}\\ &\le b_i,\end{aligned}$$

and the theorem is proved.

□

For multidimensional bin packing we get

Corollary 5.4 *Let $\tau \ge \frac{1}{\log n}$ and suppose that $l \ge 2n^{\frac{1}{2}+\tau}\sqrt{\log(2nd)}$. Then there is a NC-algorithms for the problem $BIN(l,d)$ that runs on $O(n^2(nd)^{1+\frac{1}{\tau}})$ parallel processors and finds in $O(\log n \log^3(nd))$ time a packing with L bins such that $L \le 2L_{opt}$.*

□

Remark 5.5 The reason why we have to assume $b_i = \Omega(n^{\frac{1}{2}+\tau}\sqrt{\log ns})$ is due to the estimation of the k-th moments. Improvements of this method with the goal to show a 2-factor (or even better) NC-algorithm under the weaker assumption $b_i = \Omega(\log ns)$ would be interesting. This would match the presently best sequential approximation guarantees.

Conclusion

• We proved that there are polynomial-time algorithms approximating the optimum of a resource constrained scheduling problem, where the makespan is to be minimized. On the other hand we showed that it is NP-complete to guarantee better approximations. We showed that the problem of finding the chromatic index of a graph is a special case of a resource constrained scheduling problem.

• We hope that our rounding technique can be applied to other problems of similar flavour.

Acknowledgement

The first author thanks Professor László Lovász for helpful discussions during a research stay at Yale University in Spring 1994, where a part of this paper has been written. We would also like to thank Andreas Krämer for helpful discussions.

References

[ABI86] N.Alon, L Babai, A. Itai; *A fast and simple randomized algorithm for the maximal independent set problem.* J. Algo., 7 (1987), 567 - 583.

[ASE92] N. Alon, J. Spencer, P. Erdős; *The probabilistic method.* John Wiley & Sons, Inc. 1992.

[AnVa79] D. Angluin, L.G. Valiant: *Fast probabilistic algorithms for Hamiltonion circuits and matchings.* J. Comp. Sys. Sci., Vol. 18, (1979), 155–193.

[BESW93] J. Błazewicz, K. Ecker, G. Schmidt, J. Węglarz; *Scheduling in computer and maufacturing systems.* Springer-Verlag, Berlin (1993).

[BeRo91] B. Berger, J. Rompel; *Simulating ($log^c n$)-wise independence in NC.* JACM, 38 (4), (1991), 1026 – 1046.

[Li82] J. H. van Lint; *Introduction to Coding Theory.* Springer Verlag New York, Heidelberg, Berlin (1982).

[VeLu81] W. F. de la Vega, C. S. Luecker; *Bin packing can be solved in within $1-\epsilon$ in linear time.* Combinatorica, 1 (1981), 349 - 355.

[GaJo79] M. R. Garey, D. S. Johnson; *Computers and Intractability.* W. H. Freeman and Company, New York (1979).

[GGJY76] M. R. Garey, R. L. Graham, D. S. Johnson, A.C.-C. Yao; *Resource constrained scheduling as generalized bin packing.* JCT Ser. A, 21 (1976), 257 - 298.

[GLS88] M. Grötschel, L. Lovász, A. Schrijver; *Geometric algorithms and combinatorial optimization.* Springer-Verlag (1988).

[Hol81] I. Holyer; *The NP-cpmpleteness of edge coloring.* SIAM J.Comp., 10 (4), (1981), 718 - 720.

[JaJa92] J. JaJa; *An Introduction to Parallel Algorithms.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1992.

[Kr75] K. L. Krause, V. Y. Shen, H .D. Schwetmann; *Analysis of several job-scheduling algorithms for a model of multiprogramming computer systems.* JACM 22 (1975) 522–550. Erratum: JACM 24, (1977), p. 527.

[LST90] J. K. Lenstra, D. B. Shmoys, E. Tardos; *Approximating algorithms for scheduling unrelated parallel machines.* Math. Programming, 46, (1990), 259 – 271.

[LiVi92] J.-H. Lin, J. S. Vitter; *ϵ-approximations with minimum packing constraint violation.* Proceedings 24th Annual ACM Symposium on the Theory of Computation (1992), Victoria, B.C., Canada, 771 - 782.

[McSo77] F.J. MacWilliams, N.J.A. Sloane; *The theory of error correcting codes.* North Holland, Amsterdam, (1977).

[McD89] C. McDiarmid; *On the method of bounded differences.* Surveys in Combinatorics, 1989. J. Siemons, Ed.: London Math. Soc. Lectures Notes, Series 141, Cambridge University Press, Cambridge, England 1989.

[MNN89] R. Motwani, J. Naor, M. Naor; *The probabilistic method yields deterministic parallel algorithms.* Proceedings 30the IEEE Conference on Foandation of Computer Science (FOCS'89), (1989), 8 – 13.

[RS83] H. Röck, G. Schmidt; *Machine aggregation heuristics in shop scheduling.* Math. Oper. Res. 45(1983) 303–314.

[Sp87] J. Spencer; *Ten lectures on the probabilistic method.* SIAM, Philadelphia (1987).

[SrSt94] A. Srivastav, P. Stangier; *Algorithmic Chernoff-Hoeffding inequalties in integer programming.* to appear in Random Structures & Algorithms, January 1996.

(preliminary version in: Du, Zhang (eds.), *Proccedings of the 5th Annual Inernational Symposium on Algorithms and Computation (ISAAC'94), pages 264 - 234, Lecture Notes in Computer Science, Vol 834, Springer Verlag.*)

[Ta86] E. Tardos; *A strongly polynomial algorithm to solve combinatorial linear programs.* Oper. Res. 34 (1986), 250 - 256.

[Viz64] V. G. Vizing; *On an estimate of the chromatic class of a p-graph.* (Russian), Diskret. Analiz. 3 (1964), 25 - 30.

6 Appendix: Multivalued Random Variables and $\log^c n$-wise Independence

In this section we extend the well-known derandomization technique of $\log^c n$-wise due to Berger, Rompel [BeRo91] and Motwani, Naor, Naor [MNN89] from the case of uniformly distributed $0-1$ to uniformly distributed multivalued random variables.

Let $n = 2^{n'} - 1$ for some $n' \in \mathbb{N}$. A representation of $GF(2^{n'})$ as a n'-dimensional algebra over $GF(2)$ can be explicitly constructed using irreducible polynomials, for example the polynomials given in [Li82], Theorem 1.1.28. Let $b_1, \ldots, b_n$ be the n non-zero elements of $GF(2^{n'})$ in such an irreducible representation and let $A = (a_{ij})$ the following $n \times \lceil \frac{k-1}{2} \rceil$ matrix over $GF(2^{n'})$

$$A = \begin{bmatrix} 1 & b_1 & b_1^3 & \cdots & b_1^{k-1} \\ 1 & b_2 & b_2^3 & & b_2^{k-1} \\ 1 & b_3 & b_3^3 & & b_3^{k-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & b_n & b_n^3 & & b_n^{k-1} \end{bmatrix}.$$

A can be viewed as a $n \times \ell$ matrix over $GF(2)$ with $\ell = 1 + \lceil \frac{k-1}{2} \rceil \lceil \log(n+1) \rceil = O(k \log n)$. The matrix A is well-known in coding theory. It is the parity check matrix of binary BCH codes and every set of k row vectors of A is linear independent over $GF(2)$ [McSo77]. Alon, Itai and Babai [ABI86] showed that k-wise independent $0-1$ random variables can be constructed from mutually independent $0-1$ random variables using a BCH-matrix. The extension to multivalued random variables goes as follows.

Let $Y_1, \ldots, Y_l$ be independent and uniformly distributed random variables with values in $\Omega = \{0, \ldots, d-1\}$, $d \in \mathbb{N}$. Let Y be the vector $Y = (Y_1, \ldots, Y_l)$. Define Ω-valued random variables $X_1, \ldots, X_n$ by $X_i = (BY)_i \bmod d$ for all $i = 1, \ldots, n$. With $X = (X_1, \ldots, X_l)$ we briefly write $X = BY \bmod d$.

Theorem 6.1 *The random variables $X_1, \ldots, X_n$ are k-wise independent and uniformly distributed.*

$\square$

Proof: We modify the proof of Theorem [BeRo91]. Choose $J \subseteq \{1, \ldots, n\}$ with $|J| = k$. Let $x \in \Omega^k$ be an arbitrarily choosen but fixed vector. Set $X_J = (X_j)_{j \in J}$. We must show

$$\mathbb{P}[X_J = x] = d^{-k}. \tag{11}$$

Let A_J be the submatrix of A with row indices from J and set $B_J = (A^\top)_J$. B_J is a $l \times k$ matrix over $GF(2)$. As in the proof of Theorem 6.1 we extend B_J to an invertible $l \times l$ matrix C over $GF(2)$. Define

$$\Omega_x = \{x' \in \Omega^l ; x'_i = x_i \text{ for } i = 1, \ldots, k\}. \tag{12}$$

then $|\Omega_x| = d^{l-k}$, and we have

$$\begin{aligned}
\mathbb{P}[X_J = x] &= \mathbb{P}[BY \bmod d = x] \\
&= \mathbb{P}[CY \bmod d \in \Omega_x] \\
&= \sum_{x' \in \Omega_x} \mathbb{P}[CY - x' = 0 \bmod d] \\
&= \sum_{x' \in \Omega_x} \mathbb{P}[Y = C^{-1}x' \bmod d] \\
&= \sum_{x' \in \Omega_x} d^{-l} \qquad \text{(the } Y_i\text{'s are independent !)} \\
&= \frac{|\Omega_x|}{d^l} = \frac{d^{l-k}}{d^l} = d^{-k}.
\end{aligned}$$

$\square$

For the following class of functions F, arising in the analysis of resource constrained scheduling, a NC algorithm finding points below or above the expectation $\mathbb{E}(F(X_1, \ldots, X_n))$ can be derived. We keep the notation, i.e. let k, d be as above.

Let $\vec{v}_1, \ldots, \vec{v}_n$ be vectors in $\{0,1\}^d$ and let v_{jz} denote the z-th component of $\vec{v}_j$. For $x_j \in \Omega = \{0, \ldots, d-1\}$ let $\vec{x}_j$ be the vector that has a 1 in its x_j-th coordinate and 0 elsewhere. Let (c_{ij}) be $M \times n$ matrix with rational $0 \leq c_{ij} \leq 1$. For $i = 1, \ldots, M$ and $z \in \Omega$ define the functions

$$g_{iz}(x_1, \ldots, x_n) = \sum_{j=1}^{n} c_{ij}(x_{jz} - \frac{1}{d}), \tag{13}$$

and

$$F(x_1, \ldots, x_n) = \sum_{iz} g_{iz}(x_1, \ldots, x_n). \tag{14}$$

Theorem 6.2 *Let $d, k, M, n \in \mathbb{N}$ and $N = dMn^k$. Let $X_1, \ldots, X_n$ be k-wise independent random variables with values in $\Omega = \{0, \ldots, d-1\}$ defined as in Theorem 6.1 and let F as in (14). Then with $O(N)$ parallel processors we can construct $x_1, \ldots, x_n \in \Omega$ and $y_1 \ldots, y_n \in \Omega$ in $O(k \log n \log N)$-time such that*

$$\begin{array}{lrcl}
(i) & F(x_1, \ldots, x_n) & \geq & \mathbb{E}(F(X_1, \ldots, X_n)) \\
(ii) & F(y_1, \ldots, y_n) & \leq & \mathbb{E}(F(X_1, \ldots, X_n)).
\end{array}$$

$\square$

Proof: It suffices to prove (i). Let I be the set of all k tupels $(\alpha_1, \ldots, \alpha_k)$ with $\alpha_j \in \{1, \ldots, n\}$. Set

$$g_j^{(iz)}(x_1, \ldots, x_n) = c_{ij}(x_{jz} - \frac{1}{d})$$

and define for a k-tupel $\alpha \in I$,

$$g_\alpha^{(iz)}(x_1, \ldots, x_n) = \prod_{j=1}^{k} c_{i\alpha_j}(x_{\alpha_j, z} - \frac{1}{d}).$$

Then

$$F(x_1, \ldots, x_n) = \sum_{iz} \sum_{\alpha \in I} g_\alpha^{(iz)}(x_1, \ldots, x_n).$$

The random variables $X_1, \ldots, X_n$ by definition have the form

$$X_i = (BY)_i \bmod d,$$

where the Y_i's are uniformly distributed Ω valued random variables, $B = A^T$ and A is the $n \times \ell$ parity check matrix of BCH codes defined in Theorem 6.1. Therefore we may restrict us to the assignments of the Y_i's. The conditional probability method then goes as follows:

Suppose that for some $1 \le t \le \ell$ we have computed the values

$$Y_1 = y_1, \ldots, Y_{t-1} = y_{t-1},$$

then choose for Y_t the value $y_t \in \Omega$ that maximizes the function

$$w \to \mathbb{E}(F(X_1, \ldots, X_n) \mid y_1, \ldots, y_{t-1}, Y_t = w). \tag{15}$$

After ℓ steps we have $Y = y$ for some $y \in \Omega$ and the vector $x = (x_1, \ldots, x_n)$ with $x_j = (By)_j \bmod d$ is the desired solution. Let $\vec{Y_t} = (Y_1, \ldots, Y_t)$ and $\vec{y_t} = (y_1, \ldots, y_t)$. We are done, if we can compute the conditional expectations

$$\mathbb{E}(F(X_1, \ldots, X_n) \mid \vec{Y_t} = \vec{y_t})$$

within the claimed time. By linearity of expectation, it is sufficient to compute for each tripel (i, z, α), $1 \le i \le m$, $z \in \Omega$, $\alpha \in I$ the conditional expectations

$$\mathbb{E}(g_\alpha^{(iz)}(X_1, \ldots, X_n) \mid \vec{Y_t} = \vec{y_t}). \tag{16}$$

Because the X_j's are k-wise independent, the problem reduces to the computation of terms fo the form

$$\mathbb{E}(g_j^{(iz)}(X_1, \ldots, X_n) \mid \vec{Y_t} = \vec{y_t})$$

for $j \in \{1, \ldots, n\}$. Now

$$\mathbb{E}(g_j^{(iz)}(X_1, \ldots, X_n) \mid \vec{Y_t} = \vec{y_t})) = (1 - \frac{1}{d})\mathbb{P}[X_j = z \mid \vec{Y_t} = \vec{y_t})] - \frac{1}{d}\mathbb{P}[X_j \neq z \mid \vec{Y_t} = \vec{y_t})].$$

It is straightforward to verify that $\mathbb{P}[X_j = z \mid \vec{Y_t} = \vec{y_t})]$ can be computed in constant time, because the only possible values it attains are 0, 1 or $\frac{1}{d}$. The same holds for $\mathbb{P}[X_j \neq z \mid \vec{Y_t} = \vec{y_t})]$. In other words, for fixed t, i, z, α and given $y_1, \ldots, y_{t-1}$ we can compute

$$\mathbb{E}(g_\alpha^{(iz)}(X_1, \ldots, X_n) \mid \vec{Y_t} = \vec{y_t})$$

in $O(k)$ time. Thus, for every possible value $y_t \in \Omega$ we can compute

$$\mathbb{E}(F(X_1, \ldots, X_n) \mid \vec{Y_t} = \vec{y_t})$$

in $O(\log N)$ time using $O(N)$ parallel processors. That value of y_t which maximizes (15) can be computed finding the maximum of the d conditional expectations with standard sorting algorithms (see [JaJa92], Remark 4.4 and Corollary 4.5) in $O(\log d \log\log d)$ time. The total running time over all ℓ steps then is

$$O(\ell(\log N + \log d \log\log d)) = O(k \log n \log N)$$

and the theorem is proved.

$\square$